web & mobile DEVELOPER

webundmobile.de

Alternative NoSQL

Besserer Durchsatz, höhere Verfügbarkeit und einfachere Erweiterbarkeit: Die Vorteile von NoSQL-Datenbanken S. 14

INFO-Programm gemäß § 14 JuSchG

Ausgabe 4/16

Deutschland 14,95 EUR

CH: 29,90 CHF
A, B, NL, L: 16,45 EUR

4 198255 014951 04

Auf der Heft-DVD:

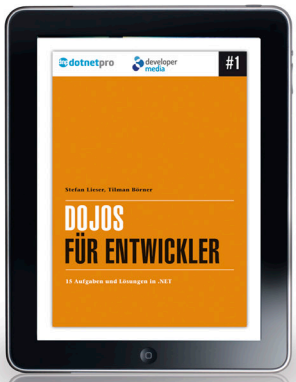
PDF-Jahresarchive 2013, 2014 & 2015

sowie eine Auswahl leistungsfähiger Tools für Entwickler S. 50

developer media Shop

Upgrades für Ihr Entwickler-Know-How

Ab sofort in unseren Shops erhältlich!



https://shop.dotnetpro.de

https://shop.webundmobile.de

Alternative

Big Data und relationale Datenbanken passen nicht immer optimal zusammen. Die Alternative heißt NoSQL.

Neue Anwendungstypen aus den Bereichen Online Analytical Processing (OLAP), Data Mining, Big Data und vor allem aus dem mobilen Web/Internet führten in den letzten Jahren zu einem enormen Wachstum an Datenbeständen und bei der Anzahl an Benutzern. Anwendungstypen mit massivem Datenaufkommen und ständig steigender Benutzeranzahl erfordern den Einsatz gänzlich neuer Typen von Datenbanken. Die Gesamtheit dieser neuen Datenbanktypen bezeichnet man im Unterschied zu den traditionellen relationalen Datenbanken als NoSQL-Datenbanken, wobei das No nicht nein bedeutet, sondern Not only. Unser Autor Frank Simon macht sich im Heft-Schwerpunkt auf die Spur der NoSQL-Datenbanken und beleuchtet die Szene. Der Artikel ab Seite 14 bildet zugleich den Auftakt zu einer mehrteiligen Serie zum Thema NoSQL.

»Neue Anwendungstypen führten zu einem enormen Wachstum an Datenbeständen.«

Neben den bei Desktop-Rechnern normalerweise benutzten Inputgeräten Maus und Tastatur gibt es die Touchscreens, die mit Fingern bedient werden. Das stellt Webdesigner und Entwickler von GUIs vor neue Herausforderungen. Dr. Florence Maurice erläutert in einem Artikel ab Seite 24, was man hierbei beachten muss.

Da Facebook mit Hack/HHVM eine konkurrenzfähige Alternative zu PHP 5.6 bieten konnte, schien eine gründliche Umarbeitung von PHP unvermeidlich. Mit PHP 7 ist diese jetzt da. In einem ausführlichen Artikel ab Seite 88 untersuchen Filipe Pereira Martins und Anna Kobylinska die aktuelle Konkurrenzsituation HHVM vs. PHP 7.

Ihr Max Bold
chefredakteur@maxbold.de

Philip Ackermann

erläutert das IndexedDB API im Detail (S. 36)

Dr. Markus Stäuble

stellt die Programmiersprache R vor (S. 108)

Katharina Sckommodau

diskutiert die Design-Grundsätze digitaler Oberflächen (S. 124)

INHALT

Aktuell

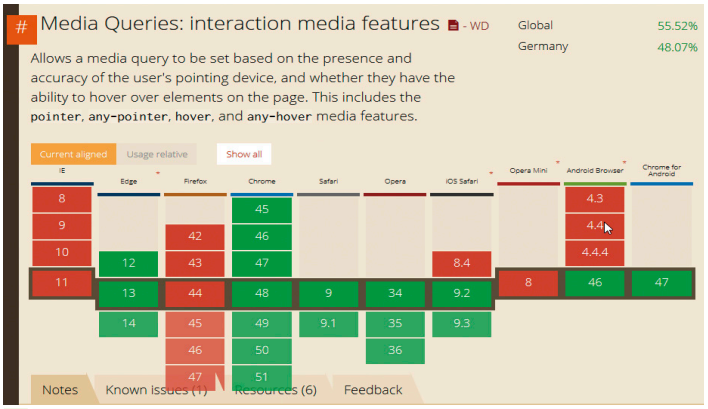
Feature

HTML, CSS & JavaScript

Mobile Development

Der aktuelle Trend: Relationale und NoSQL-Datenbanksysteme bewegen sich aufeinander zu S. 14

Foto: Mathias Vietmeier

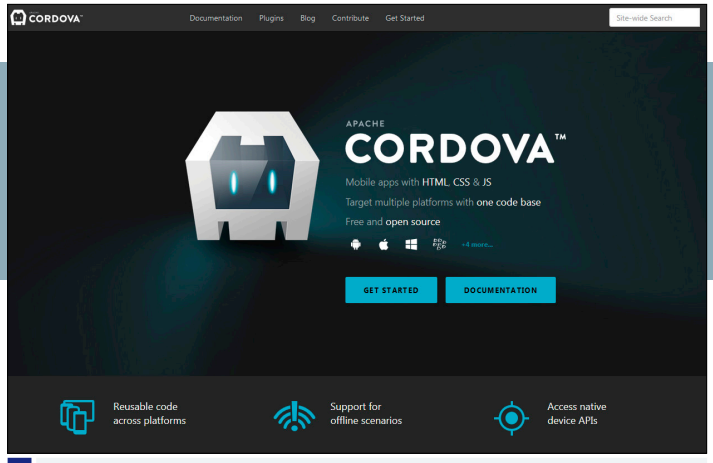


Browserunterstützung: Diese Browser unterstützen die neuen interaktiven Media-Features S. 24

Experten in dieser Ausgabe

Dr. Florence Maurice zeigt in einem Workshop, wie Webdesign für Touchscreens realisiert wird. 24

Patrick Lobacher erläutert, wie man mit dem Cordova-Framework hybride Apps für Android und Co. schreibt. 66



JavaScript: Bei einer Cordova-App stehen dem Entwickler die meisten JavaScript-Methoden zur Verfügung **S. 66**

HHVM vs. PHP 7: In Sachen Performance schenken sich die beiden Kontrahenten PHP 7 und HHVM – eine Eigenentwicklung von Facebook – nicht viel. **S. 88**

Mit Mac OS X stellte Apple das neue Oberflächen-Design Aqua vor **S. 124**

Backend

Beyond Dev

Standards

NEWS & TRENDS

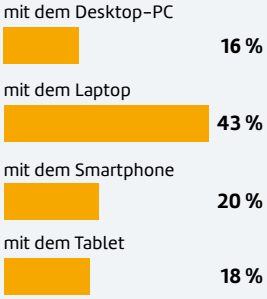
AKTUELLE NEWS AUS DER ENTWICKLERSZENE

Bitkom-Umfrage

Vor allem mit dem Smartphone online

Das Smartphone ist bereits heute für viele Deutsche im privaten Leben das Hauptzugangsgerät zum Internet – und damit beliebter als der stationäre Computer. Das zeigt eine Studie im Auftrag des Digitalverbands Bitkom. Wenn man die Internetnutzer fragt, mit welchem Gerät sie privat hauptsächlich ins Internet gehen, nennen

Privates Surfen



web & mobile developer 4/2016
Quelle: Bitkom.org

nur noch 16 Prozent den Desktop-PC. Spitzenreiter ist der Laptop mit durchschnittlich 43 Prozent. Schon auf dem zweiten Platz folgt das Smartphone mit 20 Prozent. 18 Prozent erklären, dass sie privat vor allem das Tablet zum Surfen nutzen. Eine stetig steigende Qualität der Netze sowie die Verbreitung von Flatrate-Tarifen sorgen darüber hinaus dafür, dass die mobile Internetnutzung immer beliebter wird.

Microsoft Lumia 650

Für Windows-Fans und Geschäftskunden

Nach vielen Gerüchten hat Microsoft mit dem Lumia 650 jetzt ein weiteres Windows-10-Phone vorgestellt. Zielgruppe sind

Windows-10-Phone: Das neue Microsoft Lumia 650

laut Pressemitteilung Windows-Fans und Business-Nutzer.

Das neue Lumia 650 ist schlank, hat ein 5-Zoll-HD-OLED-Display und einen geschliffenen Metallrahmen. In Deutschland soll das Windows-Phone in den Farben Schwarz und Weiß für je 229 Euro angeboten werden. Zum Smartphone für den Geschäftseinsatz sollen es die Sicherheitsmechanismen von Windows 10 sowie die einfache Einrichtung von Office 365 machen.
www.microsoft.com/en/mobile/phone/lumia650/specifications

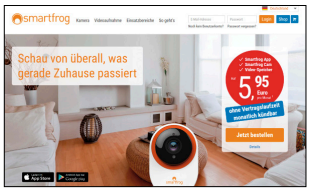
Home Security

Heimüberwachung mit Kamera

Laut einer GfK-Umfrage sind rund 71 Prozent der Deutschen bereit, bei einer einfach zu installierenden und zu bedienenden Sicherheitslösung für ihr Zuhause auch zu bezahlen.

Das Thema häusliche Sicherheit gewinnt angesichts seit Jahren steigender Zahlen bei Wohnungseinbrüchen immer größere Bedeutung. Das Marktforschungsinstitut GfK hat jetzt in einer repräsentativen Umfrage ermittelt, dass die Deutschen auch bereit wären, für den Schutz ihres Zuhauses zu bezahlen, und auch offen sind für IoT-Lösungen, denn schon bereits jeder Vierte (26,8 Prozent) denkt darüber nach, sein Zuhause mit einer Videokamera zu sichern.

Und 70,8 Prozent der Deutschen würden für eine komfortable, dabei aber einfach zu installierende und bedienende Sicherheitslösung monatlich



Sicherheitslösung für das Zuhause sind gefragt.

mindestens 5 Euro bezahlen, wie die jüngste GfK-Umfrage ergab.
www.smartfrog.com

TU München

Nanolaser für schnellere Chips

Physiker an der TU München (TUM) haben einen Nanolaser entwickelt, der direkt auf Silizium-Chips wächst.

Einen Nanolaser, der tausendmal dünner ist als ein Haar, haben Physiker an der Technischen Universität München entwickelt. Dank des ausgetüftelten Verfahrens wachsen die Nanodraht-Laser direkt auf Silizium-Chips. Leistungsfähige photonische Bauelemente lassen sich auf diese Weise kostengünstig herstellen. Damit ist eine Grundvoraussetzung für die künftige schnelle und effiziente Datenverarbeitung mit Licht geschaffen.

Seit Beginn des Computerzeitalters verdoppelt sich die Leistung von Prozessoren durchschnittlich alle 18 Monate.

Doch jetzt stößt die Miniaturisierung der Elektronik an physikalische Grenzen. »Schon heute sind Transistoren nur noch einige Nanometer groß. Reduziert man die Abmessungen noch weiter, steigen die Kosten massiv«, sagt Professor Jonathan Finley, Leiter des Walter-Schottky-Instituts der TUM.

Die Datenübertragung und -verarbeitung mit Licht hat das Potenzial, die bisherigen Grenzen der Elektronik zu überschreiten. Tatsächlich gibt es bereits erste Photonik-Chips aus Silizium. Die Lichtquellen für die Informationsübertragung müssen jedoch durch komplizierte und aufwendige Fertigungsschritte mit dem Silizium verbunden werden. Weltweit suchen Forscher daher nach alternativen Methoden.

Der Durchbruch ist jetzt Forschern an der TU München gelungen: Dr. Gregor Koblmüller vom Lehrstuhl für Halbleiter Quanten-Nanosysteme hat zusammen mit Jonathan Finley ein Verfahren entwickelt, Nanodrahtlaser direkt auf Silizium-

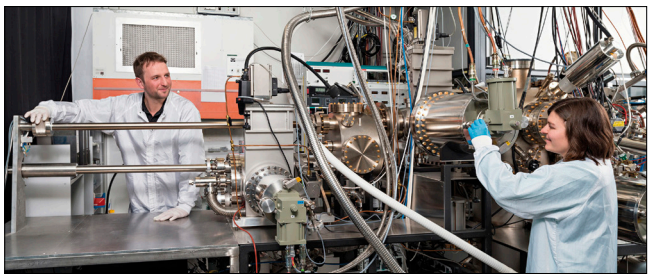
Zahl des Monats

Eine Untersuchung von Veracode beweist: **87 %** aller Andoid-Apps und **80 %** aller Apple-Apps weisen in Sachen Sicherheit gravierende Mängel auf. Am sichersten erwiesen sich Apps, die auf Java oder .NET basierten. Beiden konnten mit etwa 20 Prozent deutlich mehr Sicherheit bieten als alle anderen.

Quelle: Veracode

Chips abzuscheiden. Die Technologie wurde bereits zum Patent angemeldet.

Die Verbindung eines III-V Halbleiters mit Silizium erforderte einiges an Tüftelarbeit: »Die beiden Materialien haben unterschiedliche Gitterabstände und unterschiedliche thermische Ausdehnungskoeffizienten. Das führt zu Spannungen«, erläutert Koblmüller.



Benedikt Mayer und Lisa Janker an der Molekularstrahlepitaxie-Anlage im Walter Schottky Institut der TU München

Dem TUM-Team gelang es, dieses Problem zu umgehen: Die Nanodrähte stehen aufrecht auf dem Silizium, die Grundfläche beträgt dadurch nur noch einige Quadratnanometer. Defekte können die Wissenschaftler so weitestgehend vermeiden.

Doch wie wird ein Nanodraht zum Laser? Um kohärentes Licht zu erzeugen, müssen die Photonen am oberen und unteren Ende des Drahtes reflektiert werden, wodurch sich der Lichtpuls verstärkt, bis er die gewünschte Leistung erreicht hat.

Um diese Bedingungen zu erfüllen, mussten die Forscher tief in die physikalische Trickkiste greifen: »Die Grenze zwischen Galliumarsenid und Silizium reflektiert nicht genügend Licht. Wir haben daher einen Extra-Spiegel eingebaut – eine 200 Nanometer dünne Siliziumoxid-Schicht, die auf das Silizium aufgedampft wird«, erklärt Benedikt Mayer, Doktorand im Team von Koblmüller und Finley. »In die Spiegelschicht lassen sich dann feine Löcher ätzen, und in denen kann man mittels Epitaxie Atom für Atom, Schicht für Schicht Halbleiter-Nanodrähte züchten.«

Erst wenn die Drähte über die Spiegelfläche herausragen, dürfen sie in die Breite wachsen – so lange, bis der Halbleiter dick genug ist, dass Photonen in ihm hin und her flitzen und die Aussendung weiter Lichtteilchen anregen können.

Derzeit produzieren die neuen Galliumarsenid-Nanodraht-Laser infrarotes Licht mit einer fest vorgegebenen Wellenlänge und unter gepulster Anregung. »In Zukunft wollen wir die Emissionswellenlänge sowie weitere Laserparameter gezielt verändern, um die Lichtausbreitung unter kontinuierlicher Anregung im Silizium-Chip und die Temperaturstabilität noch besser steuern zu können«, ergänzt Finley.

www.wsi.tum.de

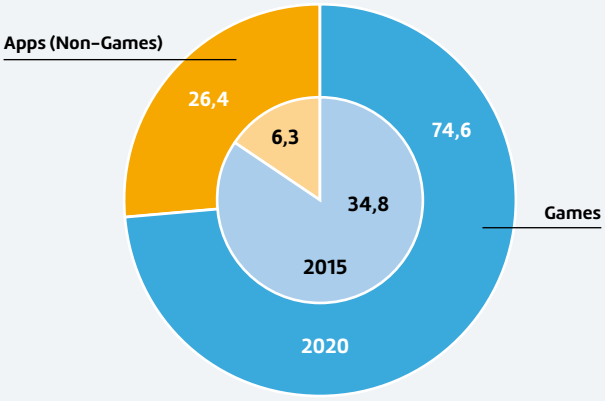
Marktprognose

Spiele-Apps sind die Umsatzbringer

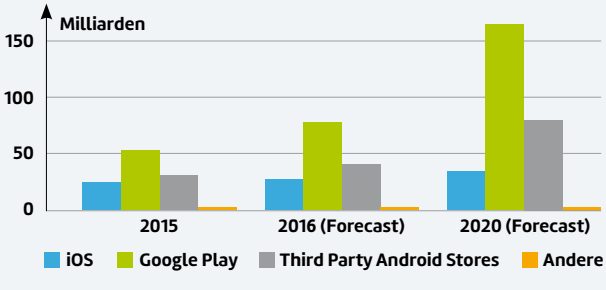
Danielle Levitas: Forschungsleiterin bei App Annie

Die Datenanalyseplattform App Annie hat eine Prognose zur zukünftigen Entwicklung des App-Markts veröffentlicht. »Unsere App-Marktprognose soll der dynamischen und wachsenden App-Wirtschaft wichtige Einblicke und Taktiken bereitstellen und den Unternehmen einen detaillierten Fahrplan bieten«, sagte Danielle Levitas, Senior Vice President of Research bei App Annie. Aufgrund preiswerter Smartphones in jungen Märkten wie Indien werden Milliarden neuer Smartphone-Besitzer in das App-Ökosystem drängen. Dies bedeutet große Chancen für Entwickler und App-Publisher, die den Fokus auf unerfüllte Bedürfnisse in diesen Märkten legen und so ganz neue Märkte schaffen können. Der Spiele-Umsatz wächst laut App Annie weiter. Und: Kategorien wie Musikstreaming, Videostreaming und Dating-Apps werden zu bedeutenden Umsatzmotoren.

Mobile App Forecast – Jahresumsätze
2015 vs. 2020 nach Kategorien, in Milliarden US-Dollar



Mobile App Forecast – Jährliche Downloads
Nach Store, in Milliarden



Quelle: Appannie.com

Uni Liechtenstein

Wie die Maus die Laune verrät

Einem internationalen Forscherteam, dem auch Dr. Markus Weinmann von der Universität Liechtenstein angehört, ist es gelungen, anhand von Bewegungen der Computermaus die emotionale Verfassung des Benutzers zu erkennen.

Emotionen bestimmen das tägliche Miteinander.

Computermaus signalisiert die emotionale Verfassung des Benutzers

Was sich im persönlichen Gespräch in Mimik und Gestik offenbart, zeigt sich auch im Internet. Fünf Forscher aus Liechtenstein, USA, Hongkong und Deutschland fanden heraus, dass sich die Gefühlslage eines Internetnutzers an der Art und Geschwindigkeit seiner Mausbewegungen ablesen lässt.

Dr. Markus Weinmann erklärt, wie das funktioniert: »Ein entspannter Computernutzer bewegt die Maus rasch in geraden Linien oder leicht gekrümmten Kurven. Je frustrierter oder negativer gestimmt er ist, umso langsamer, aber auch eckiger und länger fallen die Mausbewegungen aus«.

www.uni.li

Software Security Report von Veracode

Statische versus dynamische Analysen

Einer der Gründe für das gesteigerte Interesse an Anwendungssicherheit ist der sichtbare Anstieg von Hackerangriffen. Schwachstellen in Webanwendungen und Anwendungscodes zählen dabei zu den größten Einfalltoren für Cyberkriminelle.

Es gibt mittlerweile eine Vielzahl von Testmethoden auf dem Markt. Ganz oben auf der Liste stehen statische und dynamische Analyseverfahren. Diese sollten in keiner Test-Infrastruktur fehlen. Doch was ist das und welche Vorteile bieten sie? Arved Graf von Stackelberg, Director Central Europe bei Veracode, gibt darauf folgende Antworten:

Die Binary-SAST-Technologie – auch bekannt als White-Box-Testing – analysiert Anwendungen auf Sicherheitsschwachstellen im Code – ohne Zugang zum Quellcode zu benötigen. Die statische Binär-Analyse ist die Grundlage für die Einführung einer sicheren Entwicklung und bildet oft den ersten Schritt in der Test-Infrastruktur. Denn zum einem liefert sie das zurzeit umfassendste Ergebnis für Sicherheitstests von Anwendungen. Zum anderen gibt sie der Entwicklung alle notwendigen Informationen an die Hand, um die gefundenen Probleme leicht zu beheben.

Dynamic Application Security Testing (DAST) – Black-Box-Testing– identifiziert Schwachstellen in der Architektur und in Webanwendungen, bevor Cyberkriminelle diese entdecken. Der Tester verfährt dabei wie die Angreifer selbst, wenn diese auf die Suche nach möglichen Angriffszielen gehen, und führt mit der dynamischen Analyse automatisiert Angriffe auf kritische Webanwendungen durch.

Arved Graf von Stackelberg, Director Central Europe bei Veracode

Die Schwachstellen, die nur von DAST gefunden werden, liegen außerhalb des Anwendungscode-Kontextes, haben aber einen umfassenden Einfluss auf die Betriebslaufzeit der Anwendung.

Der Erfolg von dynamischen Analyseverfahren beruht darauf, dass sie die Angriffsfläche einer Anwendung absuchen und analysieren. Das erfordert allerdings Zeit, um eine hohe Code-Abdeckung zu gewährleisten. Leider ist oft zu beobachten, dass Unternehmen beispielsweise aus Kostengründen die Zeitfenster für eine solche Analyse kürzen.

Firmeneigene Workflows, die nicht automatisiert sind, stellen DAST ebenfalls vor eine Reihe von Herausforderungen. Daher kann es sein, dass die Häufigkeit der von DAST entdeckten Schwachstellen die wirkliche Situation nicht korrekt abbilden kann.

Auch statische Analysen stoßen bei nichtautomatisierten, firmeneigenen Abläufen an ihre Grenzen. Denn ohne Ressourcen wie Datenbanken und Dateisysteme sind solche Analysen nicht wirklich verwertbar. Statische Analysen geben vor allem Auskunft über die Codequalität einer Anwendung. Codefehler sind meist sehr einfach zu beheben, weshalb statische Analysen eine höhere Fix-Rate (28 Prozent) aufweisen als dynamische Analysen.

Nicht jedes der beiden Verfahren eignet sich für bestimmte Schwachstellen. Beide Bewertungstechniken unterscheiden sich grundlegend und weisen dadurch unterschiedliche Prävalenzen – Schwachstellenhäufigkeiten – auf.

Den aktuellen State of Software Security Report von Veracode gibt es unter https://info.veracode.com/state-of-software-security-report-volume6-pt2.html

Schwachstellenhäufigkeiten

Schwachstellen-Kategorie	Dynamische Analyse	Statische Analyse
Code-Qualität	nicht geeignet	63%
Kryptografische Schwachstellen	53%	58%
Informationslecks	80%	56%
CRLF Injection	n/a	49%
Entwicklungskonfiguration	55%	nicht geeignet
Serverkonfiguration	16%	nicht geeignet
Cross-Site Scripting (XSS)	27%	47%
SQL Injection	6%	29%
Identifikationsverwaltung	12%	25%
Code Injection	1%	2%
Time and State	nicht geeignet	23%
Directory Traversal	2%	47%
Unzureichende Eingabeprüfung	4%	37%

Proofpoint

Sicherheitsrisiko durch illegale Apps

Die IT-Security-Experten von Proofpoint fanden auf mobilen Geräten von Angestellten Apps, die es im Apple App Store nicht gibt. Sie stammen aus einem illegalen Store, der eine Million kostenpflichtige Apps kostenlos zur Verfügung stellt. Dabei wird eine Methode genutzt, mit der Apps über einen Unternehmens-Store per Sideloading installiert werden.

Apps, die es im Apple App Store nicht gibt

DarkSideLoader verwendet betrügerische oder gestohlene Signierungszertifikate von Unternehmen zur Schaffung riesiger illegaler App Stores. Die Apps sind nicht von Apple geprüft und können Sicherheitsrisiken einführen, selbst auf iOS-Geräten ohne Jailbreak, was bisher nicht möglich war.

Beim Scannen der Apps, die auf den Mitarbeitergeräten bei einem Enterprise-Kunden installiert sind, fanden IT-Sicherheitsexperten von Proofpoint eine Lego-Star-Wars-App, die es im App Store von Apple nicht gibt. Bei weiteren Untersuchungen stellte sich heraus, dass diese aus einem illegalen App Store mit einer Million unerlaubten Apps stammt. Darunter sind zahlreiche kostenpflichtige Apps, die hier kostenlos zur Verfügung gestellt wurden.

Die Anbieter missbrauchen eine Methode, mit der Unternehmen Apps über einen Unternehmens-App-Store per Sideloading auf Mitarbeitergeräten installieren können, die das Unternehmen selbst entwickelt hat. Diese Gruppe ermächtigt sich illegal der legitimen Zertifikate, die für diese Sideload-Apps genutzt werden, und erstellt damit einen ganzen App Store mit Raubkopien. Die geklonten Apps darin können auf den Geräten, auf denen sie installiert werden, ernsthafte Sicherheitsrisiken schaffen, da sie nicht die Sicherheitskontrollen von Apple bestehen mussten. Verbraucher sollten die Empfehlung von Apple beherzigen und nur Apps aus dem Apple App Store installieren.

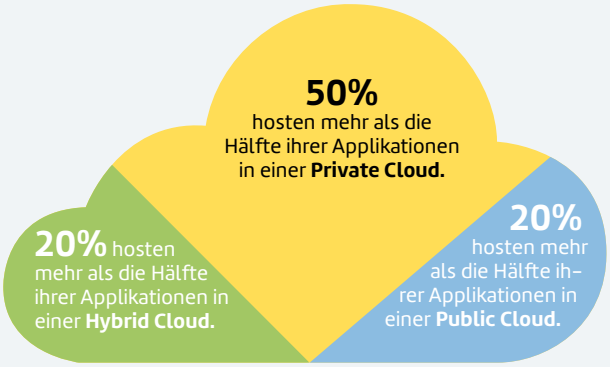
www.proofpoint.com/de

Umfrage

Unternehmen bevorzugen private Clouds

Private Clouds sind nach wie vor das bevorzugte Medium, um geschäftskritische Anwendungen zu hosten, das bestätigte eine Umfrage von ManageEngine.

Private Clouds sind für Unternehmen weiterhin das bevorzugte Medium, um geschäftskritische Anwendungen zu hosten: Laut einer Umfrage von ManageEngine unter mehr als 100 IT-Administratoren in Nordamerika und Europa nutzt die Hälfte



Unternehmen ziehen private Clouds vor

der Unternehmen (50 Prozent) private Clouds, um den Großteil ihrer geschäftlichen Anwendungen bereitzustellen.

Die Nutzung von öffentlichen und Hybrid-Cloud-Angeboten ist jedoch auf dem Vormarsch: Jedes fünfte Unternehmen (20 Prozent) nutzt eine Public Cloud für die Bereitstellung der Mehrheit ihrer Anwendungen. Weitere 20 Prozent der Umfrageteilnehmer verwenden dazu eine Hybrid-Cloud-Lösung und überlegen, weitere Business-Anwendungen dorthin auszulagern.

Der mobile Zugriff auf Business-Anwendungen nimmt zu: Laut 70 Prozent der befragten Unternehmen nutzen die meisten Endanwender Geschäftsanwendungen überwiegend über Webinterfaces; der Zugriff über Mobilgeräte gewinnt allerdings immer mehr an Bedeutung.

www.manageengine.de

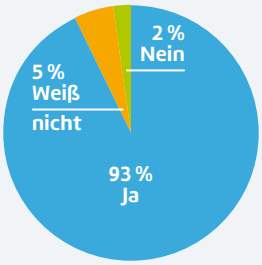
Quelle: ManageEngine.de

DevOps-Strategien

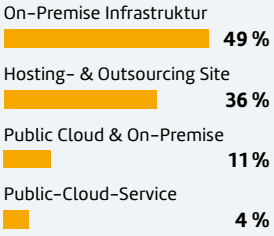
DevOps, Open Source und Business Agility

Open-Source-Lösungen gehören zu den Toptechnologien, die aktuell oder künftig in DevOps-Umgebungen genutzt werden; dazu zählen das Linux-Betriebssystem, eine Infrastructure as a Service wie OpenStack oder eine Platform as a Service wie OpenShift. So lautet ein zentrales Ergebnis des von Red Hat gesponserten IDC-InfoBriefs »DevOps, Open Source and Business Agility«. 93 Prozent der Befragten meinen, dass neue Technologien für den DevOps-Erfolg nötig sind und sie diese in den nächsten beiden Jahren auch nutzen werden. Der IDC InfoBrief ist verfügbar unter: www.redhat.com/en/resources/devops-strategy-open-source-and-business-agility

Sind neue Technologien für den DevOps-Erfolg notwendig?



Welche Technologien sind dafür im Detail erforderlich?



web & mobile developer 4/2016
Quelle: redhat.com

Bitkom

Sprachsteuerung setzt sich bei Smartphones durch

Sprechen statt tippen: Jeder zweite Smartphone-Nutzer (52 Prozent) bedient sein Gerät per Stimme – sei es, um einen Anruf aufzubauen, eine SMS zu diktieren oder nach dem Wetter zu fragen. Das zeigt eine repräsentative Befragung im Auftrag des Digitalverbands Bitkom.

Besonders unter jungen Smartphone-Nutzern ist die Sprachsteuerung beliebt: Bei den 14- bis 29-Jährigen verwenden schon 58 Prozent die Spracheingabe, bei den 30- bis 49-Jährigen sind es 54 Prozent und bei den 50- bis 64-Jährigen 52 Prozent. Aber selbst bei den Smartphone-Nutzern ab 65 gibt jeder Vierte (25 Prozent) Fragen oder Befehle per Stimme ein statt über das Display. »Die Spracherkennungssoftware ist inzwischen so weit ausgereift, dass selbst komplexe Befehle oder Fragen gut verstanden und ausgeführt beziehungsweise beantwortet werden«, sagt Bitkom-Hauptgeschäftsführer Dr. Bernhard Rohleder. »Die Sprachsteuerung macht die Bedienung des Smartphones noch intuitiver und komfortabler. Auch für ältere oder körperlich eingeschränkte Menschen kann dies eine enorme Erleichterung sein«, so Rohleder.

Wer die Sprachsteuerung seines Smartphones nutzt, tut das derzeit vor allem, um einen Anruf aufzubauen (80 Prozent), aber auch um eine Textnachricht zu verfassen (49 Prozent) oder um beispielsweise das Wetter oder die aktuellen Fußballergebnisse abzufragen (33 Prozent). Ebenfalls beliebt sind die Navigation (16 Prozent) oder das Starten von Apps (12 Prozent) per Spracheingabe.
www.bitkom.org

Gründung des Industrial Data Space e.V. in Berlin

Fraunhofer, Wirtschaft, ZVEI

Industrial Data Space e.V. gegründet

Die Fraunhofer-Gesellschaft, 16 Wirtschaftsunternehmen und der ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V. haben einen gemeinnützigen Verein zum Industrial Data Space gegründet.

Aufgabe des Industrial Data Space e.V. ist es, Wissenschaft und Wirtschaft für nachhaltige Lösungen zu vernetzen, die Architektur des Industrial Data Space mitzugestalten sowie zentrales Organ für die Kooperation mit verwandten Initiativen zu sein. Präsident der Fraunhofer-Gesellschaft Prof. Dr. Reimund Neugebauer sagte anlässlich der Gründung: »Der Industrial Data Space ermöglicht einen sicheren Datenaustausch mit gemeinschaftlichen Regeln für alle Unternehmen – auf Basis eines offenen Architekturmodells. Fraunhofer verfügt über ausgewiesene Exzellenz und Kompetenzen in den dafür notwendigen Technologiefeldern. Darum ist das Interesse der Wirtschaft immens: Als Kandidaten für Pilot-Use-Cases liegen uns bereits rund 70 Vorschläge vor. Jetzt gilt es für uns, diejenigen Projekte auszuwählen und voranzutreiben, mit denen wir die Digitalisierung der Industrie erfolgreich gestalten.«

Der gemeinnützige Verein soll die Anforderungen an den Industrial Data Space bündeln, den Erfahrungsaustausch zwischen Wissenschaft und Wirtschaft organisieren und Leitlinien für die Zertifizierung, Standardisierung und Verwertung der Ergebnisse des Forschungsprojekts »Industrial Data Space« entwickeln.
www.fraunhofer.de

Integration von Wearables

Neues Ford-Forschungslabor

Persönliche Technik-Features und Fahrzeug-Technologien sollen immer enger zusammenwachsen – dies ist das Ziel des neuen Forschungslabors für die Integration von Wearables, das Ford jetzt an seinem US-amerikanischen Hauptsitz in Dearborn/Michigan gegründet hat. Die Wissenschaftler und Ingenieure arbeiten dort an der Integration von tragbaren Gadgets und Fahrzeugen. Sie beschäftigen sich unter anderem mit der Frage, wie wichtige Gesundheitsinformationen des Fahrers mit Fahrzeug-Technologien wie beispielsweise dem Fahrspurhalte-Assistent verknüpft werden können. Ziel ist es, die Sicherheit der Autofahrer weiter zu verbessern.
www.ford.de

Das neue Forschungslabor für die Integration von Wearables von Ford

eco Sicherheitsreport 2016

Das Bewusstsein wächst mit den Risiken

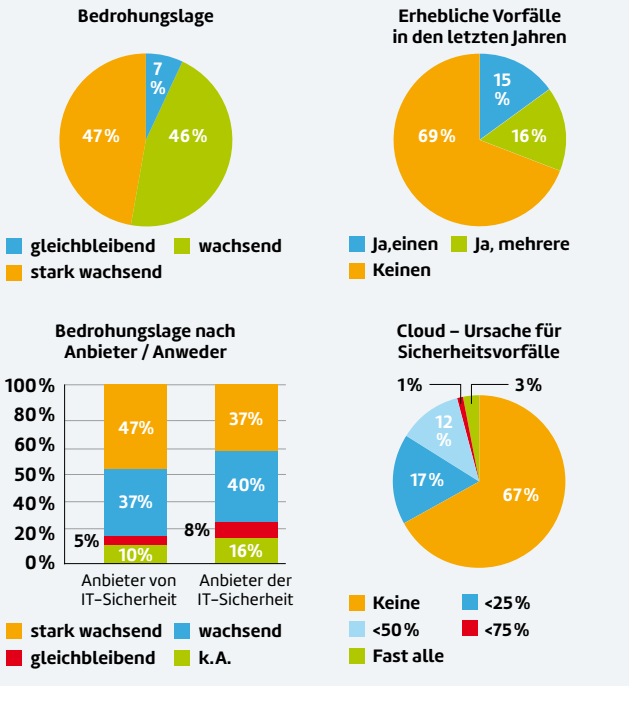
Die Bedrohungslage bei der IT-Sicherheit verschärft sich – diese Einschätzung vertritt ein Großteil der deutschen Wirtschaft im aktuellen eco Sicherheitsreport 2016. An der zugrunde liegenden Umfrage von eco – Verband der Internetwirtschaft e.V. haben 580 Unternehmen teilgenommen. Davon empfinden 47 Prozent die Bedrohungslage als »stark wachsend« und weitere 46 Prozent als »wachsend«. Sieben Prozent gehen von einer gleichbleibenden Bedrohung aus. Einen Rückgang hat kein einziges der von eco befragten Unternehmen ausgemacht.
Ein knappes Drittel (31 Prozent) der Firmen hatte »erhebliche Sicherheitsvorfälle« in den letzten Jahren, 16 Prozent waren sogar mehrfach betroffen. Laut eigenen Angaben hatten 69 Prozent der deutschen Wirtschaft in der jüngsten Vergangenheit mit keinen nennenswerten Sicherheitsproblemen zu kämpfen.

Dabei beurteilen die Anbieter von IT-Sicherheitslösungen die Bedrohung deutlich schärfer als die Anwender, hat der eco Report ergeben. 47 Prozent der Anbieter sprechen von einer stark wachsenden Bedrohung, aber lediglich 37 Prozent der Anwender sehen dies genauso. Dahinter mag kommerzielles Interesse stehen, wahrscheinlicher aber die bessere Übersicht über die tatsächliche Lage, weil Anbieter naturgemäß besonders oft mit Sicherheitsvorfällen konfrontiert werden.

Über die Hälfte (51 Prozent) der von eco befragten Firmen gehen für das Jahr 2016 von steigenden Ausgaben für IT-Sicherheit aus. Weitere 13 Prozent erwarten sogar einen starken Anstieg. Beinahe ein Drittel (32 Prozent) will 2016 genauso viele Mittel aufwenden wie im Vorjahr. Ganz ähnlich sieht die Entwicklung beim Outsourcing der IT-Sicherheit aus. 53 Prozent wollen ihren Outsourcing-Aufwand 2016 erhöhen, 34 Prozent auf dem Vorjahresniveau verharren.

Bedrohungslage

Die Bedrohungslage bei der IT-Sicherheit verschärft sich



Quelle: Verband der Internetwirtschfat e.V.

Telerik DevCraft

Erste UI für AngularJS 2

Progress hat eine neue Version der Telerik DevCraft Suite vorgestellt, einer .NET-Toolbox für die Entwicklung von Web-, mobilen und Desktop-Anwendungen. Mit dem Release unterstützt Progress AngularJS 2

Von der Telerik DevCraft Suite gibt eine neue Version

Alpha und ist damit der erste Anbieter eines UI für dieses JavaScript-Framework.

In enger Zusammenarbeit mit Google unterstützt Telerik als eines der ersten Unternehmen überhaupt die aktuellen Neuerungen von AngularJS. Das neue Release von Telerik Kendo UI, einem Bestandteil der Telerik DevCraft Suite, unterstützt jetzt sowohl AngularJS 1.x als auch die AngularJS 2 Preview. Künftig wird sie auch die AngularJS 2 Beta unterstützen, die im zweiten Quartal 2016 erscheint.

Das neue Release der Telerik DevCraft Suite erweitert diese Funktionen um verbesserte Responsive-Web-Verhaltensweisen und neue Responsive-Demos. Auch das mobile Rendering und die Gesten-Unterstützung wurden umfänglich erweitert. Darüber hinaus bieten die Telerik Web UI Tools der Telerik DevCraft Suite zahlreiche Verbesserungen. Sie erlauben Entwicklern, hochresponsive und Touch-freundliche Web-Apps für die Plattform ihrer Wahl schnell und einfach zu erstellen. Dazu zählen neue App-Templates für VisualStudio und verbesserte Smart Tags im UI für ASP.NET Ajax.

»Wir überarbeiten und erweitern unsere erfolgreichsten Produkte kontinuierlich, um die Anforderungen des Marktes nach Support für neue Webplattformen zu erfüllen. Wir sind stolz darauf, unter den ersten Unternehmen zu sein, die AngularJS 2 unterstützen«, sagt Marina Hristova, Vice President Product Marketing & Management Telerik DevTools bei Progress. »Unser DevCraft-Angebot geht weit über AngularJS 2 hinaus und beinhaltet die volle Breite an HTML5/JS- und sämtlichen .NET-Plattformen. Unsere Tools beschleunigen mit intuitiven APIs die Anwendungsentwicklung und machen Telerik DevCraft zu einer umfassenden Entwicklungslösung.«
www.telerik.com

Oxid-BeezUP-Modul

Vertriebskanäle

Mit dem BeezUP-Artikel-Export-Modul für Oxid-Shops lassen sich Produkte noch effizienter auf den unterschiedlichsten Vertriebskanälen im Internet anbieten. Online-Händler sparen damit deutlich mehr Zeit, wenn sie ihre Artikel auf Preisvergleichsportalen oder Marktplätzen listen.

BeezUP: Ein Artikel-Export-Modul für Oxid-Shops

Léon Hüls, Geschäftsführer der Response GmbH, erklärt: »Mit der neuen Schnittstelle unseres Oxid-BeezUP-Moduls können unsere Kunden ihren gesamten Artikelstamm noch schneller zuordnen und exportieren.«
www.response-gmbh.de

DWX Developer Week 2016

**20.-23. Juni 2016,
Messe Nürnberg**

Mit Code
DWX16wmd
€ 200,–
sparen!

Das Event 2016 für .NET-, Web- & Mobile-Entwickler

Das AdBoard und Track Chairs der DWX 2016 (u. a.)


Gregor Biswanger


Jan Fellien

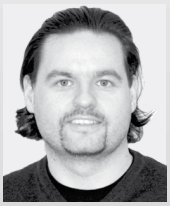

Dr. Ronald Hartwig


Johannes Hoppe


Patrik Lobacher


Hendrik Lösch


André Krämer


Björn Schotte


Ulrike Stirnweiß


David C. Thömmes


David Tielke


Holger Wendel

developer-week.de

DeveloperWeek

Aussteller & Sponsoren: adorsys, AIT, brainLight LIFE IN BALANCE, Fast Reports Reporting must be Fast!, GFU Cyrus AG, MATHEMA, MAXIMAGO, SDX, SPARX SYSTEMS, TEXT CONTROL

Veranstalter: developer media, Neue Mediengesellschaft Ulm mbH

Präsentiert von: dnc dotnetpro, web & mobile DEVELOPER

Foto: xxxxxxxxxx

NOSQL-DATENBANKEN IM ÜBERBLICK

Alternative NoSQL

NoSQL-Datenbanken bieten einen besseren Durchsatz, höhere Verfügbarkeit und einfachere Erweiterbarkeit als ihre relationalen Gegenspieler.

Neue Anwendungstypen aus den Bereichen Online Analytical Processing (OLAP), Data Mining, Big Data und vor allem aus dem mobilen Web/Internet führten in den vergangenen Jahren zu einem enormen Wachstum an Datenbeständen und der Anzahl von Benutzern. Web/Internet- und Mobile-Anwendungen wie Amazon (Webshop), Facebook, Google (Suchmaschine), LinkedIn, Twitter, Instagram, YouTube, Telegram oder WhatsApp verdeutlichen diesen Sachverhalt.

Für Realisierung und effizienten Betrieb derartiger Anwendungen stellten sich gängige Datenbank-Management-Systeme und die mit ihnen verbundenen Datenbanktechniken als ungeeignet heraus. Anwendungstypen mit massivem Datenaufkommen und ständig steigender Benutzeranzahl erfordern vielmehr den Einsatz gänzlich neuer Typen von Datenbanken.

Die Gesamtheit dieser neuen Datenbanktypen bezeichnet man im Unterschied zu den traditionellen, relationalen Datenbanken als NoSQL-Datenbanken. Dabei steht das etwas missverständliche Akronym NoSQL für Not only SQL; es schließt also SQL als Sprache zur Datenmanipulation streng genommen nicht aus (Bild 1).

Vor allem soll die Wortfolge Not only (Nicht nur) ausdrücken, dass relationale Datenbank-Management-Systeme weiterhin als bewährte Werkzeuge für die Datenhaltung anzusehen sind. Relationale Datenbanken haben sich in der Praxis als Standard der Datenhaltung in vielen Anwendungsgebieten durchgesetzt. Bei ihnen handelt es sich um ausgereifte Lösungen, auch was Backup, Recovery, Optimierung oder Tuning betrifft.

Jedoch stellen im Zusammenhang mit diesen neuen Anwendungstypen relationale Modelle nicht immer die ideale Lösung dar. Insbesondere erfüllen rein relationale Systeme nicht die stetig steigenden Anforderungen der enormen Datenmengen, was Performance, Verfügbarkeit, Erweiterbarkeit und Fehlertoleranz betrifft.

Zudem kristallisierte sich bald ein Widerspruch des durch relationale Datenbanken favorisierten Transaktionsmanagements auf Basis des ACID-Prinzips gegenüber den Anforderungen heraus. Das Einhalten des ACID-Prinzips bildet in vielen Fällen eine Hürde, die relationale Datenbanken in den Szenarien Big Data oder Data Warehousing durch Sonderlösungen umschiffen müssen.

Leider gibt es für NoSQL zum heutigen Stand noch keine standardisierte Definition. Die Abkürzung soll primär ausdrücken, dass es sich um nicht rein relationale Ansätze handelt. Trotz der seit fast zehn Jahren andauernden NoSQL-Strömung existieren bis jetzt weder Normierungsgremien noch andere Organisationen, die eine Begriffsklärung tatsächlich forcieren. Während eine relationale Datenbank vorwiegend einfach strukturierte Massendaten (numerisch, Textform) verarbeitet, wollen NoSQL-Datenbanken den mit Einzug der Web-2.0-Welle verbundenen exponentiellen Wachstumsraten Herr werden. Im Unterschied zu den relationalen bieten NoSQL-Datenbanken vor allem eine bessere Performance und Verfügbarkeit, um die mit dem Internet verbundene digitale Informationsflut zu bewältigen.

NoSQL als Sammelbegriff für nicht (rein) relational

NoSQL-Datenbanken brechen mit dem Status quo der relationalen Datenhaltung, also den durch ein Datenbankschema vordefinierten, zeilenorientierten Tabellen. Diese Tabellen nennt man gelegentlich auch Relationen, da sie über ihre Spaltenwerte Beziehungen zwischen den Anwendungsdaten abbilden.

Die NoSQL-Bewegung postuliert eine Abkehr vom relationalen Datenmodell, um die besonderen Anforderungen, was

Das ACID-Prinzip einer relationalen Datenbank

Die beiden deutschen Informatiker Theo Härder und Andreas Reuter prägten das Akronym ACID zur Charakterisierung von Ausführungseinheiten, sogenannten Transaktionen.

ACID beschreibt die von allen Verarbeitungsschritten einer relationalen Datenbank zu erfüllenden Eigenschaften:
A steht für Atomacy (Atomarität/Abgeschlossenheit): Eine Sequenz von Datenoperationen (Transaktion genannt) wird entweder ganz oder gar nicht ausgeführt.
C steht für Consistency (Konsistenzerhaltung): Vor und nach der Ausführung einer Transaktion befinden sich die Daten in einem konsistenten Zustand.
I steht für Isolation (Abgrenzung): Transaktionen können parallel, das heißt unabhängig voneinander, ausgeführt werden; sie beeinflussen sich nicht gegenseitig.
D steht für Durability (Dauerhaftigkeit): Datenoperationen werden entweder nach vollständiger Ausführung mittels einer Commit/Abschluss-Operation dauerhaft gespeichert oder bei unvollständiger Ausführung durch Abort/Rollback wieder rückgängig gemacht.

»No« im Akronym NoSQL sollte als Not only (Nicht nur) gelesen werden, da SQL-ähnliche Sprachen zur Manipulation oder Abfrage für eine NoSQL-Datenbank möglich sind (Bild 1)

Datenvolumen, Benutzeranzahl und Modellierung betrifft, überhaupt erfüllen zu können. Leider bezieht sich der Begriff NoSQL nicht auf die mit einer relationalen Datenbank zwingend verbundenen Abfragesprache SQL. Anstelle von NoSQL wäre, wie öfters gewünscht wird, die Bezeichnung NoRel sicherlich die treffendere und damit bessere gewesen.

Einer NoSQL-Datenbank liegt primär ein nichtrelationales, genauer gesagt ein nicht auf Tabellen basierendes Datenmodell zugrunde. Anstatt aus Tabellen mit Zeilen oder Spalten, die aus einfach strukturierten Daten (Zahlen, Text) bestehen, setzen sich die Elemente einer NoSQL-Datenbank aus weniger oder gar nicht strukturierten Konstrukten zusammen. Bei ihnen spielen Zahlen und Text als Konstrukte für den Aufbau von Daten an sich eine untergeordnete Rolle, da relationale Systeme diese erschöpfend behandeln. Ein Datenbanksystem heißt NoSQL-Datenbanksystem, wenn es eines oder mehrere der nachfolgenden Kriterien oder Eigenschaften erfüllt, das heißt, diese als Charakterzüge besitzt:

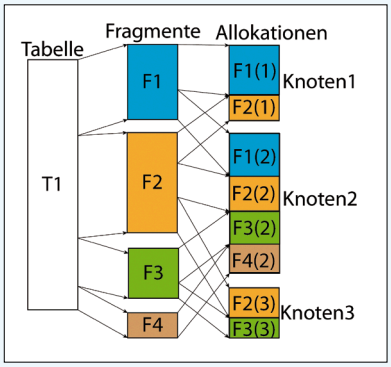
- **Open Source:** Ein NoSQL-System stellt in der Regel über eine Community-Edition ein Open-Source-System dar und ist als solches frei, also kostenlos zugänglich. Zudem vermarktet der Hersteller häufig auch eine kommerzielle Version des Datenbank-Management-Systems.
- **Schemafreiheit:** Die Datenbank besitzt im Unterschied zu einer relationalen Datenbank schwächere Schema-Restriktionen oder ist häufig auch völlig schemafrei. Damit soll sich das Speichermodell möglichst flexibel und natürlich (mit wenig Aufwand bei der Modellierung) gestalten lassen. Dies erhöht zweifelsfrei die Produktivität in der Programmierung, birgt aber gleichzeitig die Gefahr von Fehlern (bei mangelnder Abstimmung der Datenänderungen) in anderen, bereits bestehenden Programmen.
- **Skalierbarkeit:** Eine NoSQL-Datenbank implementiert eine Verteilung durch horizontale Fragmentierung; erst diese ermöglicht und vereinfacht die Skalierung. In der Regel realisiert man diese Partitionierung mittels Sharding: Auf den vernetzten Rechnern erfolgt die Verwaltung der Datenpartitionen jeweils über eine eigene Serverinstanz; man nennt diese Rechner auch Database-Shards.
- **Datenreplikation:** Aufgrund seiner verteilten Architektur unterstützt das NoSQL-Datenbanksystem meistens auch eine einfache Replikation der Datenbestände. ▶

Skalierung durch Fragmentierung/Partitionierung

Fragmentierung in der Datenbanktechnik zerlegt eine Datenbank in mehrere Teile und speichert diese separat ab. Diese Technik zielt darauf ab, in Anwendungen mit sehr großen Datenbeständen zurechtzukommen.

Generell unterscheidet man als die zwei wichtigsten Arten die vertikale und die horizontale Fragmentierung: Erstere zerlegt einzelne Datensätze in Einzelteile und speichert diese verteilt ab. Im Unterschied dazu speichert horizontale Fragmentierung (auch Partitionierung genannt) logisch zusammengehörende Datensätze als Einheit getrennt; zum Beispiel alle Kunden mit dem Anfangsbuchstaben A bis K getrennt von den Kunden mit Anfangsbuchstaben L bis Z.

Verwalten unterschiedliche Serverinstanzen eines Datenbank-Management-Systems die einzelnen Teile (Shards) einer horizontalen Partitionierung/Fragmentierung, so spricht man von Sharding.



Der Datenbankadministrator bestimmt, auf welchen Knoten/Rechnern die Datenbank einzelne Fragmente einer Tabelle/Relation speichert

- **Verzicht auf Joins:** Eine NoSQL-Datenbank kennt keine Joins. Derartige Operationen für die Verknüpfung von Tabellen (Verbund genannt) führt eine SQL-Datenbank typischerweise bei komplexeren Abfragen durch. Der NoSQL-Speicher strukturiert Daten vielmehr so, dass der Zugriff auf sie direkt und schnell erfolgen kann. Daher nennt man NoSQL-Datenbanken seit ihrer Entstehung manchmal auch strukturierte Datenspeicher (Structured Storage).
- **Konsistenzmodell:** Das Datenbanksystem kennt keine Transaktionen gemäß dem gängigen ACID-Prinzip und verzichtet bewusst auf dieses; meistens liegt ein anderes Konsistenzmodell zugrunde, in Kurzform BASE genannt.
- **Einfache Abfragesprache:** Die NoSQL-Datenbank bietet keinen Zugriff mittels reinem SQL. Sie besitzt in der Regel aber auch für Endbenutzer eine einfache Abfragesprache, die eine SQL-ähnliche Syntax aufweisen kann. Dem Programmierer gegenüber präsentiert sich die Abfragesprache zusätzlich als Application Programming Interface (API).

Inzwischen haben auch Hersteller relationaler Datenbanken wie IBM, Microsoft, Oracle, Software AG oder Teradata die Bedeutung der NoSQL-Bewegung erkannt und ihre Produkte in diese Richtung weiterentwickelt. Heutzutage bieten einige SQL-Datenbanken wie DB2, MySQL, Oracle, SQLServer oder Teradata Features an, die man normalerweise nur bei einer NoSQL-Datenbank vermuten würde. In der Regel realisierten die Datenbank-Hersteller ihre NoSQL-Features in Pilotprojekten bei strategisch wichtigen Kunden oder kauften einen kleinen Hersteller einer NoSQL-Datenbank auf.

Anders als NoSQL-Datenbanken besitzen ausschließlich relationale Systeme noch nicht die Fähigkeit, für ein Anwendungsgebiet mit hohem Aufkommen an Datenbeständen und großer Benutzeranzahl geeignet kostengünstig zu skalieren. Zudem führt eine rein relationale Datenbank Änderungen von Daten immer innerhalb einer Transaktion gemäß dem ACID-Prinzip durch. Neuerdings bezeichnet man NoSQL-Datenbanken auch als Web-Scale-Datenbanken, wobei diese beiden Attribute Web und Scale zwei wesentliche NoSQL-Eigenschaften klarer in den Vordergrund rücken.

CAP-Theorem als treibende Kraft des NoSQL-Konsistenzmodells

NoSQL-Datenbanken gehen effizient mit vielen Schreib-Lese-Operationen bei gleichzeitig hohem Lastaufkommen von Webseiten oder Streaming-Media-Anwendungen um. Sie vermeiden das bei relationalen Datenbank bekannte Leistungsproblem datenintensiver Applikationen. Diese Tatsache begründet sich auch durch das mit einer NoSQL-Datenbank verbundene spezifische Konsistenzmodell.

Im Unterschied zur relationalen Datenbank bietet es hinsichtlich der Korrektheit und Vollständigkeit der Daten nur schwache Garantien; beispielsweise Eventual Consistency oder auf einzelne Datensätze eingeschränkte (ACID)-Transaktionen.

In der Regel basieren NoSQL-Datenbanken auf horizontal verteilten Datenbanken mit redundanter Datenhaltung über viele Server. Diese Vorgehensweise begründet die mit NoSQL-Systemen gegenüber ihren relationalen Vertretern verbundene einfachere und kostengünstigere Skalierbarkeit. Grundsätzlich besitzen Anfragen an eine NoSQL-Datenbank nicht nur akzeptable Antwortzeiten, sondern werden auch stets beantwortet. Ausfälle einzelner Server führen nie zu einem Ausfall des Gesamtsystems, da das Datenbank-Management-System auch bei Verlust einzelner Server weiterarbeiten kann.

Synchronisation von Transaktionen

Anwendungen im Online-Betrieb führen Transaktionen auf einer Datenbank in der Regel nicht seriell, sondern parallel durch.

Um das ACID-Prinzip für alle Transaktionen zu erfüllen, müssen diese synchronisiert ausgeführt werden. Die Synchronisation bewirkt, dass Änderungen aller Transaktionen an den Daten so vorgenommen werden, als ob diese seriell hintereinander entsprechend ihrer Reihenfolge erfolgt wären. Die Informatik erforscht Verfahren, um Serialisierung zu gewährleisten, im Teilgebiet der Concurrency Control.

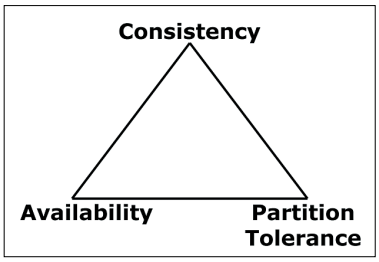
NoSQL-Systeme bieten also eine hohe Verfügbarkeit (Availability) und Partitionstoleranz (Partition Tolerance). Die beiden zuvor genannten Eigenschaften untersuchte der Informatiker Eric Brewer bei Google im Rahmen großer verteilter Systeme.

Dabei stieß Brewer bereits vor mehr als 15 Jahren auf ein Grundprinzip des verteilten Computing. Dieses Prinzip nennt man CAP-Theorem, dessen Gültigkeit einige Jahre später von der Theoretischen Informatik bewiesen wurde. Als Akronym steht CAP für die Begriffe Consistency (Konsistenz), Availability (Verfügbarkeit) und Partition Tolerance (Partitionstoleranz):

- **Consistency (C):** Soll ausdrücken, dass alle Clients jederzeit die gleichen Daten sehen. Bei Abschluss einer Transaktion in einem verteilten System muss sichergestellt sein, dass danach auch alle Replikate des manipulierten Datensatzes aktualisiert werden.
- **Availability (A):** Alle Clients können stets Lese- und Schreibzugriffe durchführen. Die Verfügbarkeit bietet stets akzeptable Antwortzeiten.
- **Partition Tolerance (P):** Auch bei Ausfall einzelner Server arbeitet das System als Ganzes weiter. Verteilte Systeme benötigen eine hohe Ausfalltoleranz.

Brewer erkannte für diese drei genannten Eigenschaften eines verteilten Systems, dass immer nur zwei davon gleichzeitig garantiert werden können, jedoch nicht alle drei. Diesen Zusammenhang veranschaulicht ein Dreieck (Bild 2), bei dem ein konkretes verteiltes System sich immer einer der Kanten/Schenkel (CA, CP oder AP) zuordnen lässt.

Die Systemeigenschaften C, A und P stellen graduelle Größen dar. Ist die Verfügbarkeit A hoch, so antwortet das System schnell, ist sie gering, dann antwortet das System langsam. Für die Konsistenz C bedeutet dies, dass sie entweder sofort sichergestellt ist (wie etwa beim ACID-Prinzip relationaler Datenbanksystem) oder erst nach einem gewissen Zeitfenster der Inkonsistenz.

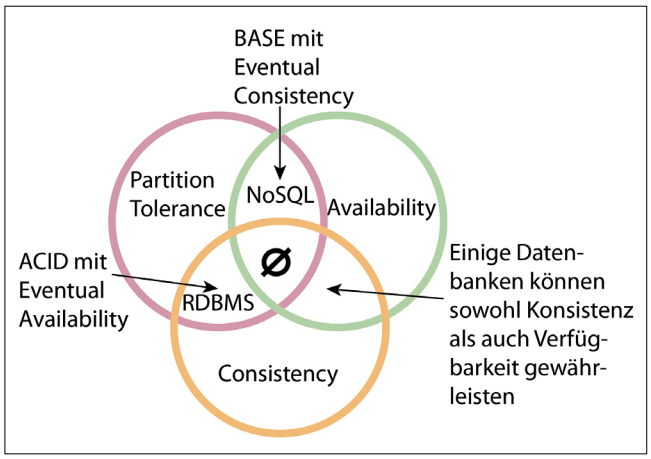


Das Dreieck visualisiert die Zusammenhänge zwischen den Systemeigenschaften des CAP-Theorems (Bild 2)

Das BASE-Prinzip als geeignete Ergänzung des ACID-Prinzips

Typischerweise wollen alle klassischen relationalen Datenbanken die Systemeigenschaft C der Konsistenz zu jedem beliebigen Zeitpunkt garantieren. Ein streng relationales Datenbanksystem fällt in die Kategorie CA (Bild 3): Verfügbarkeit und Konsistenz sind insbesondere bei einem Einzelsystem sehr hoch – es erfüllt das ACID-Prinzip. Bei einer verteilten relationalen Datenbank sinkt die Verfügbarkeit der Datensätze, da die Ausführung einer Transaktion aufgrund des zwingend zu erfüllenden ACID-Prinzips mehr Zeit beansprucht. Je größer die Anzahl der Knoten/Rechner, also der Verteilungsgrad ist, desto länger dauert eine Transaktion. Im Zeitalter des Web 2.0 sollen die vielen verteilten Daten jedoch schnell zu jedem Zeitpunkt für jeden Benutzer der Anwendung verfügbar sein.

Verschiedene Datenoperationen können in einer relationalen Datenbank zu einer Ausführungseinheitseinheit zusammengefasst werden, in der Datenbanktechnik Transaktion genannt. Jede Transaktion muss das ACID-Prinzip erfüllen; dazu stellt das Datenbanksystem entsprechende Methoden zur Verfügung, um den Anfang, das Ende, den Abbruch oder die vollständige Rücknahme aller bereits durchgeführter Datenoperationen kennzeichnen zu können. Eine Transaktion startet mit ihrem Beginn, vor deren Ende erfolgt eine Abfrage, ob alle Operationen korrekt durchgeführt werden konnten. Im Ja-Fall führt das Datenbanksystem eine sogenannte Commit-Operation durch, macht alle Datenänderungen persistent und beendet anschließend die Transaktion.



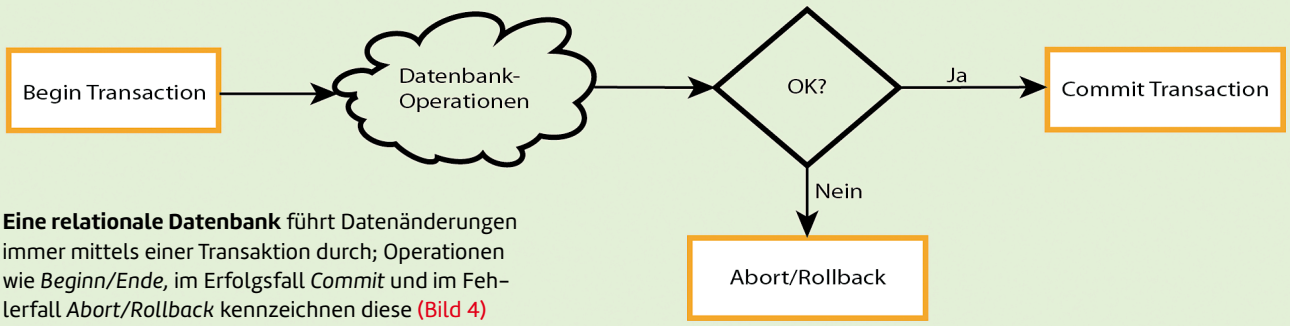
Relationale Datenbank-Management-Systeme (RDBMS) und NoSQL-Datenbanken nehmen gemäß ihren Eigenschaften für sie typische Bereiche im CAP-Theorem ein. Die leere Schnittmenge der drei Kreise verdeutlicht die Basis-Aussage des CAP-Theorems (Bild 3)

Abbruch der Transaktion

Blieb eine der Operationen erfolglos, konnte sie also nicht vollständig durchgeführt werden, so bricht das Datenbank-Management-System die Transaktion ab (Abort) und macht alle durch sie vorgenommenen Änderungen in den Datenbeständen mittels einer Rollback-Operation wieder rückgängig. Transaktionen lassen sich problemlos ineinanderschachteln oder auch parallel ausführen, da jede für sich genommen das ACID-Prinzip erfüllt. In der Abbildung (Bild 4) soll dies die Bearbeitungswolke *Datenbank-Operationen* veranschaulichen.

Bei einer Verschachtelung nutzt das Datenbank-Management-System zur Feststellung oder Auflösung von Konflikten unterschiedliche Synchronisationsverfahren.

Relationale Datenbanken greifen für die Umsetzung der Synchronisation auf Sperrverfahren, auch Locking oder Sperrprotokolle genannt, zurück. Sperrverfahren fallen in die Kategorie der pessimistischen Synchronisationsverfahren; sie gehen davon aus, dass es mit hoher Wahrscheinlichkeit zu ▶



Eine relationale Datenbank führt Datenänderungen immer mittels einer Transaktion durch; Operationen wie *Beginn/Ende*, im Erfolgsfall *Commit* und im Fehlerfall *Abort/Rollback* kennzeichnen diese (Bild 4)

einem Konflikt kommt. Im Gegensatz dazu gehen optimistische Synchronisationsverfahren davon aus, dass kein Konflikt bei der Ausführung paralleler Transaktionen entsteht. Quasi als Ergänzung zum ACID-Prinzip relationaler Datenbanken verwenden NoSQL-Datenbanken als Konsistenzmodell das BASE-Prinzip. Als grundlegende Idee verbirgt sich dahinter, das die Konsistenz der Daten als ein Zustand betrachtet wird, der irgendwann sicher erreicht wird. BASE drückt einzeln für sich gesehen aus:

- **Basically available:** Das Datenbanksystem ist generell verfügbar; diese Verfügbarkeit wird aber nicht garantiert. Zu jedem Zeitpunkt kann eine Version der Daten abgerufen oder gelesen werden, diese sind aber möglicherweise veraltet.
- **Soft state:** Der Zustand des Datenbanksystems kann sich im Lauf der Zeit selbst ohne Dateneingabe ändern. Das heißt, die Daten der Datenbank sind nicht beständig. Werden sie nicht aktualisiert, können sie verworfen werden und sind anschließend nicht mehr verfügbar.
- **Eventual consistency:** Bedeutet letztendliche Konsistenz – die Datenbank befindet sich nach Ablauf einer gewissen (möglichst kurzen) Zeitspanne der Inkonsistenz (temporäre Inkonsistenz wird erlaubt) wieder in einem konsistenten Zustand, sodass danach alle Clients die gleichen Daten sehen.

Da inkonsistente Zustände für kurze Zeitabstände erlaubt sind, könnte man meinen, es handle sich bei BASE eher um ein Gegenstück zu ACID. Führt man sich aber vor Augen, dass der Begriff oder die Ausprägung des Konsistenzbegriffs vom konkreten Anwendungsfall und der wertenden Sicht des Endbenutzers abhängt, so merkt man schnell, dass es sich bei BASE eher um eine Ergänzung oder Erweiterung (im Sinne einer Lockerung) des ACID-Prinzips relationaler Datenbanken handelt.

So erlauben einige Anwendungen, benutzerseitig eine Aktualisierung der Daten anzustoßen; danach erhält man als Benutzer eine konsistente Sicht. Werden letztendlich alle Aktivitäten an der Datenbank beendet und diese in den Ruhezustand versetzt, garantiert das Datenbank-Management-System immer die vollständige Konsistenz aller Daten.

Klassifizierung für Vorauswahl in der Praxis

Die große Anzahl über 200 am Markt aktuell verfügbarer NoSQL-Systeme verlangt eine geeignete Kategorisierung, um einen besseren Überblick für ihren Einsatz in Anwendungen zu bekommen. Eine Klassifizierung nach dem Erfüllungsgrad der NoSQL-Kriterien und -Eigenschaften ergibt wenig Sinn, da dieser sich aufgrund aktueller Weiterentwicklungen der NoSQL-Systeme ständig ändert.

In der Regel entstanden NoSQL-Datenbank-Management-Systeme im universitären Umfeld oder Forschungsabteilungen von Großunternehmen. Die treibende Kraft für Investitionen in die Entwicklung eines NoSQL-Systems bildeten stets konkrete Anwendungen, die eine hohe Priorität für die Bereitstellung in Produktion oder für die strategische Ausrichtung des Unternehmens besaßen.

Das speziell dabei entwickelte Datenmodell dieser unternehmenskritischen Anwendungen und die damit verbundenen Konstrukte zum Aufbau ihrer Speicherstrukturen (die Storage-Engine) eignet sich besser als Grundlage für eine

Abgrenzung zu Dokumentationssystemen

Eine dokumentorientierte Datenbank darf nicht mit einem Dokumentationssystem verwechselt werden.

Ein Document Store fällt in die Kategorie der Systemsoftware, während es sich bei einem Dokumentationssystem um Anwendungssoftware handelt. Ein Dokumentationssystem verwaltet und archiviert Dokumente; es stellt Verfahren bereit, um Dokumente für das Archiv zu erschließen und dort wieder auffinden zu können.

Das Archiv basiert in der Regel auf einer Datenbank mit Dokumenten, ihren Metadaten und einem Ordnungssystem für Ablage und Suche/Retrieval. Beide Prozesse finden mittels Katalogen über Stich- und Schlagwörter (Deskriptoren) statt. Während Stichwörter direkt dem Dokument entstammen, in ihm also vorkommen, findet man Schlagwörter nicht im Dokument selbst, vielmehr gehören sie zu einem kontrollierten Vokabular, auch Thesaurus genannt.

Ein Thesaurus beschreibt, welche Schlagwörter für bestimmte Sachverhalte verwendet werden. Die Zuordnung von Schlagwörtern (Verschlagwortung) für ein Dokument zur Ablage im Archiv nennt man Indexierung; der umgekehrte Vorgang der Suche nach Dokumenten über Stich- und Schlagwörtern heißt Retrieval.

Durchsatz maximieren und Latenz minimieren

Als wichtige Messgrößen für den im NoSQL-Umfeld typisch anzutreffenden hohen Leistungsanspruch (Performance) gelten Durchsatz und Latenzzeit.

Durchsatz (Throughput): Menge der pro Zeiteinheit verarbeiteten Daten; auch Datendurchsatzrate oder Durchsatzrate genannt.

Der Durchsatz darf nicht mit der bei relationalen Datenbanken bekannten Messgröße **Queries Per Second (QPS)** verwechselt werden. Bei QPS (Anfragen pro Sekunde) handelt es sich um die Anzahl der pro Sekunde vom Datenbanksystem erhaltenen Anfragen.

Latenz (Latency): Zeitdauer (Zeitintervall) bis zum Eintreten der gewünschten Antwort seitens der Datenbank; dabei handelt es sich um eine als Störung empfundene Verzögerung, daher manchmal auch Verzögerungszeit genannt.

Architektur eines Datenbanksystems

Ein Datenbanksystem besteht aus einer Menge aufeinander abgestimmter Komponenten an Systemsoftware. In der Regel stellt man die Architektur eines Datenbanksystems in Form verschiedener Schichten oder Hierarchien mit unterschiedlichen Schnittstellen dar.

Daher spricht man auch vom Schichtenmodell eines Datenbanksystems. Je nach zugrunde liegender Ausprägung der Architektur gibt es gänzlich verschiedene Schichtenmodelle. Als Bezeichnung für den Namen legt man häufig die Anzahl verwendeter Schichten zugrunde. Als bekanntester Vertreter gilt das 3-Ebenen-Modell, auf dem das ANSI-SPARC-Modell und auch das CODASYL-Datenbankmodell basiert:

Im Mittelpunkt des 3-Ebenen-Modells steht das konzeptionelle/logische Schema, das den logischen Aufbau der Datenbank vollständig inklusive aller Integritätsbedingungen beschreibt.

Alle Anwendungen/Programme und Endbenutzer greifen über ein individuelles externes Schema auf die Datenbank zu; es stellt eine Teilmenge des logischen Schemas dar.

Das interne Schema beschreibt die Ablagestrukturen, das heißt, wie eine Datenbank die Objekte des logischen Schemas physisch speichert.

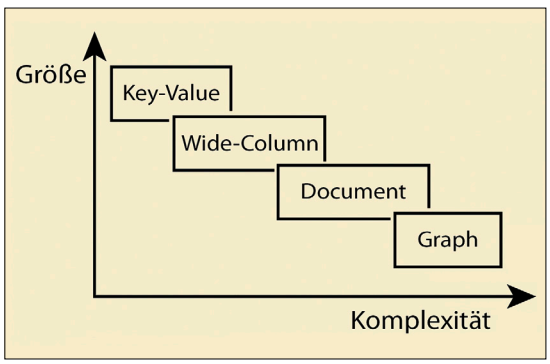
mögliche Aufteilung der NoSQL-Systeme in verschiedene Klassen, zumal das Datenmodell und die mit ihm verbundenen Speicherstrukturen als eine stabile Größe des NoSQL-Datenbank anzusehen sind.

Bei einer genaueren Analyse von Datenmodell, Speicherstrukturen und Anwendungen stellt man auf einer höheren Ebene sogar Gemeinsamkeiten fest. Fasst man die Ähnlichkeiten der einzelnen NoSQL-Datenbanken zu Gruppen zusammen, so kristallisiert sich eine Einteilung der NoSQL-Systeme in die nachfolgenden Hauptklassen als gröbste Klassifizierung heraus.

Entsprechend der Reihenfolge in der folgenden Liste steigt die Komplexität der Speicherstrukturen an, während die Größe/Anzahl an Elementen in der Datenbank abnimmt (Bild 5):

- **Schlüssel-Wert-orientierte Datenbank:** auch Key-Value-Datenbanken/Cache oder Key-Value/Tuple Store genannt.
- **Spaltenorientierte Datenbank:** auch (Wide-)Column Store/Column-Families oder Extensible Record Stores genannt.
- **Dokument-orientierte Datenbank:** auch Document Store genannt.
- **Graph-Datenbanken:** auch Graph Database Systems oder Graph-orientierte Datenbank genannt.

Vor allen Dingen aufgrund der vielfältigen Implementierungen eines NoSQL-Datenbank-Management-Systems könnte man diese Hauptklassen zusätzlich noch in Untergruppen unterteilen: Beispielsweise die Key-Value-Datenbanken nach verschiedenen Speicherformen, Graph-Datenbanken nach zugrunde liegendem Datenmodell oder der primär unterstützten Programmiersprache. Auch lassen sich die Hauptklassen nach Ausprägung oder Erfüllung des CAP-Theorems gruppieren, das heißt, einer der Kanten beziehungsweise Schenkel des CAP-Dreiecks zuordnen (Bild 6). Als weitere für die Praxis sinnvolle Bewertung der Hauptklassen bietet sich der Erfüllungsgrad von wichtigen Systemeigenschaften wie Performance, Skalierbarkeit, Flexibilität, Komplexität und Funktionalität an. Für einen Einstieg in die NoSQL-Technologie kristallisierten sich jedoch die obigen Hauptklassen als eine bewährte Vorgehensweise für die Praxis heraus. Mit ihnen lässt sich eine geeignete Vorauswahl für das Anwendungsgebiet wie folgt treffen: Anhand von ausgewählten Praxisfällen und den damit verbundenen Geschäftsprozessen analysiert man das Datenmodell und führt einen Machbarkeitsnachweis (Proof of Concept) mit mehreren NoSQL-Vertretern durch. Diese Evaluierung liefert für jeden Vertreter einer Hauptklasse den jeweiligen Grad seiner Eignung für die Anwendungsdomäne.



Mit zunehmender Größe der Datenbank sinkt ihre Komplexität und umgekehrt: Mit steigender Komplexität sinkt ihre Größe. Oder anders ausgedrückt: Mit zunehmender Komplexität einer Datenbank sinkt die Möglichkeit, die Daten effizient zu verteilen (Bild 5)

Ergänzend kommen noch weitere zu berücksichtigende Kriterien wie Datenschutz, Datensicherheit, Know-how im Unternehmen, Performance, Beschaffungskosten, System-Infrastruktur, Investitionssicherheit, IT-Strategie/Architektur und andere hinzu. ▶

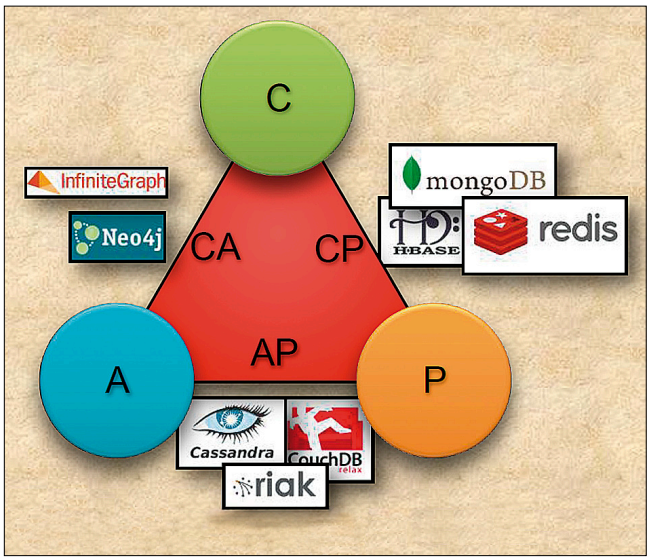
Key-Value-Datenbanken (Key-Value Stores) stellen auf sehr hohe Verarbeitungsgeschwindigkeit ausgerichtete, äußert leichtgewichtige Systeme dar. Wie ihr Name bereits ausdrückt, bilden sie Schlüssel auf Wertepaare ab. Der Schlüssel besteht im Unterschied zu einer relationalen Datenbank nicht nur aus einer einfachen Zeichenkette; manchmal handelt es sich auch um einen aus mehreren Teilen zusammengesetzten Schlüssel (Composite Key): einer primären und einer sekundären Komponente; beide zusammen bilden den Key Path. Über ihn führt die Datenbank den direkten Zugriff auf das gewünschte Objekt aus. Der Value/Wert nimmt beliebige Zeichenketten ein, ganz einfache oder auch sehr komplexe Datenstrukturen.

Sortierte Reihenfolge

Dieses NoSQL-System speichert den Schlüssel gemeinsam mit seinem Wert in einer assoziativen Datenstruktur ab; gegebenenfalls auch in sortierter Reihenfolge (Bild 7). Die Datenstruktur entspricht den aus Programmiersprachen bekannten Konstrukten: Dictionary, Map, Hash, Hash Map oder Hash Table. Ein Schlüssel-Wert-Paar nennt man zusammen manchmal auch Tupel; innerhalb eines Programms greift man über den Schlüssel darauf zu. Der Schlüssel identifiziert das Objekt/Tupel in der Datenbank eindeutig. Für derartige Zugriffe gibt es bei vielen Key-Value Stores weder Mengenoperationen noch sekundäre Zugriffspfade; nur den direkten Zugriff über diesen Key oder Key Path. Im Wesentlichen beschränken sich die Anfrage- und Änderungsfunktionalitäten auf die drei Operationen: Lesen/Get/Read, Einfügen/Put/Insert und Löschen/Delete.

Eine Key-Value-Datenbank in ihrer Reinform verfügt über kein Schema und keine zusätzlichen Meta-Informationen. Das Datenbank-System interpretiert nicht die gespeicherten Daten, diese Aufgabe muss die Anwendung selbst übernehmen. Zudem kennt das Datenbank-System auch keine Transaktionen im Sinne des ACID-Prinzips. Operationen (Abfrage, Einfügen, Ändern, Löschen) auf den Key-Value-Paaren führt die Datenbank immer einzeln aus. Datenbankseitig lassen sich nur einfache Anfragen durchführen, Joins muss der Anwendungscode übernehmen. Dank dieser Realisierung besitzen Key-Value Stores eine sehr hohe Performance und eine sehr gute Skalierbarkeit. Wegen des Verzichts auf das ACID-Prinzip führt dieser Typ von NoSQL-Datenbank alle Operationen ohne zusätzlichen Overhead sofort aus.

Eine Key-Value-Datenbank erzielt für Lese- und Schreib-Operationen sehr hohe Durchsatzraten bei gleichzeitig niedriger Latenz. Generell lassen sich Schlüssel-Wert-Paare recht einfach und mit nur minimalem administrativem Aufwand auf mehrere Knoten verteilen. Arbeiten zum Lastausgleich (Load Balancing) wie Hinzufügen und Entfernen von Knoten übernimmt entweder ein allgemeiner Load Balancer oder sogar das Datenbanksystem selbst, führt diese also automatisch durch. Außerdem verfügen Key-Value Stores über integrierte Mechanismen, um Fehler zu erkennen und zu beheben. Insgesamt garantiert die verteilte Datenbank eine sehr hohe Verfügbar-



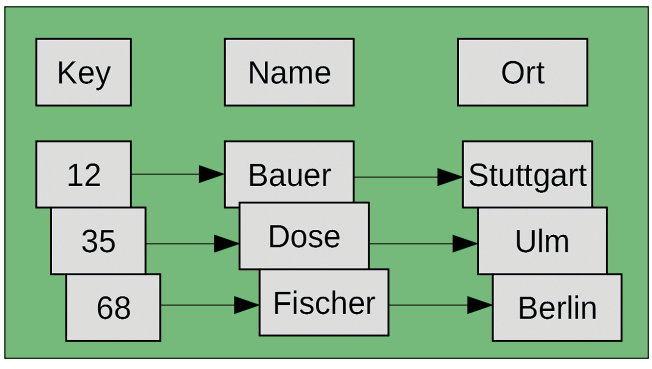
Zuordnung einiger ausgewählter NoSQL-Datenbank-Management-Systeme anhand ihrer Eigenschaften auf das CAP-Dreieck (Bild 6)

Multi-Model-Datenbanksysteme – ein NoSQL-Trend

Eine Multi-Model-Datenbank bietet Entwicklern gleichzeitig mehrere Datenmodelle zur Auswahl an.

Dabei scheint sich als wesentlicher Vorteil für Datenbank-Administratoren das einheitliche Backend zu erweisen. Für Entwickler bietet es, wie bei der Auswahl einer Programmiersprache, die für ihre Anwendungen erforderliche Wahlfreiheit, verbunden mit höherer Produktivität.

Intern besitzt eine Multi-Model-Datenbank verschiedene Storage-Engines, verbunden mit Skalierbarkeit und Fehlertoleranz; zudem verfügt sie über mehrere Konzepte des Transaktionsmanagements. Als aktuelle Forschungsgegenstände für Multi-Model-Datenbanken können Verbindungen mit den in der relationalen Welt bestehenden Anwendungen und die Wahrung der Datenkonsistenz über verschiedene Storage-Engines hinweg angesehen werden



Spalten in einer Tabelle können beliebig viele unterschiedliche Werte einnehmen; Schlüssel machen diese schnell zugänglich (Bild 8)

Einsatz im Data Warehousing/Big Data

Auch für die Analyse von Massendaten (Big Data), wie sie beim Data Warehousing oder OLAP (Online Analytical Processing) auftreten, eignen sich Graph-Datenbanken.

OLAP darf nicht mit OLTP (On-line Transaction Processing) verwechselt werden; bei OLTP handelt es sich um kritische Geschäftsanwendungen, die Transaktionen im ACID-Sinne auf einer Datenbank in Echtzeit ausführen.

Data Warehousing findet im Rechnerverbund (Computer-Cluster/Server-Farm) durch High-Performance-Computing statt. Als Ergänzung zum Graph-Datenbanksystem kommt für die in dieser Cluster-Umgebung anfallenden Aufgaben in der Regel noch eine Big-Data-Plattform wie Spark oder Hadoop zum Einsatz.

Integritätsbedingungen und Datenkonsistenz

Bedingungen, die eine Datenbank sicherstellen muss, um einen konsistenten Zustand für die Daten anzunehmen, nennt man Integritätsbedingungen: In diesem Fall gewährleistet die Datenbank Datenkonsistenz.

In der Regel beschreiben Integritätsbedingungen Annahmen, die seitens aller Objekte in der Datenbank nach Abschluss sämtlicher Programme zu erfüllen sind. Beim Einfügen, Ändern oder Löschen eines Elements in der Datenbank muss die Einhaltung der Integritätsbedingungen entweder automatisch oder programmseitig geprüft werden. Durch Programmierfehler entstehen häufig inkonsistente Zustände der Datenbank, welche den Integritätsbedingungen widersprechen.

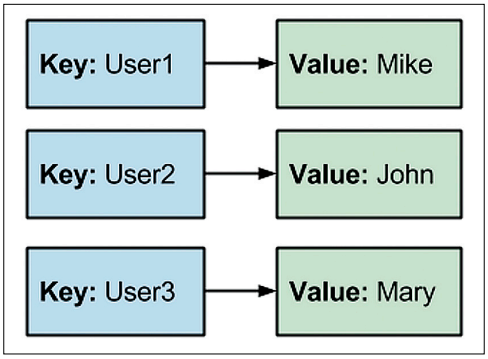
keit. Bekannte Vertreter eines Key-Value Store stellen DynamoDB (Amazon), Memcached, Riak, Redis und Voldemort dar; auch Object-Storage-Services als Dienste zum Speichern von Objekten in der Cloud wie Azure Storage von Microsoft, Amazon S3 oder Google Cloud Storage zählen dazu.

Wide Column Stores

Auf konzeptueller Ebene visualisiert man die Daten eines Wide Column Store, einer spaltenorientierten Datenbank, mittels Tabellen, die aus Zeilen und Spalten bestehen. In der Begriffswelt ihrer Bausteine ähneln Wide Column Stores daher relationalen Datenbanken, da auch sie Datensätze mittels Tabellen über Zeilen und Spalten verarbeiten. Allerdings unterscheidet sich ihr Datenmodell grundlegend vom relationalen Ansatz.

Ein Wide Column Store identifiziert eine Zeile wie ein relationales System über einen Schlüssel (Row Key genannt); ihre Spalten verfügen aber in der Regel über eine beliebige Anzahl von Einträgen; während bei einer relationalen Datenbank eine Zeile immer die gleiche Anzahl von Spalten mit identischem Datentyp besitzt. Ein Datensatz in einem Wide Column Store besitzt auf Spaltenebene eine beliebige Anzahl von Schlüssel-Wert-Paaren (Bild 8), daher betrachten manche Informatiker diese NoSQL-Klasse auch als Sonderform eines Key-Value Store.

Technisch gesehen bauen die internen Datenstrukturen eines Wide Column Store auf den mit einer Data-Engine gewonnenen Erfahrungen eines Key-Value Store auf. Die Namen und Formate der Spalten eines Wide Column Store variieren von Zeile zu Zeile und nehmen daher eine dynamische Größe ein. Einige Vertreter dieser NoSQL-Klasse legen die Daten physikalisch spaltenorientiert ab. Manche Vertreter besitzen sogar eine SQL-ähnliche Abfragesprache. Ein Tupel (Datensatz) einer relationalen Datenbank entspricht in einem Wide Column Store einer Row; während ein Attribut ein Schlüssel-Wert-Paar darstellt. Die Werte eines Attributs werden als mehrdimensionales assoziatives Array gespeichert; dieses kann sämtliche primitive Datentypen, aber auch komplexe Objekte und sogar Bilder oder Videos aufnehmen.



Beispiel einer Key-Value-Datenbank mit drei Datensätzen, nach Benutzernummer sortiert (Bild 7)

Google gilt mit der Bigtable-Datenbank als Begründer dieser Klassen von NoSQL-Datenbanken. Ein speziell für die Datenablage entwickeltes Konstrukt, Bigtable genannt, kommt bei der Google-Suche für die Verwaltung des Google-Index, aber auch in Google Analytics, Google Earth oder Google Maps zum Einsatz. Neben einem breiten Einsatzgebiet erfüllt Bigtable vor allem die Anforderung einer linearen horizontalen Skalierbarkeit mit hoher Performance und permanenter Verfügbarkeit.

Bemerkenswerterweise reagiert Bigtable recht flexibel auf widersprüchliche Parameter wie latenzsensitive Anfragen, hohen Durchsatz oder fehlerresistente Persistenz. Die NoSQL-Systeme HBase, Accumulo oder Hypertable stellen eine Implementierung von Googles Bigtable dar, während Cassandra, der bekannteste Vertreter dieser Klasse, eine Mischung aus Amazon Dynamo und Bigtable einsetzt.

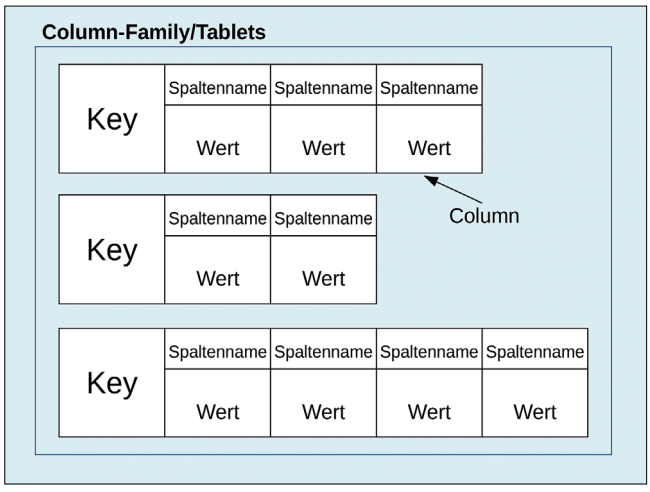
Ein Wide Column Store verfügt über ergänzende Konstrukte, um ähnliche oder zusammengehörende Attribute/Zeilen/Spalten zu einer Einheit zusammenzufassen. Das einfachste unter diesen Konstrukten stellt die Bildung einer Menge von Zeilen zu einer sogenannten Column Family/Tablets dar (Bild 9); dieses Strukturelement kennt man in der relationalen Welt nicht. Im Unterschied zur relationalen Datenbank sind Wide Column Stores zudem schemafrei; sie besitzen deshalb eine sehr große Flexibilität beim Umgang mit heterogenen Daten. Wegen des fehlenden Datenbank-Schemas kennen Column Families auch keine Beziehungen zwischen den Da- ▶

tensätzen. Solche Beziehungen müssen immer seitens der Anwendung gepflegt werden, womit Wide Column Stores auch keine Joins unterstützen.

Bei einem Document Store handelt es sich um eine Sammlung von echten Dokumenten oder zumindest semistrukturierten Daten (keine einheitliche, sondern verschachtelte Struktur, unterschiedliche Werttypen, mehrwertige Spalten); vergleichbar einer Key-Value-Datenbank mit einem Dokument als Value. Diese Sammlung oder Zusammenfassung von Dokumenten nennt man auch Collections beziehungsweise Views. Document Stores besitzen wie relationale Datenbanken ein ausdrucksstarkes Datenmodell; sie verwalten aber keine zusätzlichen Strukturinformationen über die gespeicherten Daten – sind also schemafrei. Allerdings enthalten Dokumente neben den Attributwerten auch Metadaten.

Auf konzeptioneller Ebene modelliert man einen Document Store als verschachtelte Tabelle mit erweiterbaren Attributen. Intern präsentieren sich die einzelnen Datensätze meist in Form von JSON- (JavaScript Object Notation) oder BSON-Dokumenten (Binary JSON) (Listing 1). Die auf der internen Ebene verwendete JSON-Notation verarbeitet die Anwendung direkt weiter, da man JSON-Zeichenfolgen sehr einfach in JavaScript-Arrays/Objekte umwandeln kann. Aufgrund der Metadaten verfügt ein Document Store auch über einen großen Umfang an Operationen und Operatoren. Diese Klasse einer NoSQL-Datenbank kennt bis auf die *Join*-Operation alle Anfrageoperationen einer relationalen Datenbank. Dabei führt die Datenbank jede Operation einzeln auf einem Dokument aus. Zur Verarbeitung von Anfragen greift der Document Store auf sekundäre Zugriffspfade zurück.

Ursprünglich unterstützen Document Stores keine Transaktionen – bildet man zusammengehörige Daten über Aggregationen in einem Dokument ab, so lassen sich diese jedoch atomar ändern. Wegen der fehlenden Strukturinformationen implementiert ein Document Store keine Integritätsbedingungen. Allerdings bieten Document Stores für konkurrierende Schreiboperationen eine individuelle Synchronisierung an. Da dieser NoSQL-Datenbanktyp Dokumente nicht fragmentiert speichert und auch keine Joins und Transaktionen unterstützt, lassen sich Document Stores leicht auf mehrere Rechner verteilen. Jedes Dokument wird atomar auf einen Server verteilt; entsprechende Anfragen direkt an diesen weitergeleitet. Bekannte Vertreter dieser NoSQL-Klasse sind MongoDB, Couchbase oder CouchDB.



Beispiel einer Column-Family mit drei Datensätzen (Bild 9)

Listing 1: JSON-Dokument für Kundeninformationen

```
{
  "vorName": "Otto",
  "nachName": "Bauer",
  "adresse":
  {
    "strasse": "Hirschgasse",
    "stadt": "Aidlingen",
    "postleitzahl": "71134"
  },
}
```

Graph-Datenbanken

Graph-Datenbanken eignen sich besonders gut zur Modellierung ausgewählte Netzstrukturen oder ganzer Netzwerke, da ihre interne Ebene stark vernetzte Daten unmittelbar performant verarbeitet. In der Regel handelt es sich bei diesen Datenstrukturen um Netzwerke im Sinne der Graphentheorie: Ein Graph (Bild 10) stellt eine Menge von Knoten (manchmal auch Ecken genannt) und eine Menge von Kanten (Pfeile, Verbindungen) dar. Eine Kante verbindet zwei Knoten miteinander; diese Verbindung bildet eine Beziehung (Relationship) zwischen diesen zwei Knoten ab. Sowohl Knoten als auch Kanten (Beziehungen) eines Graphen können Eigenschaften (Attribute) aufweisen.

Verwendet man Pfeile, so drücken sie aus, dass es sich um gerichtete Beziehungen handelt. Generell lässt sich jedes Datenmodell in einen Graphen transferieren; notfalls bildet man dazu mehrwertige Beziehungen zwischen unterschiedlichen Objektklassen durch verschiedene einwertige Beziehungen ab. Graph-Datenbanken verfügen generell über kein Schema, können aber rekursive Strukturen abbilden.

Bekannte Vertreter einer Graph-Datenbank sind: Giraph, HyperGraphDB, InfiniteGraph, InfoGrid, Neo4j oder Titan. Die wesentliche Besonderheit eines Graph-orientierten Datenbanksystems liegt jedoch in seiner Optimierung für stark vernetzte Datenstrukturen.

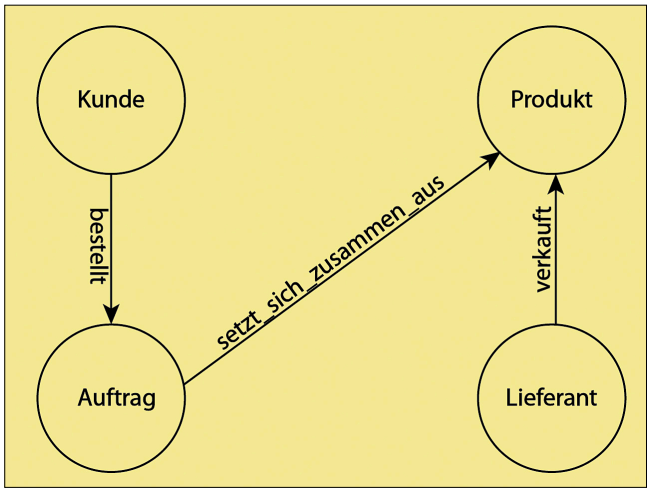
Zum anderen bieten Graph-Datenbanken spezialisierte Graph-Algorithmen, um komplizierte Abfragen und Analysen ihrer Datenbestände vorzunehmen. Dazu gehören einfache Algorithmen, um alle direkten und indirekten Nachbarn eines Knotens zu finden (traversieren) oder kürzeste Pfade zwischen zwei Knoten zu berechnen.

Zum Teil handelt es um mächtige Algorithmen für eine Suche im gesamten Graphen, der Analyse ausgewählter Abschnitte oder einzelner Elemente eines Netzwerks: Zusammenhangskomponenten, minimalspannende Bäume oder das Problem des Handlungsreisenden (Traveling Salesman Prob-

lem). Einige dieser Algorithmen bilden die Grundlage für allgemein wiederverwendbare Graph-Algorithmen; zu diesen zählen die Breitensuche (Breadth-First Search) oder die Tiefensuche (Depth-First Search).

Die bekanntesten Graph-Algorithmen stammen aus Gebieten der Operation-Research wie Graphentheorie oder Netzplantechnik; auch in der Informatik trifft man auf sie: im Compilerbau bei Graphersetzungssystemen, in der Sprachverarbeitung bei der Erkennung von Mustern (Pattern-Matching) oder für die Datenanalyse im OLAP/Big-Data-Bereich.

Aus Gründen der Wiederverwendbarkeit, verbunden mit Wirtschaftlichkeit und hoher Komplexität ausgewählter Graph-Algorithmen, entstanden eigenständige Bibliotheken/Frameworks wie Boost Graph Library (BGL), JGraphT, igraph (The network analysis package), Implicit Graph Search Library, JUNG (Java Universal Network/Graph Framework), TinkerPop oder YAGSBPL (Yet Another Graph-Search Based Planning Library).



Kunden-Auftrag-Lieferanten-Datenbank, modelliert als einfacher Graph (Bild 10)

Multi-Model-Datenbanken als neuer Trend

Bisherige Anwender relationaler Datenbanken erkannten bei der Entwicklung völlig neuer Anwendungen für das Internet, dass sich diese nicht mit der relationalen Datenbanktechnik realisieren lassen. Die mit relationalen Systemen verbundenen Konzepte und deren Implementierung widersprechen den zu erfüllenden Anforderungen. Zunehmend entdeckten diese Anwender, dass sich sogenannte NoSQL-Datenbanken für derartige Einsatzbereiche besser eignen als die von ihnen eingesetzten relationalen Produkte.

Diesen Trend haben auch einige der großen Hersteller relationaler Datenbanksysteme (IBM, Microsoft, Oracle, Software AG, Teradata) erkannt: Sie versuchen, ihr eigenes Produkt durch Anpassungen an die NoSQL-Bewegung flexibler zu machen, oder wollen sich das Know-how für diese neue Technologie sichern. Dazu implementieren diese Hersteller konkrete NoSQL-Systeme oder kaufen kleinere Software-Häuser mit den aus ihrer Sicht geeigneten Produkten auf und beginnen, diese unter ihrer Flagge zu vermarkten. Einige Hersteller wie Apple integrieren die aufgekauften Produkte in von ihnen geplante eigene Anwendungen.

Leider mangelt es bisher bei allen NoSQL-Systemen an einer bei vielen Großanwendern anzutreffenden wichtigen Anforderung: der Standardisierung. Zudem hat sich gezeigt, dass NoSQL-Alternativen selbst bei diesen Großanwendern eher auf wenige Einsatzgebiete beschränkt sind. Auch lassen sich bewährte relationale Datenbanksysteme nicht immer durch eine NoSQL-Datenbank ablösen: Die derzeitige NoSQL-Technologie verfügt noch nicht über die geforderten Konzepte, wie sie für geschäftskritische Anwendungen notwendig sind. Dies erkannten auch die Hersteller einiger NoSQL-Produkte und begannen mit der Umsetzung der aus ihrer oder Kundensicht wichtigen relationalen Techniken. Derzeit erscheinen am Markt sogar Hersteller, die hier bisher noch nicht tätig waren, mit völlig neu entwickelten Produkten.

Diese aktuellen Entwicklungen verdeutlichen die Notwendigkeit der Praxis von Brücken zur Kopplung beider Datenbanktechniken: der NoSQL und der relationalen. Zukünftig wird es daher zu einer Verbindung relationaler und NoSQL-Produkte kommen, um die Vorteile beider nutzen zu können. Schon jetzt bewegen sich beide Welten, die relationale und die der NoSQL-Systeme, mit einer herstellerspezifischen Geschwindigkeit aufeinander zu. Im Unterschied zu den Herstellern relationaler Produkte erkennen immer mehr Hersteller der NoSQL-Produkte die übergeordnete Bedeutung gewonnener Erfahrungen aus der jeweils anderen Welt: Sie beginnen, mit ihrem Datenbanksystem mehrere verschiedene Datenmodelle zu unterstützen. Neben ihrem ursprünglichen Datenmodell verfügen manche inzwischen auch über ein (oder gar mehrere) andere. Als Bezeichnung für diese neuartige Datenbanktechnologie zeichnet sich der Begriff eines Multi-Model-Datenbanksystems ab. ■

Links zum Thema

- Liste von Datenbanksystemen des Non-relational-Universums
  http://nosql-database.org
- Frei zugängliche Liste verfügbarer Datenbanksysteme
  http://db-engines.com/de/systems

Frank Simon
arbeitet in der Software-Entwicklung mit folgenden aktuellen Arbeitsgebieten: Entwicklung, Programmierung, Test und Debugging von Cloud-, Rich-Internet-, Mobile- und Webanwendungen inklusive deren System-Management.
web_mobile_developers@gmx.eu

WEBDESIGN FÜR TOUCHSCREENS

Touch me!

Mit den richtigen Anpassungen werden Webseiten auf Touchscreens besser bedienbar.

Webseiten müssen sich gut an die verschiedenen Bildschirmgrößen anpassen – das ist uns allen dank responsiven Webdesigns präsent. Dabei verliert man allerdings aus dem Blick, dass wir es bei unserer neuen Gerätevielfalt nicht nur mit den verschiedensten Bildschirmgrößen zu tun haben, sondern auch mit unterschiedlichen Eingabearten: Neben den bei Desktop-Rechnern normalerweise benutzten Inputgeräten Maus und Tastatur gibt es die Touchscreens, die mit Fingern bedient werden. Was muss man hierbei beachten?

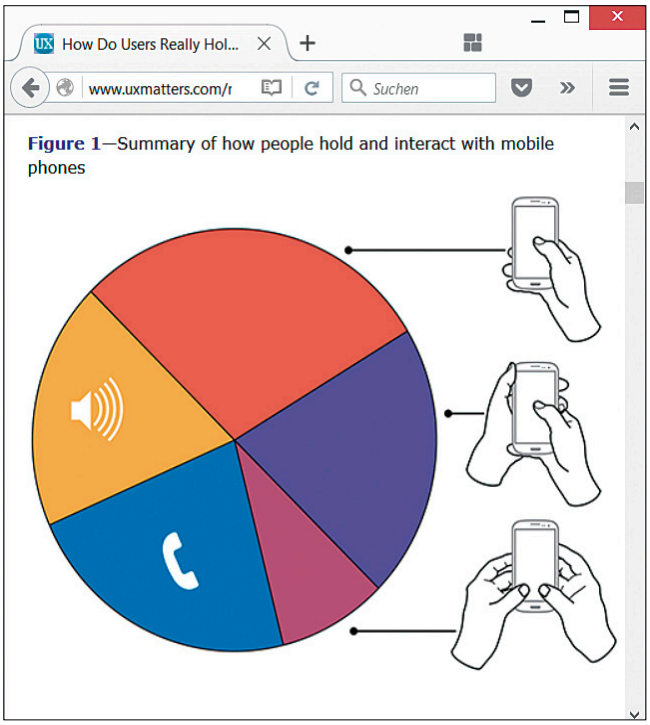
Finger sind anders

Die Bedienung einer Webseite mit der Maus unterscheidet sich ganz wesentlich von der Bedienung per Finger, weil Mauszeiger und Finger wenig Gemeinsamkeiten haben.

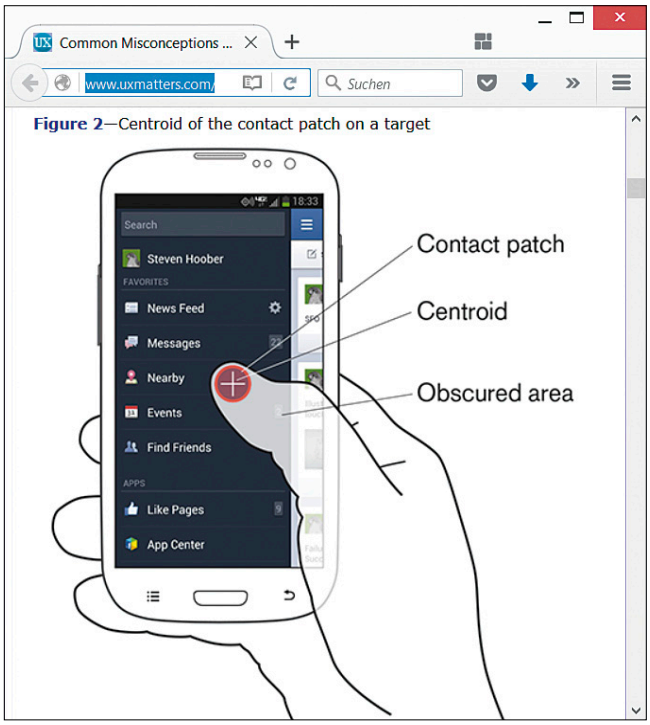
Auffällig ist erst einmal der Größenunterschied: Der Mauszeiger ist sehr klein, die Berührungsfläche eines Fingers ist wesentlich größer. Dies gilt auch und gerade für den Daumen, der in der überwältigenden Anzahl der Fälle zum Einsatz kommt, und nicht etwa der Zeigefinger. Steven Hoober hat untersucht, wie Leute ihre Smartphones halten, und dabei festgestellt, dass Smartphones am häufigsten nur in einer Hand gehalten und mit dem Daumen bedient werden. Etwas weniger verbreitet ist das zweihändige Halten des Geräts, aber auch hier werden üblicherweise Daumen für die Eingabe genutzt – daneben gibt es aber auch Einsätze des Smartphones ohne Fingerberührung des Displays beispielsweise bei Sprachsteuerung oder natürlich beim Telefonieren (Bild 1).

Der Mauszeiger kann sich verändern, je nachdem, wo er sich befindet, und damit dem Benutzer Hinweise auf die gerade mögliche Aktionen geben. Bei Touchscreens fehlt diese Option, Änderungen können nur am Element, das angetippt wird, angezeigt werden; das hat unter anderen den Nachteil, dass das berührte Element durch den Finger selbst teilweise verdeckt wird.

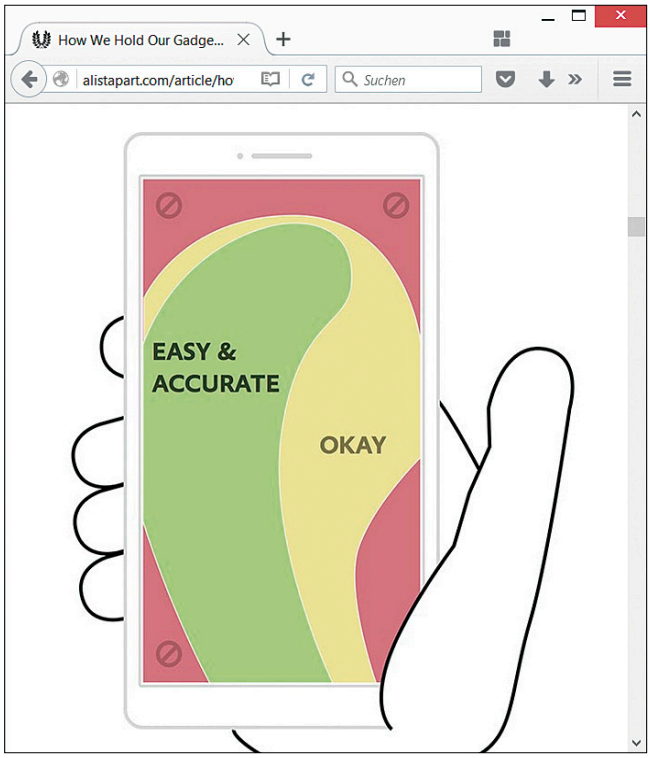
Eine weitere Besonderheit der Eingabe mit Fingern ist die geringere Präzision. Es erfordert keine besonderen Fähigkeiten, mit der Maus einen Punkt auf dem Bildschirm genau zu treffen, mit dem Finger ist es schwieriger oder sogar unmöglich. Wann haben Sie zuletzt aus Versehen den falschen Menüpunkt ausgewählt? Mit großer Wahrscheinlichkeit ist Ihnen das auf einem Touchscreen und nicht bei der Bedienung mit der Maus passiert. Darüber hinaus gibt es einen weiteren, ganz wesentlichen Unterschied: Der Mauszeiger steht für sich alleine, der Finger hingegen ist Teil der Hand.



Die unterschiedlichen Arten, ein Smartphone zu halten, nach www.uxmatters.com (Bild 1)



Der Finger verdeckt auch einen Teil der Bedienoberfläche, wie Steven Hoober bei uxMatters zeigt (Bild 2)

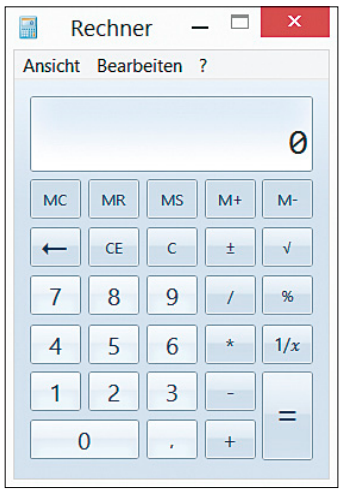


Nicht alle Bereiche sind gleich gut erreichbar (Josh Clark bei Alistapart) (Bild 4)

Das heißt, je nachdem, welchen Bereich man berührt, kann es passieren, dass dadurch ein Teil oder sogar ein wesentlicher Bestandteil des Inhalts nicht mehr sichtbar ist (Bild 2). Der Finger verdeckt nicht nur das, auf was er tippt, die Hand kann auch noch einen großen Teil des restlichen Screens überdecken. Im Extremfall sehen Sie den größten Teil des Screens nicht mehr, etwa wenn Sie mit der rechten Hand auf einem Laptop mit Touchscreen die linke obere Ecke berühren. Bei klassischen Geräten befinden sich deswegen die Bedienelemente immer unterhalb des Screens – denken Sie etwa an einen Taschenrechner (Bild 3). So ist sichergestellt, dass die Hand bei der Eingabe nicht die Anzeige verdeckt.

Ein weiterer Unterschied besteht darin, dass Sie mit der Maus sofort ohne große Mühe von der linken unteren zur rechten oberen Ecke wandern – ohne dass das aufwendig wäre; die dafür notwendige Bewegung ist kleiner als die mit der Maus bewirkte Strecke; zusätzlich gibt es Hilfsmittel wie das Scrollrad. Anders verhält es sich bei Touchscreens: Hier gibt es keine Abkürzungen, und weit auseinander liegende Elemente auf großen Screens zu bedienen kann physisch anstrengend werden.

Bisher wurden nur scheinbare Einschränkungen von Touchscreens genannt – aber die Touchinteraktion hat auch einen deutlichen Vorteil gegenüber der Mausinteraktion: die Unmittelbarkeit. Die Bedienung mit den Fingern kann sich intuitiver anfühlen.



Taschenrechner: Damit man bei der Eingabe nicht die Anzeige verdeckt, befinden sich Bedienelemente üblicherweise unterhalb der Anzeige wie hier beim Taschenrechner (Bild 3)

Eine entscheidende Verbesserung für Touchscreens erreichen Sie damit, dass Sie die Bedienelemente genügend groß gestalten. Laut Microsofts Richtlinien für Windows 8 variiert die typische Fingergröße von 8 mm bei einem Baby bis zu 19 mm bei einem Basketballspieler, die Normalsterblichen haben eine Fingergröße von 11 mm (Bild 3).

Für Bedienelemente wird deswegen im Allgemeinen eine Größe von 44px empfohlen – die man natürlich in em umrechnen sollte. Wenn man von der Entsprechung 16px = 1em ausgeht, bedeutet das eine Breite/Höhe von 2.75em. Um unerwünschten Ergebnisse bei der Vererbung zu vermeiden, benutzen Sie am besten gleich 2.75rem:

```
html {
  font-size: 100%;
}
.button {
  height: 44px;
  height: 2.75rem;
}
```

Damit die definierte Größe richtig funktioniert, müssen Sie natürlich die Meta-Viewport-Angabe einsetzen, die die automatische Verkleinerung auf Smartphones außer Kraft setzt:

```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

Die Größe von 44px und ihre em/rem-Entsprechungen sind nur eine grobe Richtlinie. Es gibt eine Reihe von Faktoren, die einen Einfluss auf die Wahl der passenden Größe haben:

- Die Bedienelemente können kleiner sein, wenn mehr Abstand zwischen ihnen ist.
- Umgekehrt gilt: Je näher mehrere Bedienelemente beieinander liegen, desto wichtiger ist, dass sie genügend groß sind.
- Außerdem kommt es darauf an, wie leicht oder schwer ein Bedienfeld erreichbar ist. Wenn Sie an die klassische Haltung von Smartphones denken, leuchtet ein, dass es Bereiche gibt, die man gut mit dem Daumen erreicht, und andere, an die man etwas umständlich herankommt (Bild 4).
- Ebenfalls zu berücksichtigen ist, was passiert, wenn der Benutzer danebentippt. Ein extremes Negativbeispiel wäre, dass ein Löschen- und ein Speichern-Button ganz nah beieinander liegen und noch dazu zu klein sind – in diesem Fall sind unerwünschte Aktionen vorprogrammiert.

Bei der Bedienung mit der Maus kann man unterscheiden zwischen dem Bewegen ▶

über einen Bereich (Hovern) und dem eigentlichen Klicken und Bewegen mit gedrückter Maustaste. Hover-Effekte lassen sich nutzen, um anzuzeigen, dass eine Interaktion möglich ist, oder auch um zusätzliche Informationen einzublenden.

Ein typisches Beispiel für zusätzliche Informationen sind die *title*-Attribute, deren Inhalte beim Überfahren mit der Maus eingeblendet werden. Neben diesen einfachen Tooltipps gibt es natürlich auch ausgefeiltere, optisch aufwendiger gestaltete Tooltipp-Lösungen mit JavaScript.

Die Sache mit dem Hover

Das Problem dabei: Diese Zusatzinformationen bleiben den Mausnutzern vorbehalten. Bei Touchscreens funktioniert Hover im Normalfall nicht. Allerdings gibt es auch Ausnahmen: Beispielsweise können Sie bei den Samsung S Pen-Stylus-Geräten den Stift über einem Element schweben lassen, wodurch Sie den Hover-Effekt auslösen.

Wie löst man jetzt das Problem mit der teilweise fehlenden Hover-Funktionalität? Es wäre praktisch, wenn man überprüfen könnte, ob ein Gerät Hover unterstützt oder nicht. Genau das ist in der nächsten Version der Media-Queries-Spezifikation des W3C vorgesehen. In Media Queries Level 4 werden neue Features eingeführt, auf die Sie testen können. Vorgesehen ist eine Abfrage nach sogenannten Interaktions-Media-Features, über die sich die Hover-Unterstützung abfragen lässt. Drei Werte sind für *hover* vorgesehen:

- *none* bedeutet, dass Hover nicht funktioniert oder sogar überhaupt kein Zeiger vorhanden ist.
- *on-demand* heißt, dass prinzipiell Hover unterstützt wird, der Benutzer aber einen besonderen Aufwand betreiben muss, um Hover-Aktionen auszulösen.
- Bei *hover* werden Hover-Aktionen unterstützt.

Sehen wir uns dazu ein Beispiel an: Gehen wir von einem Link aus, auf den ein weiteres Element mit Zusatzinformationen folgt. Diese Zusatzinformationen sollen – sofern Hover unterstützt wird –, zuerst einmal ausgeblendet werden und nur beim Hovern über den Link erscheinen:

```
<a class="ausloeser" href="#" >Hover me! </a>
<div class="zusatz">Zusätzliche Infos</div>
```

Dank der neuen Media-Features können wir jetzt dafür sorgen, dass nur bei Unterstützung für Hover das Ausblenden stattfindet. Für die Überprüfung setzen wir *@media (hover)* ein:

```
@media (hover) {
  .zusatz {
    display: none; }
  .ausloeser:hover + .zusatz {
    display: block; }
}
```

@media (hover) ist dabei eine Abkürzung für *@media (hover:hover)*. Im Beispiel wird *.zusatz* normalerweise angezeigt. Wenn hingegen ein Browser die @media-Query interpretiert und außerdem das Media-Feature *hover* unterstützt, wird *.zusatz* ausgeblendet und erst beim Hovern wieder angezeigt. Dieses neue Media-Feature wird bereits von Chrome, Safari ab 9, iOS 9.2 sowie im neuen Edge-Browser interpretiert (Bild 5) und wird sich in Zukunft sicher als nützlich erweisen.

Solche Beispiele lassen sich aber auch ohne Abfrage des Interaktions-Media-Features bauen, sodass zusätzliche Informationen auf Touchscreens nach Klick auf ein Element angezeigt werden. Prinzipiell wird nämlich, wenn ein Element per Tippen ausgewählt wird, zwar erwartungsgemäß die CSS-Pseudoklasse *:active* aktiviert, aber – und das ist entscheidend – zusätzlich auch *:hover*. Chris Wilson und Paul Kinlan zeigen bei HTML5rocks folgendes Beispiel, in dem die Zusatzinformationen auf Touchscreens per Klick eingeblendet werden:

```
<style>
img ~ .content {
  display: none;
}
img:hover ~ .content {
  display: block;
}
</style>

<img src="/awesome.png">
<div class="content">This is an awesome picture of me
</div>
```

Aber der entscheidende Unterschied liegt auf der Hand: Während der Benutzer mit der Maus „en passant“ die bei Hover erscheinenden Informationen entdeckt, muss er bei einem Touchscreen erst auf das Element tippen.

Wer auf der sicheren Seite sein möchte, ohne viel Aufwand zu betreiben, sollte Hover-Effekte nur für zusätzliche Verbesserungen und nicht für existenzielle Inhalte oder Informationen benutzen.

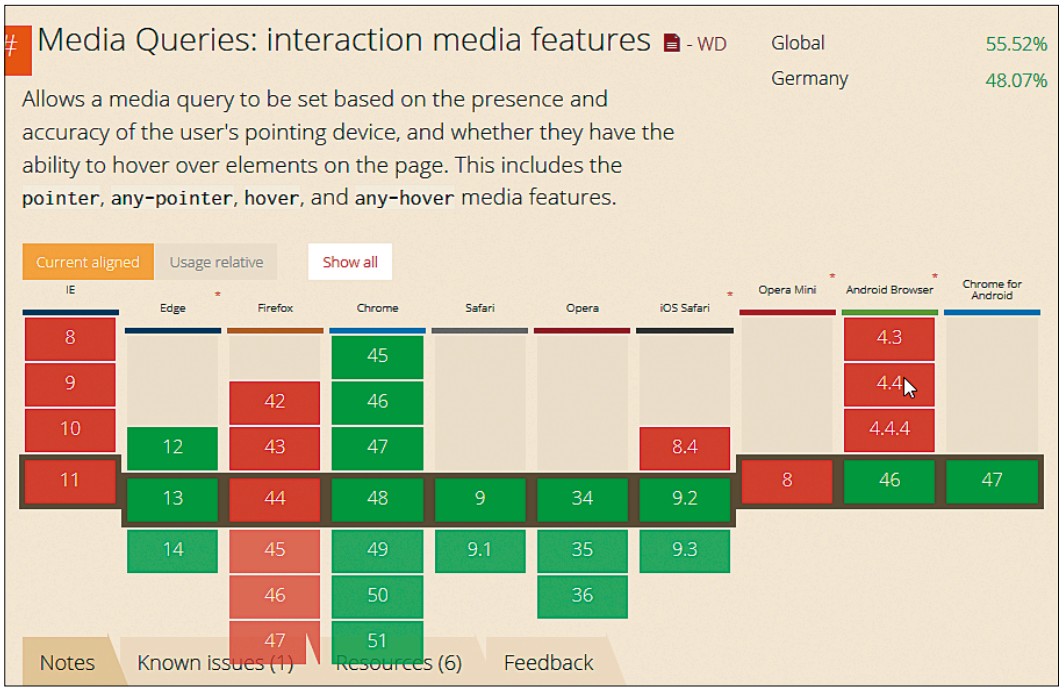
Touch per CSS prüfen

Gerade eben ging es um die Frage, wie man überprüfen kann, ob ein Browser Hover unterstützt. Aber wie sieht es denn allgemein mit der Überprüfung auf Touchkompatibilität per CSS aus?

Häufig wird für diesen Check die Größe des Viewports herangezogen und dabei von dem folgenden Prinzip ausgegangen: Kleiner Screen bedeutet Eingabe per Touch, großer Screen Eingabe per Maus und Tastatur.

Eine so grobe Vereinfachung funktioniert heutzutage aber nicht mehr: Denn es gibt große Bildschirme mit Touchscreen wie Tablets und kleine Laptops mit Mausbedienung. Deswegen verrät die Bildschirmgröße an sich nichts über die Art des Screens.

Auch dafür hält die nächste Version der Media-Spezifikation eine Lösung parat. Das dafür relevante Media-Feature ist pointer. Sie können überprüfen, ob eine exakte Platzierung möglich ist *(fine)* oder nicht *(coarse* = grob).



Browserunterstützung für die neuen, interaktiven Media-Features (Bild 5)

So könnte es sinnvoll sein, bei einem unpräzisen Eingabegerät eine Mindestgröße für Bedienelemente wie Checkboxen und Radiobuttons festzulegen:

```
@media (pointer:coarse) {
  input[type="checkbox"], input[type="radio"] {
    min-width: 30px;
    min-height: 40px;
  }
}
```

Die Unterstützung für dieses Media-Feature ist genauso gut wie beim Hover-Feature.

any-pointer und any-hover

Neben *pointer* und *hover* sieht die nächste Media-Queries-Spezifikation auch *any-pointer* und *any-hover* vor. Sie sind im Prinzip Erweiterungen von *pointer* oder *hover*, und zwar speziell für Geräte mit mehreren Eingabemöglichkeiten – ein typisches Beispiel dafür wäre ein Hybrid-Notebook.

Bei *pointer* oder *hover* wird die primäre Eingabeart eines Hybridgeräts berücksichtigt, bei den Varianten mit *any-* werden hingegen alle Inputmöglichkeiten herangezogen, die das jeweilige Gerät bietet. Daher können auch mehrere Charakteristika gleichzeitig zutreffen. Damit wissen Sie aber nur, was möglich ist, aber nicht, wie der Benutzer gerade interagiert.

Touch-APIs

Bisher ging es um besondere CSS-Angaben für Touchscreens; sehen wir uns an, wie es diesbezüglich mit JavaScript aussieht. Vorweg: Auf Touchscreens funktionieren normale Mausereignisse wie *click* ebenfalls – damit ist sichergestellt, dass auf Mausinteraktion ausgelegte Webseiten auch auf Touchscreens bedienbar sind, und bei einfachen Interaktionen genügt es, weiter mit *click* zu arbeiten. Es gibt aber zusätzlich spezielle APIs für Touch – auch und gerade für mobile Apps. Diese brauchen Sie, wenn Sie beispielsweise Gesten wie Swipe et cetera implementieren wollen.

Als Erstes zu erwähnen wäre hier das Touch-API, das zuerst im iPhone implementiert war, inzwischen aber eine W3C-Recommendation ist. Dieses API wird außer in iOS auch von Chrome und Edge unterstützt; es funktioniert in Android-Browsern, im Firefox muss erst das entsprechende Flag aktiviert werden.

In der Spezifikation werden eine Reihe von Events festgelegt, von denen drei – *touchstart, touchmove* und *touchend* – die wichtigsten sind.

Wenn ein Benutzer seinen Touchscreen berührt und das Touch-API unterstützt wird, dann werden, wie schon erwähnt, nicht nur Touch-, sondern auch Mausereignisse ausgelöst, und zwar in der folgenden Reihenfolge:

```
touchstart
touchmove
touchend
mouseover
mousemove
mousedown
mouseup
click
```

Mit *preventDefault()* innerhalb eines Touch-Ereignisses können Sie dafür sorgen, dass das entsprechende Mausereignis nicht mehr ausgelöst wird. Das sollten Sie aber nur für einzelne Elemente und nicht global durchführen, sonst verhindern Sie normale Benutzerinteraktionen wie das Scrollen.

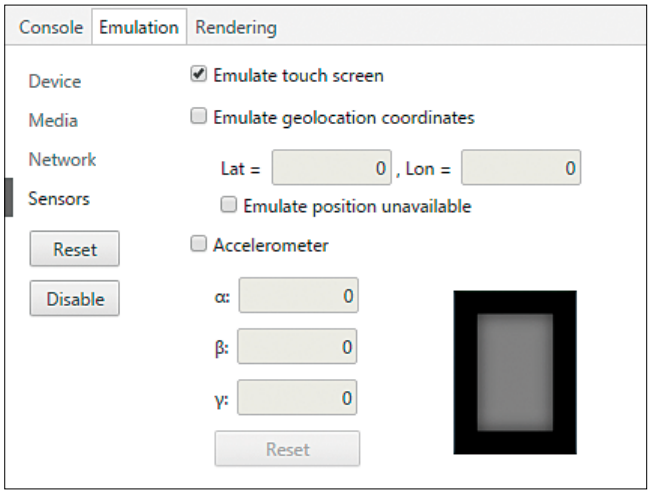
Der prinzipielle Nachteil der Touch-Events ist, dass die Entwickler eventuell unterschiedlichen Code für Maus- und ▶

Touch-Interaktionen schreiben müssen. Genau das ist bei dem alternativen API, den Pointer-Events, anders. Microsoft hat die Pointer-Events im Internet Explorer 10 auf Windows 8 eingeführt. Inzwischen werden die Pointer-Events vom W3C betreut und sind seit Februar 2015 eine Recommendation.

Touch, Maus, Stift

Das Besondere an den Pointer-Events ist, dass alle Eingabemöglichkeiten – Touch, Maus, Stift et cetera – zusammengefasst und gemeinsam behandelt werden. Dafür sind eine Reihe von Events wie *pointerenter, pointerover, pointerdown, pointermove, pointerup, pointerout, pointerleave* definiert.

Bei Bedarf kann man aber auch über *pointerType* den Eingabetyp abfragen und unterschiedliche Aktionen planen. Zudem gibt es die nützliche Eigenschaft *maxTouchPoints* des *navigator*-Objekts, über die sich die Anzahl an Touch-Berührungspunkten ermitteln lässt. Letztere wird gerne eingesetzt, um zu testen, ob es sich um einen Touchscreen handelt.



In den Entwickler-Tools von Chrome lässt sich auch die Emulation eines Touchscreens aktivieren (Bild 6)

Nimmt man die Touch- und die Pointer-Events zusammen, so lässt sich – theoretisch – auf folgende Art überprüfen, ob es sich um einen Touchscreen handelt. Überprüft wird dabei, ob das Ereignis *ontouchstart* (Touch) vorhanden ist oder die Eigenschaften *maxTouchPoints* (Pointer) größer als 0 ist – wobei die letzte Eigenschaft für den IE auch mit dem herstellerspezifischen Präfix geschrieben werden muss.

```
if (('ontouchstart' in window) ||
  (navigator.maxTouchPoints > 0) ||
  (navigator.msMaxTouchPoints > 0))
{
  /* Touch-fähig */
}
```

Derartige Tests finden sich häufig zur Überprüfung auf Touchscreens – aber sie sind problematisch. Denn eigentlich prüft man damit nur die Unterstützung für bestimmte APIs und nicht die Geräte-Features. Selbst wenn damit ermittelt werden könnte, dass ein Touchscreen zum Einsatz kommt, wissen Sie nicht, ob der Benutzer etwa bei einem Hybridgerät wirklich den Touchscreen nutzt oder doch die Eingabe per Maus bevorzugt. Gerade für Hybridgeräte ist es auch typisch, dass ein Benutzer zwischen den unterschiedlichen Eingabemethoden wechselt.

Schnelle Reaktion

Eine weitere Besonderheit zeigt sich bei Touchscreens: Wenn ein Besucher auf einen Link oder einen Button klickt, kann es vorkommen, dass die damit verbundene Aktion erst nach einer kurzen Verzögerung ausgeführt wird.

Die verzögerte Reaktion hat einen Grund: Es kann ja sein, dass der Benutzer doppelt tippen möchte, um zu zoomen oder Ähnliches. Deswegen wird mit der Verarbeitung eines einzelnen Tipp-Signals gewartet – ungefähr 300 ms, also deutlich spürbar.

Dieses Verhalten ist unschön und bewirkt, dass sich mobile Webseiten im Vergleich zu Apps träger anfühlen. Es gibt mehrere Möglichkeiten, diese Verzögerung zu verhindern.

Zunächst einmal reagieren bestimmte Browser auf Viewport-Angaben, bei denen das Zoomen von vornherein schon deaktiviert ist. Wenn bei der Viewport-*meta*-Angabe *user-scalabel=no* angegeben ist, oder wenn *minimum-scale* und *maximum-scale* auf denselben Wert gesetzt sind, werden die Klicks schneller interpretiert:

```
<meta name="viewport" content="width=device-width,
user-scalable=no">
```

Andererseits ist es gerade aus Usability-Sicht nicht empfehlenswert, das Zoomen zu deaktivieren. Deswegen hat WebKit im Dezember 2015 bekannt gegeben, dass ab jetzt bei Seiten mit *viewport*-Angabe das Doppeltippen deaktiviert ist, das heißt, dass es für schnelle Klickreaktionen genügt, eine *meta*-Angabe mit *width=device-width* zu nutzen. Das gilt ebenfalls für Chrome ab Version 32 auf Android.

Eine weitere Möglichkeit, die Verzögerung abzuschalten, besteht im Einsatz von CSS. Sie können durch die Angabe *touch-action: manipulation* das doppelte Tappen zum Zoomen deaktivieren und haben damit die gewünschte schnelle Klickreaktion. Diese CSS-Eigenschaft ist Teil der Pointer-Spezifikation:

```
.button {
  -ms-touch-action: manipulation;
  touch-action: manipulation;
}
```

Der Internet Explorer 10 benötigt die Angabe mit *-ms-*, ab Internet Explorer 11 geht es ohne. Die präfixlose Angabe wird ebenfalls von Chrome und – neu – in den Webkit Nightly Builds unterstützt. Sinnvoll ist es, diese CSS-Angabe für Buttons und Links anzugeben – so findet sich beispielsweise diese Definition bei dem Framework Bootstrap für alle Elemente mit der Klasse *btn*.

Außerdem gibt es fertige JavaScript-Lösungen, die sich um das Problem der verzögerten Reaktion beim Antippen kümmern.

Eleganter ist es aber, wenn man komplett darauf verzichten kann – und mit der *viewport*-Angabe oder der CSS-Eigenschaft *touch-action* sollte das in den meisten Fällen auch klappen.

Testen

Inzwischen emulieren die Entwickler-Tools der Browser teilweise bereits Touchfunktionalität. In Chrome funktioniert das folgendermaßen: Sie wechseln in die Entwickler-Tools und wählen den mobilen Modus durch Klick auf das Smartphone-Symbol. Dann können Sie bei *Emulation* – das ist der Tab neben *Konsole* – unter dem Link *Sensors* die Emulation von Touchscreens aktivieren (Bild 6). Bei emuliertem Touch verwandelt sich der Cursor in einen Kreis, der der Größe eines Fingers entsprechen soll, und Touch-Events wie *touchstart*, *touchmove* oder *touchend* werden ausgelöst.

Auch in den Entwickler-Tools von Firefox gibt es einen entsprechenden Button, wenn *Bildschirmgrößen testen* ausgewählt ist. Wird dieser Button aktiviert, so ändert sich dadurch zwar nicht die Anzeige des Mauscursors, aber Touchereignisse werden ausgelöst (Bild 7).

Emulation von Touch-Ereignissen in Firefox (Bild 7)

Wichtig ist aber wie immer der zusätzliche Test Ihrer Anwendung auf realen Geräten. Häufig werden Tests auf unterschiedlichen Geräten so durchgeführt, dass über eine Synchronisierungssoftware die Anzeige vom Desktop-Rechner auf verschiedenen mobile Geräten parallel geschaltet wird, wobei diese mobilen Geräte auf dem Tisch liegen. Das ist aber für Smartphones eine untypische Art der Nutzung – als mobile Geräte werden sie überwiegend in der Hand gehalten. Und die Bedienung ist eine ganz andere, wenn man das Gerät gleichzeitig halten und bedienen muss. Bei Ihren Tests sollten Sie das berücksichtigen und die Geräte selbst in der Hand haben.

Bedenken sollte man außerdem, dass die reale Nutzung der Smartphones oft gleichzeitig mit anderen Tätigkeiten geschieht, weswegen häufig auch Rechtshänder ein Smartphone mit der linken Hand bedienen, weil die rechte Hand anderweitig beschäftigt ist.

Kleine Verbesserung mit Wirkung

Wenn man sich die Techniken rund um Touchscreens ansieht, präsentiert sich einem ein eher uneinheitliches Bild: Hier CSS3-Features wie die neuen Media Queries, die erst teilweise in den Browsern angekommen sind, dort die konkurrierenden JavaScript-APIs für den Umgang mit Touch, und dann noch die Hybridgeräte, bei denen der Benutzer zwischen Maus und Touch wechseln kann.

Doch immerhin gibt es zwei Dinge, die sich leicht bei Webseiten durchführen lassen und die gleichzeitig eine deutliche Verbesserung für Touchscreen-Nutzer mit sich bringen:

- Achten Sie darauf, Links und Buttons ausreichend groß zu gestalten – im Zweifelsfall sollten Sie immer damit rechnen, dass die Webseite auf einem Touchscreen benutzt wird.
- Außerdem sollten Sie keine wichtigen Funktionalitäten oder Informationen von der Unterstützung von Hover abhängig machen. Hover eignet sich eher für fortgeschrittene, aber nicht für essenzielle Verbesserungen. ■

Links zum Thema

- Die typischen Smartphone-Haltungen
  www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php
- Finger verdeckt teilweise den angeklickten Bereich
  www.uxmatters.com/mt/archives/2013/03/common-misconceptions-about-touch.php
- Hover-Ereignisse auf Touchscreens auslösen
  www.html5rocks.com/en/mobile/touchandmouse
- Josh Clark bei Alistapart über das Halten von Geräten
  http://alistapart.com/article/how-we-hold-our-gadgets
- Touch-APIs
  www.w3.org/TR/touch-events
  https://www.w3.org/TR/pointerevents
- Warum sich Touchscreens nicht zuverlässig ermitteln lassen
  www.stucox.com/blog/you-cant-detect-a-touchscreen
- Media Queries Level 4
  https://drafts.csswg.org/mediaqueries-4
- Schnellere Reaktionen bei Klick
  https://webkit.org/blog/5610/more-responsive-tapping-on-ios

Dr. Florence Maurice
ist Autorin, Trainerin und Programmiererin in München. Sie schreibt Bücher zu PHP und CSS3 und gibt Trainings per Video. Außerdem bloggt sie zu Webthemen unter:
http://maurice-web.de/blog

WEBAPPLIKATIONEN FÜR RESPONSIVE DESIGN ANPASSEN

Modernisierung

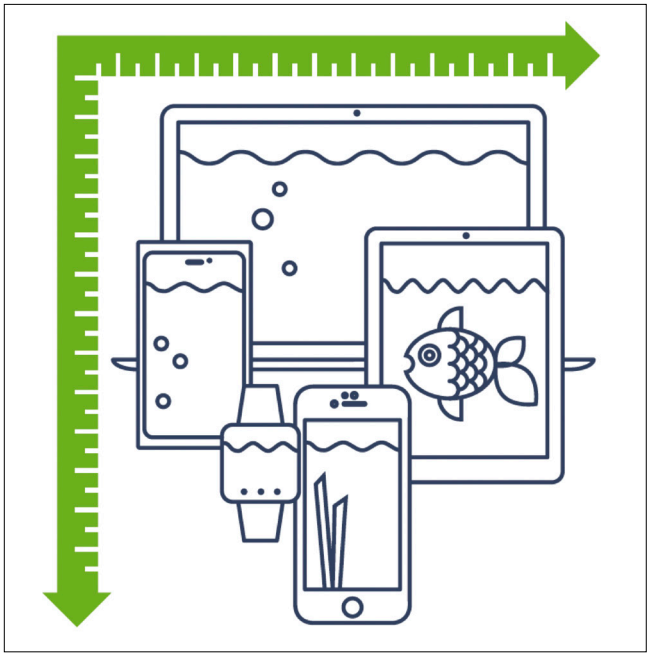
Die Modernisierung von Applikationen kann in den unterschiedlichsten Formen erfolgen.

Eine Form der Modernisierung ist die Überarbeitung im Einsatz befindlicher Desktop- und Webapplikationen, um sie optimal angepasst auch mit allen Möglichkeiten mobiler Endgeräte nutzen zu können.

Viele IT-Entwicklungsabteilungen in den Unternehmen haben umfangreiche Investitionen in Desktop-Apps getätigt, die sich über lange Jahre hinweg als sehr produktiv und zuverlässig erwiesen. In der Zwischenzeit haben sich die Anforderungen geändert, und es ist Zeit für einen Umbau und eine Neuinvestition für den mobilen Einsatz der Software.

Rein technisch betrachtet eignet sich eine Webapplikation bereits für mobile Endgeräte, denn diese verfügen über einen Browser, um die Anwendung zu öffnen. Der Haken bei der Sache: Wenn die Applikation für den Einsatz auf Desktops geschrieben wurde, ergibt sich auf Smartphones und Tablets eine nicht akzeptable Mobile User Experience: Es kommt zu Darstellungsfehlern und langen Ladezeiten, die Applikationen lassen sich nicht intuitiv bedienen und die Kernfunktionalität ist nicht akzeptabel.

Eine Umrüstung einer Webapplikation im Hinblick auf ein Responsive Design bietet eine interessante Möglichkeit für den Einstieg in die mobile Welt, da die Werkzeugunterstützung und die Expertise der Entwickler als Basis vorhanden sein dürften. Die Modernisierung einer Webapplikation mit Hilfe von Responsive-Design-Techniken erweist sich oft als der Weg des geringsten Widerstands. Entscheidend aber ist, mögliche Hindernisse zu erkennen, um potenzielle Probleme im weiteren Projektverlauf zu vermeiden (Bild 1).



Quelle: Progress

Der Bedarf an responsiven Design-Elementen für die Entwicklung plattformübergreifender und mobiler Applikationen steigt weiter (Bild 1)

Inline Styles legen Formate fest

Enthält ein HTML-Element die visuellen Stilvorgaben von einem Style Attribute, bezeichnet man das als Inline Style. Die Inline Styles waren gängig, bevor sich CSS Stylesheets durchsetzten:

```
<h1 style="color:blue;margin-left:30px;">
This is a heading.</h1>
```

In modernen Architekturen wie dem Responsive Webdesign verursachen Inline Styles schnell Probleme, da sie Vorrang gegenüber externen CSS-Werten haben und sie damit unwirksam machen. Daher müssen Inline Styles vor einer Migration zu einem Responsive Webdesign entfernt werden. Auch Frameworks mit einer langen Historie, wie ASP.NET Web Forms, können HTML-Code mit Inline Styles generieren. Wer ASP.NET Web Forms nutzt, sollte möglicherweise die Steuerelemente mit Alternativen wie der Telerik UI for ASP.NET AJAX ersetzen. Wer UI for ASP.NET AJAX einsetzt, sollte den Light Weight Rendering Mode aktivieren, mit dem sich semantische HTML-Elemente ohne Inline Styles erzeugen lassen:

```
<appSettings>
  <add key="Telerik.Web.UI.RenderMode"
  value="lightweight" />
</appSettings>
```

Ist eine Applikation bereits vor vielen Jahren entstanden, so ist die Wahrscheinlichkeit hoch, dass sie für das Seitenlayout HTML-Tabellen verwendet. In den Anfangsjahren des Webdesigns waren Seitenlayouts weit verbreitet. Mittlerweile aber sind tabellenbasierte Layouts obsolet:

```
<table border="0" cellpadding="0" cellspacing="0"
class="columns-container">
  <tr>
    <td valign="top">
      Column 1
    </td>
    <td valign="top">
      Column 2
```

```
    </td>
  </tr>
</table>
```

Moderne Seitenlayouts bestehen aus einer Kombination von semantischen HTML-Elementen, wie Artikel und Auswahl, und nichtsemantischen Elementen sowie CSS Styles. Manchmal bietet sich auch die Einführung eines Responsive-Grid-Systems wie Bootstrap, Foundation oder RadPageLayout an, mit dem sich alte Tabellenelemente ersetzen lassen.

HTML und Attribute

Tabellenelemente werden nicht mehr für Seitenlayouts verwendet, im HTML5-Standard sind sie für Data Grids immer noch relevant. Allerdings gibt es auch Elemente und Attribute, die veraltet sind und die Responsive-Design-Techniken behindern. HTML-Elemente wie *<font>* und Attribute wie *Height, Width, Marginheight* und andere obsolete Features wurden durch CSS Styles verdrängt. Solche Features können zu erheblichen Problemen führen, wenn sie mit CSS definierte Stile überschreiben. Abhängig von der Menge dieses veralteten Codes kann es sehr viel Arbeit bedeuten, eine Applikation im Hinblick auf Responsive Design zu überarbeiten.

Auch Applikationen, die per Mausklick gesteuerte Desktop-Bedienelemente wie Drag and Drop und Kontextmenüs enthalten, können bei der Modernisierung für die mobile Welt einen erheblichen Aufwand verursachen. Auf den mobilen Endgeräten haben die Benutzer keine großen Bildschirme und auch keine Maus zur Hand. Kleine Icons und Buttons müssen daher durch Touchscreen-geeignete Alternativen ersetzt werden. Das Gleiche gilt für Desktop-Bedienelemente, die Adaptive-Control-Techniken weichen müssen.

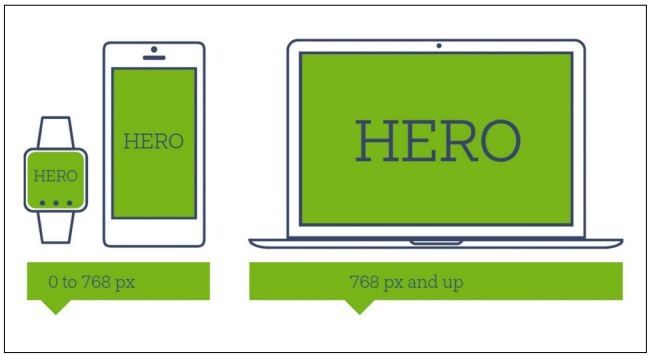
Eine der größten Herausforderungen für den Erfolg in der mobilen Welt ist der Benutzerkontext. Anwender wollen mit ihren mobilen Endgeräten schnell, benutzerfreundlich und mit einem minimalen Aufwand ihre Ziele erreichen.

Wichtig sind beispielweise der Ort, an dem Benutzer sich gerade aufhalten, die Netzqualität und die Möglichkeit, ein bestimmtes Vorhaben möglichst schnell abzuschließen. So ist es möglicherweise notwendig, komplexe Vorgänge in einzelne, kleinere Aktivitäten zu untergliedern, die individuell abgeschlossen werden können. Bei einer doch hin und wieder schwankenden Verbindungsqualität im Mobilfunknetz können einfache Aufgaben schrittweise abgearbeitet werden.

Anstelle einer umfangreichen Komplettrenovierung einer vorhandenen Webapplikation bietet es sich möglicherweise auch an, eine dazu komplementäre native oder hybride App zu entwickeln.

Der Einsatz von Bildern

Über den Einsatz von Images jeder Art und Größe in einer Applikation haben sich Entwickler lange Zeit keine weiteren Gedanken gemacht. Das ändert sich schlagartig bei einem Responsive Design. Früher wurden Bilder in einer fixen Größe verwendet. Da die Bildschirmgröße eines mobilen Endgeräts nicht von vornherein bekannt ist, müssen auch die Images flexibel sein (Bild 2).



Quelle: Progress

Da die Bildschirmgröße eines mobilen Endgeräts nicht von vornherein bekannt ist, müssen auch die Images flexibel sein (Bild 2)

Eine wichtige Rolle spielt darüber hinaus die Dateigröße des Bildes. Da im Mobilfunknetz eine geringere Bandbreite als bei einer Festnetzverbindung zur Verfügung steht, verursachen große Bilddateien schnell Performanceprobleme.

Eine Lösung für responsive Images ist das *<picture>*-Element. Es ist erst seit einiger Zeit Bestandteil von HTML5 und ermöglicht Entwicklern, unterschiedliche Image Sources zu definieren, um dann anhand von Faktoren wie der Auflösung oder der vorhandenen Bildschirmgröße das passende Image für das jeweilige mobile Endgerät zu laden. Allerdings ist es dazu auch erforderlich, Bilder in verschiedenen Auflösungen und Größen vorrätig zu haben. Eine weitere Möglichkeit bietet der Telerik Responsive Image Service, der mit vorhandenen Bildbeständen arbeitet und den Prozess mit Hilfe cloudbasierter Backend-Services automatisiert.

Die erfolgreiche Modernisierung existierender Applikationen ist von mehreren Faktoren abhängig, etwa der Komplexität der Bedienoberfläche und den Möglichkeiten, wie sich die Codebasis mit modernen HTML-Best-Practices verträgt. Bei vorhandenen Applikationen mit komplexen Formularen für die Dateneingabe und modalen Windows ist der Aufwand beträchtlich. Webapplikationen, die mit vergleichsweise modernen Techniken wie semantischen Elementen und CSS anstelle von Tabellen oder Fixed-Grid-Systemen erstellt wurden, sind schon eher geeignete Kandidaten.

Bereits im Vorfeld eines Modernisierungsprojekts empfiehlt sich ein kurzer Check, der ermittelt, wie viele und welche der erwähnten Hindernisse bei einer Applikation vorliegen. Anhand dieser Prüfung lässt sich dann auch sehr schnell entscheiden, ob die Modernisierung oder eine völlige Neuentwicklung der richtige Weg ist. ■

Ed Charbeneau
ist Developer Advocate bei Telerik, einem Tochterunternehmen von Progress.
www.telerik.com

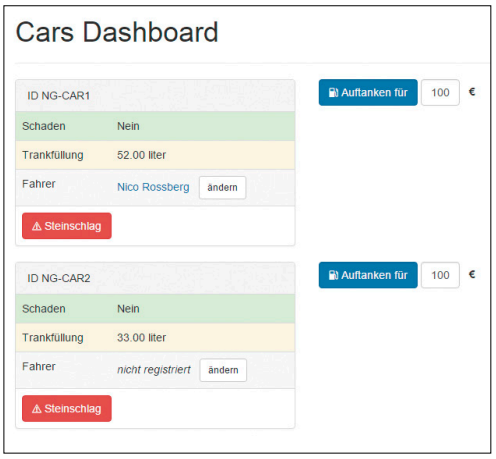
ROUTING MIT ANGULAR 2.0

Seitenwechsel ohne merkbaren Ladevorgang

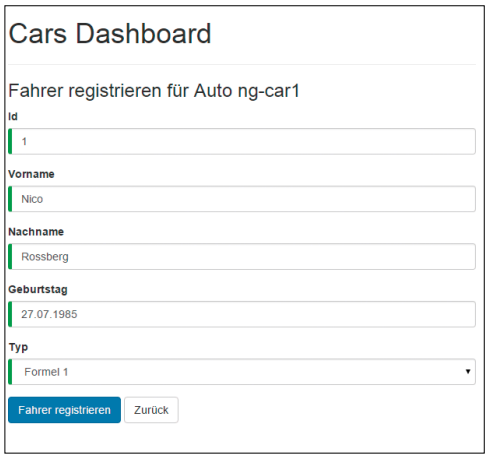
Zwischen verschiedenen Ansichten einer Single-Page-Anwendung navigieren.

Bei der Navigation in Single-Page-Anwendungen profitiert der Nutzer von Seitenwechseln ohne merkbaren Ladevorgang. Das Routing wurde in Angular 2 intensiv ausgebaut und deckt nun auch fortgeschrittene Anwendungsfälle ab.

Dies ist der fünfte und letzte Artikel aus unserer Reihe zu Angular 2. In den vorherigen Artikeln haben wir bereits SystemJS, Templates, Dependency Injection, Unit-Testing, HTTP-Kommunikation und die Verarbeitung von Formulardaten kennengelernt. Mit dabei ist stets das Car Dashboard, das kontinuierlich um neue Funktionen erweitert wird. In diesem Artikel wollen wir alle Bereiche der Anwendung über die neue Routing-Engine miteinander kombinieren (Bild 1, Bild 2).



Erste Komponente: Das Cars Dashboard mit allen verfügbaren Funktionen (Bild 1)



Zweite Komponente: Formular zum Eintragen der Fahrerdaten (Bild 2)

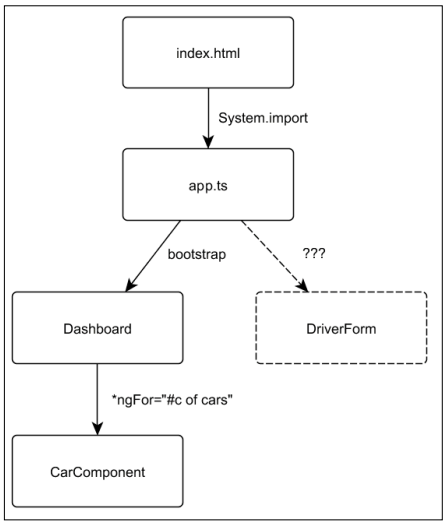
Wie gewohnt steht ein komplettes, lauffähiges Beispiel auf GitHub zur Verfügung. Sie finden alle besprochenen Inhalte unter https://github.com/Angular2Buch/angular2-routing.

Komponenten

Zugegeben, in der vorangegangenen Ausgabe mussten wir ein wenig schummeln. Wir haben neben der Komponente Dashboard eine zweite Komponente namens DriverForm vorgestellt und anhand dieser die Formularverarbeitung erläutert. Das Problem war: Beide Komponenten waren jeweils einzeln in die Website eingebunden (Listing 1).

Die hierarchische Anordnung der Komponenten aus der vorangegangenen Ausgabe sehen wir in Bild 3.

Nun wollen wir natürlich auch in der Lage sein, beide Ansichten gleichzeitig zu verwenden, damit wir durch die Anwendung navigieren können. Hier kommt das Prinzip des Routings in Spiel. Als Routing bezeichnen wir das Laden von Bereichen der Anwendung abhängig vom Zustand.



Ohne Routing kommt man hier nicht weiter (Bild 3)

Im Prinzip geschieht dasselbe, was wir auch manuell gemacht haben: Komponenten werden ausgetauscht. Der Dienst, der den Zustand der Angular-Anwendung verwaltet, nennt sich Router. Mittels Routing wollen wir nun Dashboard, Registrierungs-Formular und eine Detailansicht erreichbar machen. Alle Ansichten sollen vom Nutzer über verschiedene URLs aufrufbar sein.

Das Prinzip der Single-Page-Applikation sieht eine einzige HTML-Seite vor, deren tatsächliche Inhalte asynchron nachgeladen werden. Dabei findet in der Regel kein Neuladen der Seite statt. Das HTML5 History API liefert die technische Grundlage, um das Routing adäquat anzugehen.

Für ältere Browser, die dieses API nicht unterstützen, existieren Fallbacks, wie zum Beispiel die Verwendung von URLs mit einem *#Hash* (HashLocationStrategy). Der in Angular 2 standardmäßig vorhandene Rou-

ter nennt sich Component Router. Er kann mit verschiedenen Strategien verwendet werden, die bestimmen, wie der Router seinen Zustand persistiert.

Standardmäßig wird das HTML5 History API (PathLocationStrategy) verwendet. Es werden hierbei zur Identifikation der einzelnen Zustände gut lesbare URLs generiert. Wenn eine Grundstruktur der Anwendung mit mehreren Komponenten vorhanden ist, sind drei Schritte nötig, um den Router zu verwenden:

1. Routen konfigurieren: Wir weisen einem URL-Pfad eine zu ladende Komponente zu.
2. Komponenten anzeigen: Wir binden die geladene Komponente in das Template ein.
3. Routing booten: Wir aktivieren das Routing global in unserer Anwendung.

Routen-Konfigurationen werden in Angular mit Hilfe des Decorators RouteConfig aus dem Modul *angular2/router* vorgenommen. Anders als noch in AngularJS 1 muss die Routenkonfiguration nicht mehr zentral für die gesamte Anwendung definiert werden. Verzweigungen können am Ort des Geschehens, also bei der jeweiligen Komponente, bestimmt werden. Die erste Komponente, die beim Bootstrapping der Anwendung geladen wird, ist der Einstiegspunkt des Routers. Sie wird Root-Komponente genannt.

Dem Decorator RouteConfig wird eine also Liste von Routen übergeben, die für die Anwendung registriert werden sollen. Eine solche Route ist folgendermaßen aufgebaut:

```
{ path: '/path', name: 'MyRoute', component: MyComponent
}
```

Dabei ist *path* der URL-Pfad für diese Route, *name* der Name der Route (in CamelCase) und *component* die Komponente, die durch die Route geladen werden soll (zum Beispiel Dashboard oder DriverForm)

Der Routen-Name ist ein Bezeichner, den wir später verwenden, um die Route zu identifizieren. Es spricht nichts dagegen, die Namen der Komponente (als String) zu verwenden. Verwendet man eine Komponente im Routing mehrmals, muss man sich einen Namen ausdenken – denn er muss eindeutig sein.

Wir erzeugen also eine neue Komponente, die wir *DashboardApp* nennen (Listing 2).

Mit dem Schlüssel *useAsDefault* können wir eine Standardroute festlegen, die geladen wird, falls keine Route vom Nutzer aufgerufen wird. ▶

Listing 1: Code zum Einbinden der Root-Component

```
<!-- index.html -->
<dashboard>loading...</dashboard>

oder

<driver-form>loading...</driver-form>

// app.ts
bootstrap(Dashboard);

oder

bootstrap(DriverForm);

// dashboard.ts
@Component({selector: 'dashboard'})
@View({templateUrl: 'app/dashboard.html'})
export class Dashboard { }

// driver-form.ts
@Component({selector: 'driver-form'})
@View({templateUrl:
'app/components/driver-form/driver-form.html'})
export class DriverForm { }
```

Listing 2: Neue Komponente DashboardApp

```
@RouteConfig([
  { path: '/dashboard', as: 'Dashboard',
  component: Dashboard, useAsDefault: true },
  { path: '/drivers/...', as: 'Drivers',
  component: Drivers }
])
@Component({
  selector: 'dashboard-app',
  templateUrl: 'app/dashboard-app.html',
  directives: [ROUTER_DIRECTIVES]
})
export class DashboardApp { }
```

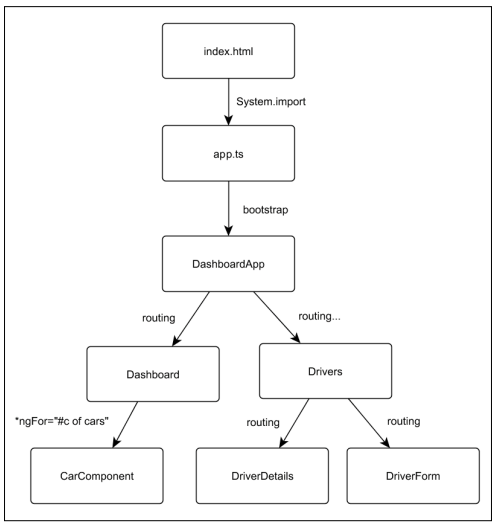
Alle Angular-2-Artikel im Überblick

Die nachfolgende Aufstellung gibt einen Überblick über die einzelnen Artikel unserer AngularJS-2-Serie.

- **Teil 1 – Modularer Code mit SystemJS und jspm** (12/2015)
  Code: https://github.com/Angular2Buch/angular2-module
- **Teil 2 – Template-Syntax und Web Components** (01/2016)
  Code: https://github.com/Angular2Buch/angular2-template-syntax
- **Teil 3 – Dependency Injection und Unit-Testing** (02/2016)
  Code: https://github.com/Angular2Buch/angular2-testing
- **Teil 4 – Formularverarbeitung und Validierung** (03/2016)
  Code: https://github.com/Angular2Buch/angular2-forms
- **Teil 5 – Routing** (04/2016)
  Code: https://github.com/Angular2Buch/angular2-routing

Wenn wir das Listing betrachten, so fällt auf, dass für die Komponente *DriverForm* gar keine Route definiert wurde. Stattdessen sieht man eine Notation mit drei Punkten. Der Component-Router von Angular 2 geht Hand in Hand mit dem Konzept der Komponenten und erlaubt die Vererbung von Routen. Das bedeutet, dass eine durch das Routing geladene Komponente weitere Routen-Konfigurationen besitzen kann et cetera.

Die Komponenten lassen sich beliebig tief verschachteln. So bleibt alles übersichtlich: Eine Komponente ist jeweils für die Verwaltung ihrer eigenen Routen zuständig und gibt die Verantwortlichkeit für Kind-Routen an die Kind-Komponente ab (Listing 3). Damit bleibt unsere Anwendung modular und die Komponenten haben definierte Schnittstellen. Um das Beispiel ein wenig zu komplettieren, haben wir eine weitere Komponente eingebunden. Sie soll die Details zu einem Fahrer anzeigen *(DriverDetails)* (Bild 4).



Mittels Routing sind nun alle Komponenten erreichbar (Bild 4)

Komponenten anzeigen

Nun sind die Routen und deren Pfade konfiguriert, doch ist noch nicht klar, wo die jeweiligen Komponenten im Template angezeigt werden sollen. Hierfür wird die Direktive *RouterOutlet* aus dem Modul *angular2/router* verwendet. Wo diese Direktive ins Template eingebunden wird, soll der Router die jeweils aktuelle Komponente dynamisch austauschen:

```
<!-- dashboard-app.html -->
<h1>Cars Dashboard</h1>
<hr>
<router-outlet></router-outlet>
```

Die Überschrift bleibt dadurch permanent in der Anwendung sichtbar. Darunter wird dann die jeweils angeforderte Komponente eingefügt. Die Kind-Komponente *Drivers* ist nur für die Verteilung der Routing-Anfragen auf den URL */drivers/...* zuständig. Sie benötigt auch die *DirektiveRouterOutlet* im Template, um die Komponente anzuzeigen. Mehr Aufgaben erfüllt *Drivers* aber nicht, deshalb fällt das Template denkbar kurz aus. Es bietet sich an, alles direkt im Quelltext der Komponente zu konfigurieren:

```
@Component({})
@View({
  directives: [ROUTER_DIRECTIVES],
  template: '<router-outlet>
  </router-outlet>'
})
@RouteConfig( /* [...] */ )
export class Drivers {}
```

Damit wären schon fast alle Änderungen durchgeführt. Nun muss das Routing nur noch aktiviert werden. Dazu verwenden wir die *ROUTER_PROVIDERS*. Diese Abhängigkeiten werden der *bootstrap*-Methode übergeben, mit der auch die Start-Komponente unserer Anwendung geladen wird. Sie konfiguriert den IoC-Container von Angular.

Die Mechanismen hinter Inversion of Control (IoC) und Dependendy Injection (DI) haben wir in der Ausgabe 2/2016 der **web & mobile developer** kennengelernt:

```
// app.ts
import {bootstrap} from 'angular2/platform/browser';
import {ROUTER_PROVIDERS} from 'angular2/router';

bootstrap(DashboardApp, [ROUTER_PROVIDERS]);
```

Benutzbar wird das Routing natürlich erst mit klickbaren Links innerhalb der Anwendung. Wichtig ist hierbei, dass Verlinkungen zwischen den Zuständen nicht manuell gesetzt, sondern automatisch erstellt werden. Das hat den Vorteil, dass der tatsächliche URL nicht vom Entwickler fest einprogrammiert werden muss. Außerdem wird so sichergestellt, dass die aktuelle Strategie verwendet wird. Bei der Verwendung des HTML5 History API wird zum Beispiel nicht wirklich eine neue Seite aufgerufen, sondern lediglich der Browserverlauf manipuliert.

Wir verwenden die Direktive *RouterLink* aus dem Modul *angular2/router*. In der Direktive geben wir an, welche Route verlinkt werden soll. Die Notation erfolgt als Liste, das sogenannte Link-Parameter-Array:

```
<a [routerLink]="['/Dashboard']">zum Dashboard</a>
```

Im ersten Element des Arrays geben wir die zu ladende Route an. Wir müssen hier den Namen verwenden, den wir bei der Routen-Konfiguration in der Komponente festgelegt haben. Wichtig ist, dass ein Link immer relativ zur aktuellen

Listing 3: Kind-Komponente Drivers

```
@Component({})
@View({
  /* [...] */
})
@RouteConfig([
  { path: '/details/:forCarId', as: 'Details',
  component: DriverDetails },
  { path: '/create/:forCarId',  as: 'Create',
  component: DriverForm }
])
export class Drivers {}
```

Komponente ist. Das gilt es zu berücksichtigen, wenn wir mehrere Komponenten verschachteln und die Routen vererben. Das Array hat im vorherigen Beispiel nur ein Element. Es kann aber beliebig viele Elemente besitzen, die jeweils auf weitere Kind-Routen verweisen.

Möchten wir mit einer Route weitere Werte übergeben, so verwenden wir als letztes Element im Array ein Objekt mit Routen-Parametern. Zur *KomponenteDriverDetails* wechselt man zum Beispiel mit folgendem Link, bei dem die ID des Fahrzeugs übergeben wird, sodass die Komponente die Details des gewählten Fahrers abrufen kann:

```
<a [routerLink]="['/Drivers', 'Details',
{ forCarId: car.id }]">Fahrer anzeigen</a>
```

Klicken wir auf den generierten Link, so wird die Adresszeile auf *http://example.org/drivers/create/ng-car1* aktualisiert. Dank des HTML5 History API verursacht der Wechsel kein echtes Neuladen der Seite.

Es kann aber vorkommen, dass die sichtbare Adresse per Reload oder per Bookmark aufgerufen wird. Dieser Fall wird von Angular ohne Probleme berücksichtigt. Es muss aber sichergestellt werden, dass auch der Webserver bereit für eine Single-Page-Anwendung ist. Bei einer Route handelt es sich nicht um einen echten Verzeichnispfad auf dem Server.

Bei einer unbekannten Adresse wie etwa *drivers/create/ng-car1* darf der Server allerdings nicht mit einem Fehler 404 antworten. Er muss so konfiguriert werden, dass für jeden Aufruf stets die *index.html* ausgeliefert wird. Dieses Verhalten kann im Apache mit dem Modul *mod_rewrite* erreicht werden. Dabei gilt zu beachten, dass nur Routen-Aufrufe auf die HTML-Seite verwiesen werden dürfen, nicht aber die Anwendung und statische Elemente (Bilder et cetera).

In der *index.html* biegt die folgende Angabe alle relativen Pfade wieder zurecht:

```
<!-- index.html -->
<base href="/">
```

Wenn man mittels Routen-Parametern Werte an die Komponente übergibt, so muss man diese natürlich auch empfangen können. Dazu dient die Klasse *RouteParams*, die wir in den Konstruktor injizieren und damit in der Komponente bekannt machen können.

Listing 4: RouteParams

```
import {RouteParams} from 'angular2/router';
@Component({ /* [...] */ })
export class DriverDetails {
  constructor(params: RouteParams) {
    var id = params.get('forCarId');
    console.log("Fahrer für Auto", id);
  }
}
```

Links zum Thema

- Offizielles Angular-2-Repository
  https://github.com/angular/angular
- Das neue Kommandozeilen-Tool für Angular
  https://github.com/angular/angular-cli
- Dokumentation von Angular zu Routing und Navigation
  https://angular.io/docs/ts/latest/guide/router.html

Die Instanz von *RouteParams* ist stets mit den jeweiligen Parametern der Route befüllt. Über die Methode *RouteParams.get()* können wir nun einen Parameter abrufen. Als Argument übergeben wir den Bezeichner, den wir bei der Definition des Links festgelegt haben (Listing 4).

Fazit

Die wichtigsten Bestandteile des Routings haben wir hiermit kennengelernt. Unsere Anwendung Cars Dashboard ist nun voll funktionsfähig. Ganz selbstverständlich haben wir dabei ein *RouterOutlet* im *RouterOutlet* verwendet. Solche verschachtelten Views waren in AngularJS 1 noch nicht mit Bordmitteln realisierbar.

Die Verwendung des AngularUI-Routers war daher die Defacto-Lösung für komplexere Szenarien. Der neue Router in Angular 2 ist zwar weiterhin nicht ganz so mächtig, deckt aber viel mehr Anwendungsfälle ab. Dies ist eine gute Entwicklung des Frameworks.

Mit diesem Artikel wollen wir unsere Reihe zum neuen Framework von Google abschließen. Mittels der vorgestellten Themen über fünf Ausgaben lässt sich bereits ein Most-Valuable-Produkt erstellen. Die GitHub-Repositories haben wir alle auf den neuesten Stand gebracht. Viele weitere Aspekte von Angular 2 gilt es zu erforschen. Wir – das sind Johannes Hoppe, Danny Koppenhagen, Ferdinand Malcher und Gregor Woiwode – laden Sie gern zu einer längeren Entdeckungsreise ein. ■

Johannes Hoppe
ist selbstständiger IT-Berater, Software-Entwickler und Trainer. Er arbeitet derzeit als Architekt für ein Portal auf Basis von .NET und AngularJS.
http://blog.johanneshoppe.de

Ferdinand Malcher
Ferdinand Malcher ist selbstständiger Software-Entwickler und Mediengestalter aus Leipzig. Seine Schwerpunkte liegen auf Webanwendungen mit Angular und Node.js.

DAS INDEXEDDB API

Datenbank im Browser

Das IndexedDB API bietet Webentwicklern clientseitige Datenbanken.

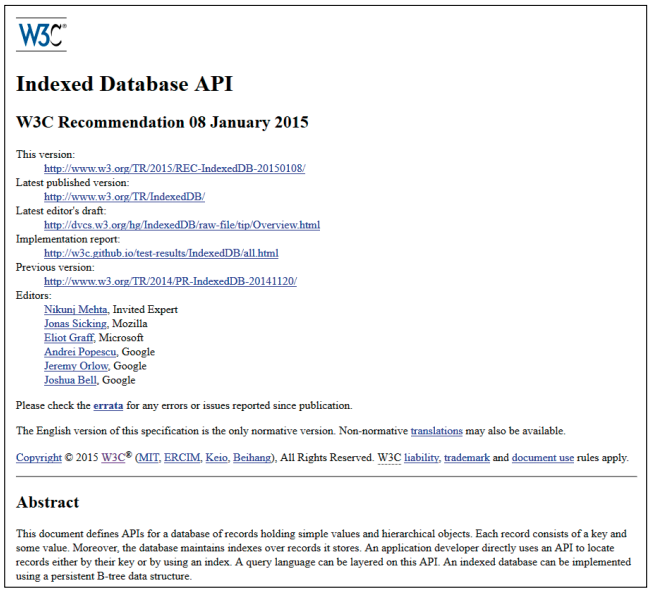
Vom Prinzip her funktioniert die browserinterne Datenbank genauso wie eine serverseitige Datenbank, das heißt, man kann in ihr Datensätze erstellen, Datensätze auslesen, aktualisieren oder wieder löschen. Im Fachjargon fasst man diese Operationen bekanntermaßen auch unter dem Begriff CRUD zusammen (create, read, update, delete).

Die Grundidee bei der IndexedDB ist es, JavaScript-Objekte unter bestimmten Schlüsseln abzuspeichern, das heißt, es handelt sich um einen sogenannten Key-Value Store (Bild 1). Vom Prinzip her funktioniert das wie bei der Datenstruktur Map: Die einzelnen Objekte in einer solchen Datenbank werden als Werte unter bestimmten Schlüsseln abgelegt, über die man anschließend wieder an das Objekt gelangt.

Den Kern einer IndexedDB bilden die sogenannten Object Stores beziehungsweise Objektspeicher. Diese sind in etwa vergleichbar mit Tabellen bei relationalen Datenbanken und enthalten die verschiedenen Schlüssel-Wert-Paare.

Öffnen einer Datenbank

Zugriff auf die IndexedDB erlangt man über das *window*-Objekt. Hinter dessen Eigenschaft *indexedDB* verbirgt sich ein Objekt vom Typ *IDBFactory,* über das sich unter anderem eine Datenbank öffnen lässt. Dazu stellt dieses Objekt die Methode *open()* zur Verfügung. Ihr übergibt man den Namen der Datenbank und optional als zweiten Parameter eine Versionsnummer. Letztere bestimmt intern das verwendete Datenbankschema für das Speichern von Objekten.



W3C

Indexed Database API

W3C Recommendation 08 January 2015

This version:
http://www.w3.org/TR/2015/REC-IndexedDB-20150108/
Latest published version:
http://www.w3.org/TR/IndexedDB/
Latest editor's draft:
http://dvcs.w3.org/hg/IndexedDB/raw-file/tip/Overview.html
Implementation report:
http://w3c.github.io/test-results/IndexedDB/all.html
Previous version:
http://www.w3.org/TR/2014/PR-IndexedDB-20141120/
Editors:
Nikunj Mehta, Invited Expert
Jonas Sicking, Mozilla
Eliot Graff, Microsoft
Andrei Popescu, Google
Jeremy Orlow, Google
Joshua Bell, Google

Please check the errata for any errors or issues reported since publication.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Copyright © 2015 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

This document defines APIs for a database of records holding simple values and hierarchical objects. Each record consists of a key and some value. Moreover, the database maintains indexes over records it stores. An application developer directly uses an API to locate records either by their key or by using an index. A query language can be layered on this API. An indexed database can be implemented using a persistent B-tree data structure.

Die W3C-Empfehlung zum IndexedDB API (Bild 1)

Als Rückgabewert liefert die Methode *open()* ein Objekt vom Typ *IDBOpenDBRequest.* Man hält also zu diesem Zeitpunkt noch nicht die Datenbank selbst (beziehungsweise ein diese repräsentierendes Objekt) in den Händen, sondern hat vielmehr lediglich eine Anfrage formuliert, dass man die entsprechende Datenbank öffnen möchte.

Um auf die Datenbank selbst beziehungsweise im Fehlerfall auf den entsprechenden Fehler zugreifen zu können, lassen sich an diesem Anfrageobjekt verschiedene Event Handler definieren: Über die Eigenschaft *onerror* lässt sich ein Event Handler für den Fehlerfall definieren, der beispielsweise dann ausgeführt wird, wenn der Nutzer im Browser keine Berechtigungen für den Zugriff auf die Datenbank gegeben hat.

Über die Eigenschaft *onsuccess* wiederum kann analog dazu ein Event Handler definiert werden, der immer dann aufgerufen werden soll, wenn die Datenbank erfolgreich geöffnet wurde (Listing 1).

Listing 1: Öffnen einer Datenbank

```
// Zugriff auf das Helferobjekt
let idbFactory = window.indexedDB;
// Öffnen der Datenbank ...
let request = idbFactory.open(
  // ... anhand des Namens ...
  'TestDatabase',
  // ... und optional der Version.
  1
);
// Event Handler für den Fehlerfall
request.onerror = (event) => {
  // Zugriff auf den Fehler
  let error = event.target.error;
  // Ausgabe des Fehlers
  console.error(error.message);
};
// Event Handler für den Normalfall
request.onsuccess = (event) => {
  // Zugriff auf die Datenbank
  let database = event.target.result;
  // Ausgabe: "TestDatabase"
  console.log(database.name);
  // Ausgabe: 1
  console.log(database.version);
};
```

Innerhalb dieser Event Handler hat man dann entsprechenden Zugriff auf das Fehlerobjekt (im Fehlerfall) oder das Datenbankobjekt (im Normalfall). So gelangt man beispielsweise über *event.target.result* innerhalb des *onsuccess*-Handlers an ein Objekt vom Typ *IDBDatabase,* über das man wiederum auf weitere Informationen wie den Namen, die Version oder auch die in der Datenbank enthaltenen Objektspeicher Zugriff hat und Objektspeicher anlegen und löschen sowie Transaktionen ausführen kann.

Erstellen einer Datenbank

Das Erstellen einer neuen Datenbank unterscheidet sich vom Code her prinzipiell nicht von dem Öffnen einer Datenbank. Es gilt nämlich: Versucht man, eine Datenbank zu öffnen, die es noch nicht gibt, wird diese Datenbank implizit erstellt.

Mit anderen Worten: Der Code für das Öffnen und das Anlegen einer Datenbank ist nahezu identisch. Der einzige Unterschied ist, dass beim Anlegen einer Datenbank auch die benötigten Objektspeicher angelegt werden sollten, beim späteren Öffnen der Datenbank dann nicht mehr.

Jetzt stellt sich natürlich die Frage, wie man zwischen dem Öffnen und dem Anlegen einer Datenbank unterscheidet. Die Antwort liefert der Event Handler *onupgradeneeded.* Dieser Event Handler wird nämlich immer dann aufgerufen, wenn es entweder eine angeforderte Datenbank noch nicht gibt und sie daher angelegt werden muss oder es die angeforderte Datenbank nicht in der angegebenen Version gibt.

In beiden Fällen erhält der aufgerufene Event Handler als Parameter ein Objekt vom Typ *IDBVersionChangeEvent,* über das man Zugriff auf die alte und die neue Version hat (Listing 2).

Da das *onupgradeneeded*-Event eben nur dann aufgerufen wird, wenn es eine Datenbank in dieser Form noch nicht gibt, ist der entsprechende Event Handler auch die geeignete Stelle, um Objektspeicher anzulegen oder, allgemeiner, das Datenbankschema zu definieren. Man kann sich *onupgradeneeded* quasi wie ein initiales Script vorstellen, in dem man die Datenbank beziehungsweise die benötigten Objektspeicher initialisieren kann.

Um die Funktionsweise von *onupgradeneeded* besser zu verstehen, schauen Sie sich das Skript aus Listing 3 an. Hier werden hintereinander fünf Anfragen für das Öffnen einer Datenbank formuliert, wobei bei der ersten Anfrage keine Versionsnummer angegeben wird, bei allen anderen Anfragen dagegen schon. Für alle Anfrageobjekte wird der gleiche Event Handler für das *onupgradeneeded*-Ereignis definiert, nämlich die Funktion *log(),* die lediglich jeweils die alte und die neue Version der Datenbank ausgibt. Ruft man nun dieses Skript das erste Mal auf, erzeugt es die in den Code-Kommentaren enthaltene Ausgabe.

Ruft man dagegen das Skript ein weiteres Mal auf, wird keiner der *onupgradeneeded*-Handler mehr aufgerufen. Dies aus verschiedenen Gründen: *request1* öffnet die Datenbank ohne Angabe einer Versionsnummer, wodurch einfach die letzte vorhandene Version (sprich die Version 5) verwendet wird. Das heißt, *request1* öffnet ohne Probleme, aber auch ohne Aufruf des *onupgradeneeded*-Handlers.

Ähnlich verhält es sich mit *request5.* Hier wird explizit die Version mit angegeben. Da diese mit der aktuellen Version übereinstimmt, wird auch hier die Verbindung zur Datenbank erfolgreich hergestellt.

Die Anfragen *request2, request3* und *request4* dagegen verwenden Versionsnummern, die kleiner als die aktuelle Versionsnummer sind, daher scheitern diese Anfragen und führen zu Fehlern (in der Form *The requested version (2) is less than the existing version (5)).*

Arbeiten mit Objektspeichern

Nachdem Sie nun wissen, wie Sie eine Verbindung zur Datenbank herstellen können, und die Funktionsweise von *onupgradeneeded* verstanden haben beziehungsweise wissen, in welchen Fällen dieses Ereignis ausgelöst wird, geht es als Nächstes daran, einen Objektspeicher anzulegen. In der Regel werden Sie dies nämlich innerhalb des *onupgrade-* ▶

Listing 2: Anlegen oder Ändern einer Datenbank

```
request.onupgradeneeded = (event) => {
  // Alte Version der Datenbank
  console.log(event.oldVersion);
  // Neue Version der Datenbank
  console.log(event.newVersion);
  // Zugriff auf die Datenbank
  let database = event.target.result;
}
```

Listing 3: onupgradeneeded-Handler

```
let idbFactory = window.indexedDB;
let DATABASE_NAME = 'TestDatabase7';
function log(event) {
  console.log('Alte Version: ' + event.oldVersion +
', '
  + 'Neue Version: ' + event.newVersion);
};
// Alte Version: 0, Neue Version: 1
let request1 = idbFactory.open(DATABASE_NAME);
// Alte Version: 1, Neue Version: 2
let request2 = idbFactory.open(DATABASE_NAME, 2);
// Alte Version: 2, Neue Version: 3
let request3 = idbFactory.open(DATABASE_NAME, 3);
// Alte Version: 3, Neue Version: 4
let request4 = idbFactory.open(DATABASE_NAME, 4);
// Alte Version: 4, Neue Version: 5
let request5 = idbFactory.open(DATABASE_NAME, 5);
request1.onupgradeneeded = log;
request2.onupgradeneeded = log;
request3.onupgradeneeded = log;
request4.onupgradeneeded = log;
request5.onupgradeneeded = log;
```

Tabelle 1: Schlüsselerzeugung bei der IndexedDB

keyPath	autoIncrement	Beschreibung
Nicht angegeben	*false*	Im Objektspeicher können alle Arten von Werten, insbesondere auch für primitive Datentypen wie Strings oder Zahlen, gespeichert werden. Allerdings muss bei jedem Hinzufügen ein Schlüssel explizit angegeben werden.
Angegeben	*false*	Im Objektspeicher können nur Objekte gespeichert werden (also keine primitiven Datentypen). Voraussetzung hierbei ist jedoch, dass diese Objekte eine gleichnamige Eigenschaft wie der angegebene keyPath haben.
Nicht angegeben	*true*	Im Objektspeicher können alle Arten von Werten, insbesondere auch für primitive Datentypen wie Strings oder Zahlen, gespeichert werden. Schlüssel werden automatisch erzeugt, können aber optional auch beim Hinzufügen von Werten angegeben werden.
Angegeben	*true*	Im Objektspeicher können nur Objekte gespeichert werden (also keine primitiven Datentypen). Schlüssel werden automatisch erzeugt und dem neuen Objekt – für den Fall, dass dieses nicht bereits eine gleichnamige Eigenschaft wie der angegebene keyPath hat – hinzugefügt. Gibt es dagegen die Eigenschaft im Objekt bereits, wird kein Schlüssel generiert, sondern der bereits hinterlegte Wert als Schlüssel verwendet.

needed-Handlers erledigen (schließlich will man einen Objektspeicher ja nur einmal anlegen).

Ein Objektspeicher lässt sich über die Methode *createObjectStore()* auf dem Datenbankobjekt erzeugen. Übergeben werden dabei zum einen der Name des Objektspeichers und zum anderen optional weitere Konfigurationsparameter. In Listing 4 beispielsweise wird ein Objektspeicher mit Namen *Books* angelegt und zusätzlich über den Konfigurationsparameter *keyPath* definiert, dass die Eigenschaft *isbn* (von Objekten, die in der Datenbank gespeichert werden) als Schlüssel dienen soll.

Die IndexedDB ist eine Datenbank, in der Objekte als Ganzes gespeichert werden und nicht wie bei relationalen Datenbanken beispielsweise einzelne Eigenschaften eines Objekts in einzelnen Spalten in einer Tabelle. Jedes Objekt wird dabei in dem Objektspeicher hinter einem Schlüssel abgelegt, über den später ein Zugriff auf das Objekt erfolgen kann. Was genau als Schlüssel verwendet wird, wird neben dem eben genannten Parameter *keyPath* auch über den Parameter *autoIncrement* des Konfigurationsobjekts bestimmt. Tabelle 1 zeigt die verschiedenen Kombinationen dieser beiden Parameter und was dies für die Schlüssel bedeutet.

Listing 4: Erstellen eines Objektspeichers

```
request.onupgradeneeded = (event) => {
  // Zugriff auf das Datenbankobjekt
  let database = event.target.result;

  // Erzeugen des Objektspeichers
  let objectStore = database.createObjectStore(...
    // ... wobei ein Name ...
    'Books',
    // und optionale Parameter mitgegeben werden.
    { keyPath: 'isbn' }
  );
};
```

Tabelle 2: Erlaubte Werte zur Transaktionssteuerung

Wert	Beschreibung
readonly	Daten können gelesen, aber nicht geändert werden.
readwrite	Daten können sowohl gelesen als auch geändert werden.
versionchange	Alle Arten von Operationen auf den Daten sind erlaubt (das heißt Lesen und Schreiben). Dies beinhaltet auch Operationen, die Objektspeicher und Indizes löschen oder anlegen.

Operationen, die man auf einer Datenbank ausführt, werden im Fall der IndexedDB in Transaktionen gekapselt. Wie das funktioniert, zeigt Listing 5. Über die Methode *transaction()* gelangt man über das Datenbankobjekt an ein Transaktionsobjekt (vom Typ *IDBTransaction)*. Als ersten Parameter übergibt man dieser Methode ein Array von Objektspeichernamen, auf welche im Rahmen der Transaktion zugegriffen werden soll. Ist dies nur ein einzelner Objektspeicher, kann statt eines Arrays auch eine einzelne Zeichenkette übergeben werden.

Modus der Transaktion bestimmen

Über den (optionalen) zweiten Parameter lässt sich der Modus der Transaktion bestimmen: Mögliche Werte sind hier *readonly* (Daten können nur gelesen werden), *readwrite* (Daten können gelesen und geschrieben werden) sowie *versionchange* (Daten, Objektspeicher sowie Indizes können verändert werden).

Gibt man keinen Parameter an, wird standardmäßig *readonly* als Basis verwendet. Da es an dieser Stelle aber darum geht, Objekte zum Objektspeicher hinzuzufügen (also Daten in die Datenbank zu schreiben), muss im Beispiel der Wert *readwrite* explizit angegeben werden (Tabelle 2).

Über das Objekt vom Typ *IDBTransaction,* das in der Variablen *transaction* gespeichert wird, gelangt man im nächsten Schritt an die (innerhalb der Transaktion zur Verfügung stehenden) Objektspeicher. Im Beispiel wird auf diese Weise der

zuvor angelegte Objektspeicher mit Namen *Books* geöffnet. Das zurückgegebene Objekt ist vom Typ *IDBObjectStore* und verfügt über verschiedene Methoden, um beispielsweise Objekte hinzufügen zu können, Objekte zurückzugeben, Objekte löschen zu können und einiges mehr. Im Beispiel wird die Methode *add()* verwendet, um die Objekte aus dem *books*-Array einzeln dem Objektspeicher hinzuzufügen. Optional kann dieser Methode (im Beispiel nicht gezeigt) als zweiter Parameter der Schlüssel übergeben werden, unter dem das Objekt im Objektspeicher abgelegt werden soll.

Listing 5: Hinzufügen von Objekten

```
let idbFactory = window.indexedDB;
let request = idbFactory.open(
  'TestDatabase',
  9
);
let books = [
  {
    'isbn': '978-3836217408',
    'title': 'Schrödinger programmiert Java',
    'author': 'Philip Ackermann'
  },
  {
    'isbn': '978-3836223799',
    'title': 'Professionell entwickeln mit
    JavaScript',
    'author': 'Philip Ackermann'
  }
]
request.onerror = (event) => {
  let error = event.target.error;
  console.error(error.message);
};
request.onsuccess = (event) => {
  let database = event.target.result;
  // Öffnen der Transaktion
  let transaction = database.transaction(
    ['Books'],
    // Schreibender Zugriff
    "readwrite"
  );
  // Öffnen des Objektspeichers
  let objectStore =
  transaction.objectStore('Books');
  books.forEach((book) => {
    // Hinzufügen der Objekte
    objectStore.add(book);
  });
};
request.onupgradeneeded = (event) => {
  let database = event.target.result;
  let objectStore = database.createObjectStore(
    'Books',
    { keyPath: 'isbn' }
  );
}
```

Lesen von Objekten

Hat man einmal das Prinzip verstanden, wie sich ein Objektspeicher öffnen lässt, fällt das weitere Arbeiten damit nicht schwer. Um beispielsweise Daten aus einem Objektspeicher zu lesen, geht man vom Prinzip her vor wie bereits zuvor auch: Das heißt, zunächst die Verbindung zur Datenbank herstellen, anschließend eine Transaktion starten *(readonly* reicht in diesem Fall aus) und sich per Methode *objectStore()* den entsprechenden Objektspeicher holen. Das kennen Sie schon alles vom Vorangegangenen, und es ist nur der Vollständigkeit halber in Listing 6 noch einmal mit aufgeführt.

Auf Objekte im Objektspeicher zugreifen lässt sich dann mit Hilfe der Methode *get()* und des Schlüssels, den man dieser Methode als Parameter übergibt. Als Rückgabewert liefert *get()* ein Anfrageobjekt vom Typ *IDBRequest,* wie Sie es schon von der Methode *open()* für das Öffnen einer Datenbank kennen (auch wenn es sich dort genau genommen um ein Objekt vom Typ *IDBOpenDBRequest* handelt). Auch hier lassen sich also Event Handler für den Normalfall (über die Eigenschaft *onsuccess)* und den Fehlerfall (über die Eigenschaft *onerror)* definieren. Vorausgesetzt, in dem Objektspeicher gibt es zu dem übergebenen Schlüssel ein entspre- ▶

Listing 6: Lesender Zugriff

```
let idbFactory = window.indexedDB;
let request = idbFactory.open('TestDatabase5', 9);
request.onerror = (event) => {
  let error = event.target.error;
  console.error(error.message);
};
request.onsuccess = (event) => {
  let database = event.target.result;
  // Öffnen der Transaktion
  let transaction = database.transaction('Books');
  // Öffnen des Objektspeichers
  let objectStore =
  transaction.objectStore('Books');
  // Suche nach Schlüssel
  let request = objectStore.get('978-3836223799');
  // Event Handler für Fehlerfall
  request.onerror = (event) => {
    // Ausgabe der Fehlermeldung
    console.error(event.target.error.message);
  };
  // Event Handler für Normalfall
  request.onsuccess = (event) => {
    // Ausgabe des Objekts
    console.log(request.result);
  };
};
```

chendes Objekt und es kommt nicht zum Fehler, hat man innerhalb des *onsuccess*-Handlers Zugriff auf dieses Objekt.

Löschen von Objekten

Auch das Löschen von Objekten aus einem Objektspeicher funktioniert bezüglich des Codes nach gleichem Prinzip. Da man durch das Löschen die Daten in der Datenbank verändert, ist es notwendig, die entsprechende Transaktion im Modus *readwrite* zu starten. Um ein Objekt zu löschen, bedient man sich der Methode *delete()* am Objektspeicher, wobei die Methode als Parameter den Schlüssel des zu löschenden Objekts erwartet. Rückgabewert ist wie auch zuvor ein Anfrageobjekt vom Typ *IDBRequest* (Listing 7).

Neben dem Hinzufügen und dem Löschen von Objekten aus dem Objektspeicher braucht man noch eine Möglichkeit,

Links zum Thema

- IndexedDB
  https://www.w3.org/TR/IndexedDB

Listing 7: Löschen eines Objekts

```
let idbFactory = window.indexedDB;
let request = idbFactory.open('TestDatabase5', 9);
request.onerror = (event) => {
  let error = event.target.error;
  console.error(error.message);
};
request.onsuccess = (event) => {
  let database = event.target.result;
  // Öffnen der Transaktion
  let transaction = database.transaction(
    'Books',
    // Schreibender Zugriff
    'readwrite'
  );
  // Öffnen des Objektspeichers
  let objectStore =
  transaction.objectStore('Books');
  // Löschvorgang
  let request =
  objectStore.delete('978-3836217408');
  // Event Handler für Normalfall
  request.onerror = (event) => {
    // Ausgabe der Fehlermeldung
    console.error(event.target.error.message);
  };
  // Event Handler für Normalfall
  request.onsuccess = (event) => {
    // Löschvorgang erfolgreich
  };
};
```

Listing 8: Aktualisieren eines Objekts

```
let idbFactory = window.indexedDB;
let request = idbFactory.open('TestDatabase5', 9);
request.onerror = (event) => {
  let error = event.target.error;
  console.error(error.message);
};

request.onsuccess = (event) => {
  let database = event.target.result;
  // Öffnen der Transaktion
  let transaction = database.transaction(
    'Books',
    // Schreibender Zugriff
    'readwrite'
  );
  // Öffnen des Objektspeichers
  let objectStore =
  transaction.objectStore('Books');
  // Lesen des Objekts
  let request = objectStore.get('978-3836217408');
  request.onerror = (event) => {
    console.error(event.target.error.message);
  };

  request.onsuccess = (event) => {
    let book = request.result;
    // Aktualisieren des Objekts
    book.title = 'Schrödinger programmiert Java'
    + ' - Das etwas andere Fachbuch';
    // Speichern des Objekts
    let requestUpdate = objectStore.put(book);
    requestUpdate.onerror = (event) => {
      // Fehler
    };
    requestUpdate.onsuccess = (event) => {
      // Erfolgreich gespeicher
    };
  };
};
```

Objekte im Objektspeicher zu aktualisieren. Vom Prinzip ist alles wieder altbekannt: Öffnen der Datenbank, Erstellen einer Transaktion im *readwrite*-Modus, Öffnen des Objektspeichers und Lesen des entsprechenden Objekts über die Methode *get()*. An dem auf diese Weise gelesenen Objekt nimmt man anschließend die gewünschten Änderungen vor und verwendet daraufhin die Methode *put()* auf dem Objektspeicherobjekt, um das Objekt im Objektspeicher zu aktualisieren (Listing 8).

Verwenden von Cursorn

Möchte man alle Objekte eines Objektspeichers auslesen, bieten sich sogenannte Cursor an. Über die Methode *open-*

Listing 9: Verwenden eines Cursors

```
let idbFactory = window.indexedDB;
let request = idbFactory.open(
  'TestDatabaseCursor',
  1
);
let books = [
  {
    'isbn': '978-3836217408',
    'title': 'Schrödinger programmiert Java',
    'author': 'Philip Ackermann'
  },
  {
    'isbn': '978-3836223799',
    'title': 'Professionell entwickeln mit
    JavaScript',
    'author': 'Philip Ackermann'
  }
]
request.onerror = (event) => {
  let error = event.target.error;
  console.error(error.message);
};
request.onsuccess = (event) => {
  let database = event.target.result;
  let transaction = database.transaction('Books');
  let objectStore = transaction.
objectStore('Books');
  var request = objectStore.openCursor();
  let books = [];
  request.onsuccess = (event) => {
    let cursor = event.target.result;
    if (cursor) {
      console.log(cursor.key);
      console.log(cursor.value);
      books.push(cursor.value);
      cursor.continue();
    } else {
      console.log(books);
    }
  };
};
request.onupgradeneeded = (event) => {
  let database = event.target.result;
  let objectStore = database.createObjectStore
  ('Books', { keyPath: 'isbn' });
  books.forEach((book) => {
    objectStore.add(book);
  });
}
```

Cursor() erzeugt man einen Request für das Erstellen eines Cursors. Im entsprechenden *success*-Event-Handler gelangt man über das Event-Objekt an das Cursor-Objekt selbst. Dieses wiederum stellt zwei Eigenschaften bereit: Die Eigenschaft *key* enthält den Schlüssel des jeweiligen Eintrags, die Eigenschaft *value* dagegen den Wert selbst, also das Objekt. Über die Methode *continue()* veranlasst man den Cursor (soweit vorhanden), zum nächsten Eintrag weiterzugehen, was wiederum dafür sorgt, dass der *success*-Event-Handler ausgeführt wird (Listing 9).

Tabelle 3: Die verschiedenen Typen des IndexedDB API

Typ	Beschreibung
IDBCursor	Dient dazu, über die Objekte in einem Objektspeicher und Indizes zu iterieren.
IDBCursorWithValue	Dient dazu, über Objektspeicher und Indizes zu iterieren und jeweils den aktuellen Wert zurückzugeben.
IDBDatabase	Repräsentiert eine Datenbankverbindung.
IDBFactory	Repräsentiert den Typ der indexedDB-Eigenschaft.
IDBIndex	Dient dazu, auf die Metadaten eines Index zugreifen zu können.
IDBKeyRange	Dient dazu, Schlüsselbereiche von Datensätzen festzulegen.
IDBObjectStore	Repräsentiert einen Objektspeicher in einer Datenbank.
IDBOpenDBRequest	Enthält das Ergebnis für Anfragen, die eine Datenbank öffnen. Wird von der *open()*-Methode von IDBFactory zurückgegeben.
IDBRequest	Enthält das Ergebnis für Anfragen, die an eine Datenbank gestellt werden.
IDBTransaction	Repräsentiert eine Transaktion, die auf einer Datenbank ausgeführt wird.
IDBVersionChangeEvent	Repräsentiert das Event-Objekt für das *upgradeneeded*-Event, das ausgelöst wird, wenn eine Datenbank unter Verwendung einer höheren Versionsnummer geöffnet wird.

Fazit

Das IndexedDB API bietet Zugriff auf eine browserinterne Datenbank und wird von nahezu allen namhaften Browsern unterstützt. Hat man einmal das Prinzip verstanden, ist das Arbeiten mit der IndexedDB recht einfach. Eine Übersicht der Objekttypen (beziehungsweise Interfaces), die in dem API definiert sind, ist in Tabelle 3 aufgelistet. ■

Philip Ackermann
arbeitet beim Fraunhofer-Institut für Angewandte Informationstechnologie FIT an Tools zum teilautomatisierten Testen von Web Compliance und ist Autor zweier Fachbücher über Java und JavaScript.
http://philipackermann.de

FUNKTIONAL-REAKTIV PROGRAMMIEREN MIT UND RXJS UND CYCLE.JS

Asynchrone Datenströme

Mit funktional-reaktiver Programmierung werden Programme deklarativ beschrieben.

JavaScript ist die wohl verbreitetste funktionale Programmiersprache. Die Bibliothek RxJS ist die JavaScript-Implementierung von Microsofts API ReactiveX, mit dem asynchrone Programmierung erleichtert werden soll. Grundlage dafür sind asynchrone Datenströme, die man Observables nennt. Das UI-Framework Cycle.js nutzt RxJS, um Bedienoberflächen deklarativ zu erzeugen.

2015 war das Jahr, in dem funktionale Programmierkonzepte die JavaScript-Community im Sturm eroberten. Der Vorteil von puren Funktionen, bei welchen der berechnete Wert ausschließlich von den Eingabeparametern abhängt, liegt in der einfachen Nachvollziehbarkeit und der damit auch besseren Testbarkeit.

Ein solcher Programmbaustein ist eben nicht abhängig von Seiteneffekten aus seiner Umgebung. Ein weiterer interessanter Aspekt bei der Vermeidung von Seiteneffekten betrifft Datenstrukturen: Gerade bei objektorientierter Programmierung und wenn Schleifen ins Spiel kommen, hat man es nahezu zwangsweise mit Seiteneffekten zu tun. Objekte kapseln veränderbaren Zustand (mutable State), und dieser verträgt sich im Allgemeinen nicht gut mit asynchroner Programmierung.

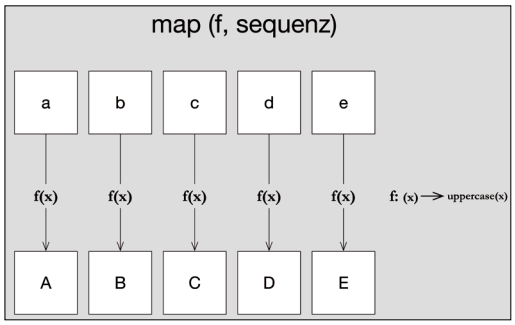
Schleifen berechnen keinen Rückgabewert

In JavaScript gibt es *for()*, *while()* und *do/while*-Schleifen, und diese verhalten sich im Prinzip auch wie ihre Gegenstücke in C oder Java. Sie berechnen keinen Rückgabewert. Möchte man irgendeinen Effekt erzielen, so benötigt man deshalb Seiteneffekte innerhalb der Schleife: Eine Lösung ist die Zuweisung an eine äußere Variable. Aber auch Ein- und Ausgaben wie mit console.log oder die Manipulation des Browser-DOM sind Seiteneffekte, mit denen Ergebnisse einer Schleife weitergegeben werden können.

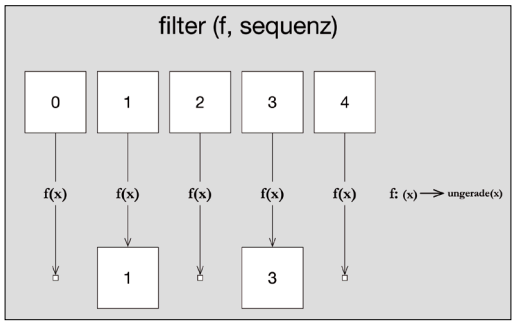
Fragt man erfahrene JavaScript-Programmierer, wie häufig sie eines der eingebauten Schleifenkonstrukte benutzen, so hört man oft: Nahezu nie.

Der Grund: Man kommt mit einer kleinen Zahl an funktionalen Operatoren aus, um nahezu jeden erdenklichen Anwendungsfall einer Schleife zu ersetzen. Das Ergebnis ist ein oft kürzerer Programmcode, der darüber hinaus auch noch leichter verständlich ist.

Bei funktionaler Programmierung gilt das folgende Prinzip: Man beschreibt stets das Ergebnis, und nicht die Schritte, um zu diesem Ergebnis hinzugelangen (deklarativ).



Map(): Schematische Darstellung der *map()*-Funktion (Bild 1)



Filter(): Schematische Darstellung der *filter()*-Funktion (Bild 2)

Funktionen höherer Ordnung

Das Mittel zum Zweck nennt man Funktion höherer Ordnung. Gemeint sind damit Funktionen, die andere Funktionen als Parameter erhalten oder als Rückgabewert liefern. Die wohl bekanntesten vier Beispiele für Funktionen höherer Ordnung sind *map()*, *filter()*, *reduce()* und *zip()*.

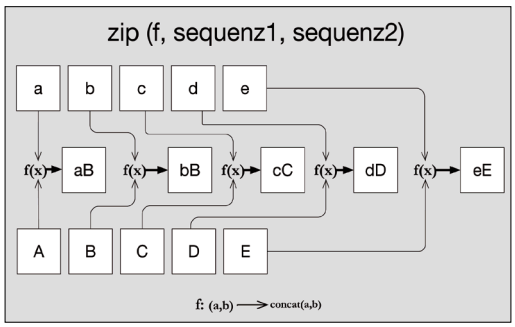
Diese vier funktionalen Operatoren gehören zu den wichtigsten Konzepten, die man in der eigenen Programmierlaufbahn lernen kann. Da sie eine wichtige Grundlage sind, sollen sie hier noch einmal genauer betrachtet werden.

Bild 1 zeigt, wie die *map()*-Funktion arbeitet: Jedes einzelne Element einer Quell-Sequenz wird auf eine Ziel-Sequenz anhand einer Abbildungsfunktion *f* gewandelt. Es entsteht eine neue Sequenz mit der gleichen Länge wie die Quellsequenz. Die Werte entsprechen stets dem Ergebnis der Abbildungsfunktion anhand des jeweiligen Elements. Die Abbildungsfunktion sollte pur sein. Das bedeutet, sie sollte für denselben Parameterwert auch stets dasselbe Ergebnis liefern und keine Seiteneffekte produzieren.

Auch *filter()* (Bild 2) ist ein Operator, der eine Sequenz auf eine andere abbildet. Allerdings besteht die resultierende Sequenz nur aus jenen Elementen der Quell-Sequenz, für welche die sogenannte Prädikatsfunktion *wahr* liefert. Trifft das auf alle Elemente zu, ist die Zielsequenz genauso lang wie die Quellsequenz. Trifft es auf kein Element zu, dann ist das Ergebnis eine leere Sequenz.

Dabei ist *zip()* (Bild 3) insofern besonders, als es wie ein *map()* über zwei Sequenzen funktioniert. Wie die zwei Seiten eines geöffneten Reißverschlusses werden die beiden Sequenzen zu einer gemeinsamen Zielsequenz zusammengefügt.

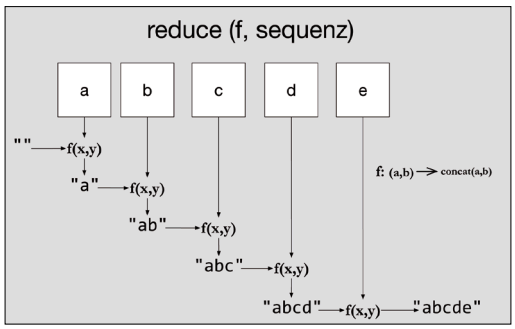
Dabei ist das jeweils resultierende Element das Ergebnis der übergebenen Abbildungsfunktion, welche die beiden Elemente der Sequenzen als Parameter erhält und das Zielelement zurückliefert. Mit *zip()* kann man also mehrere Sequenzen miteinander verknüpfen.



Zip(): Schematische Darstellung der *zip()*-Funktion (Bild 3)

Mächtige Funktion

Eine sehr mächtige Funktion ist *reduce()* (Bild 4): Ausgehend von einem initialen Wert entsteht Schritt für Schritt ein neuer Wert. Dazu wird der aktuelle Ergebniswert gemeinsam mit dem aktuellen Sequenzelement an die Reduce-Funktion übergeben.



Reduce(): Schematische Darstellung der *reduce()*-Funktion (Bild 4)

Das Ergebnis ist der neue aktuelle Ergebniswert. Die Anwendungsmöglichkeiten dieser Funktion sind sehr vielfältig: Mit einem initialen Wert von *0* und Addition als Reduce-Funktion kann man die Elemente einer Sequenz aufsummieren:

```
[1,2,3,4,5].reduce((a,b)=>a+b,0)
15
```

Mit einem initialen Wert von *1* und Multiplikation als Reduce-Funktion kann man das Produkt bilden:

```
[1,2,3,4,5].reduce((a,b)=>a*b,1)
120
```

Man kann eine Sequenz aus Zeichenketten zusammenfügen, indem man als initialen Wert den Leerstring und als Reduce-Funktion eine String-Konkatenation bildet:

```
["Eins", "Zwei", "Drei"].
reduce((a,b)=>a+b, "");
"EinsZweiDrei"
```

Doch die Anwendungen sind nicht nur auf derartige mathematische Ideen beschränkt: Mit einer Sequenz aus Legobausteinen und einer Reduce-Funktion, die eine Bauvorschrift enthält, entsteht Baustein für Baustein ein Lego-Modell. Natürlich muss dabei die Sequenz aus Legobausteinen in einer für die Bauvorschrift sinnvollen Reihenfolge vorliegen.

Auch die im React-Umfeld beliebte und bekannte Bibliothek Redux basiert auf dem Konzept von *reduce*, indem ein initialer Anwendungszustand fortlaufend durch Aktionen auf neue Zustände abgebildet wird.

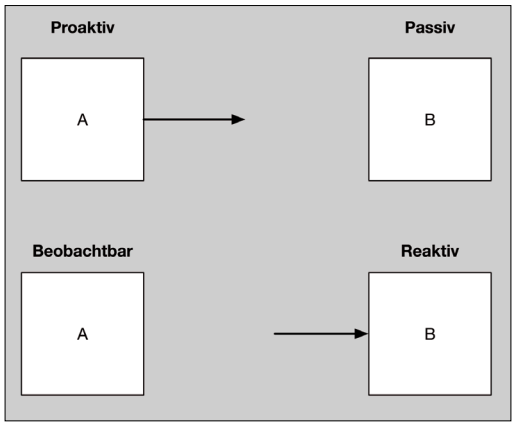
Ein Zweck bekannter Bibliotheken wie Underscore.js, Lodash oder Ramda war stets, diese grundlegenden Funktionen höherer Ordnung auch in älteren Browsern bereitzustellen.

Die JavaScript-Engines moderner Browser besitzen bereits alle Funktionen außer *zip()*. Es handelt sich um feste Bestandteile des ECMA-Script-Standards. Es ist also nicht unbedingt notwendig, solche Bibliotheken einzubinden. Diese, Art in reinem JavaScript zu programmieren, ist mittlerweile State of the Art.

Reaktive Programmierung

Das sogenannte Reaktive Manifest (www.reactivemanifesto.org/de) benennt vier Eigenschaften, die ein reaktives System ausmachen: antwortbereit, widerstandsfähig, elastisch und nachrichtenorientiert. Das dort beschriebene System ist sehr abstrakt und es fällt etwas schwer, die Verbindung zu einer konkreten Bibliothek wie RxJS zu sehen.

Wesentlich eingängiger ist dabei schon André Staltz' einfache Definition: Reaktive Programmierung sei letztlich einfach Programmieren mit asynchronen Datenströmen. Ob diese Definition auch strengere Vertreter reaktiver Programmierung überzeugt, ist nicht bekannt – aber es bringt zumindest das Wesentliche rüber.



Passiv versus reaktiv: Darstellung der Unterschiede (Bild 5)

Reaktiv versus passiv

Um den Begriff »reaktiv« besser zu verstehen, kann man ihn dem Begriff »passiv« gegenüberstellen: Bild 5 zeigt zwei Module A und B. Das Modul A verändert den Zustand des Moduls B, indem es beispielsweise eine Methode in B aufruft.

Man nennt dieses direkte Handeln des Moduls A auch proaktiv, weil A selbst entscheidet, wann B geändert wird. B ist passiv, weil es lediglich ohne eigenes Zutun modifiziert wird. Reaktiv bedeutet, dass B nicht passiv ist, sondern auf A ▶

lauscht und, sobald es ein Signal erhält, dass es sich updaten muss, sich selbst darum kümmert. Man sagt auch, dass A beobachtbar ist. Reaktiv bedeutet also, dass man auf Ereignisse reagiert, und es ist ein Programmiermodell, das sich insbesondere für asynchrone Programmierung anbietet.

Gemeinsames Konzept

Vereint man nun funktionale und reaktive Programmierung zu einem gemeinsamen Konzept »funktional-reaktive Programmierung«, so entsteht ein mächtiges Programmiermodell. Warum ist das so?

Es ist die Verbindung aus asynchroner Programmierung und der Flexibilität von funktionalen Operatoren wie *map*, *filter* und *reduce*. Dazu kann man sich überlegen, wie funktionale Operatoren eigentlich auf sehr unterschiedliche Typen angewendet werden können und wie universal sie damit einsetzbar sind.

In den einfachsten Fällen kennt man einfache Operatoren auf skalaren Typen wie Zahlen:

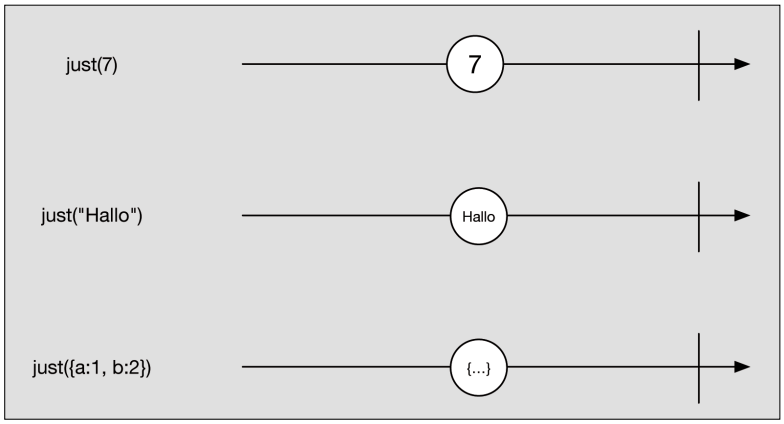
```
let seven = 3 + 4
let twelve = 3 * 4
```

Die Operatoren + (Plus für Addition) und * (für Multiplikation) werden auf Zahlen angewandt und haben immer ein Resultat. Die Rechenoperationen sind bei solchen Konstruktionen beliebig kombinierbar:

```
let thirtyfive = (3 + 4) * 5
```

Es mag ungewohnt erscheinen, dass Addition, Subtraktion und Multiplikation als funktionale Operatoren gelten, aber sie erfüllen sämtliche Vorgaben einer puren Funktion. Wenn man die gleichen Zahlen addiert, wird stets dasselbe Ergebnis dabei herauskommen.

Schon Assembler-Sprachen besitzen Operatoren, die auf einfachen Skalaren definiert sind. Der nächste Sprung sind Operatoren, die auf Sequenzen von Objekten definiert sind. Die bereits in diesem Artikel an anderer Stelle behandelten Funktionen *map*, *filter* und *reduce* sind typische Beispiele für solche Operatoren.



Datenströme mit *Observable.just()* (Bild 6)

Listing 1: Promise

```
function getUserData(name) {
  return jQuery.getJSON("/user/"+name);
}

function authenticate (user, password) {
  // Promise der User-Daten
  // string -> Promise({user, password})
  let user = getUserData(user);
  return user.then(u=>{
    return (u.password === password)
  });
}

// Ein Promise des 'authenticated' Boolean
// string -> string -> Promise(boolean)
let authenticated = authenticate("johndoe",
"secret")
authenticated.then((auth)=> {
  if (auth) {
    console.log("Authentication successful");
  } else {
    console.log("Authentication not successful");
  }
})
```

Der Hauptunterschied zu skalaren Operatoren ist, dass eben mit potenziell null oder mehreren Werten gerechnet wird.

Promises: asynchrone Werte

Der zweite Aspekt neben einzelnen Werten und Wert-Sequenzen ist deshalb das Loslösen von synchroner Verarbeitung: Alle gezeigten Operatoren berechnen einen Wert und liefern diesen als Ergebnis des Funktionsaufrufs zurück. Solange der Wert berechnet wird, blockiert der Operator. Zeit spielt keine Rolle.

In komplexen Softwaresystemen hat man es aber oft mit Asynchronität zu tun. Beispiele dafür sind Netzwerkanfragen, bei der die Latenz der Anfrage zu einer merklichen zeitlichen Verzögerung führt, oder Ereignisse von UI-Elementen, die tatsächlich zu vorher undefinierten Zeitpunkten auftreten können – eben dann, wenn der Nutzer agiert. Das asynchrone Äquivalent zum einfachen Werten ist der sogenannte Promise.

Dieser steht als Stellvertreter für einen Wert, der erst nach einiger Zeit zur Verfügung stehen wird. Es ist ein asynchroner Wert. Ein Beispiel zur Verdeutlichung zeigt Listing 1.

Da das Ergebnis der Funktion *getUserData* nicht sofort zur Verfügung steht (Serveranfrage per Ajax), gibt sie einen Promise zurück. Die Funktion *authenticate* ermittelt anhand der Nutzernamen per *getUserData* die Userdaten und

prüft dann das Passwort. Das Ergebnis dieser Prüfung wird wieder als Promise zurückgegeben. Der Code gleicht synchronem Code im Aufbau, kann jedoch mit asynchronen Wertberechnungen umgehen.

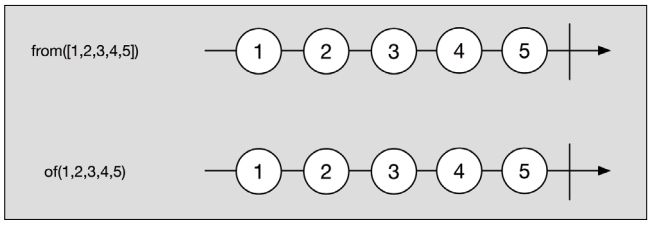
Von Promises zu Observables

Von Promises ist es nur noch ein kleiner Schritt zum Konzept der Datenströme oder Observables: Ebenso wie man von Operatoren für einzelne Werte zu Operatoren für Sequenzen abstrahieren kann, so kann man von einzelnen asynchronen Werten (Promises) zu Strömen von asynchronen Werten übergehen (Observables).

Eine *filter()*-Operation auf einem Observable erzeugt ein neues Observable, bei welchem automatisch Elemente per *filter()* herausgefiltert werden. Die Operation wird also nicht sofort auf bestehenden Daten durchgeführt, sondern erst bei Bedarf (eine Eigenschaft, die durch die Reaktivität gegeben ist). Das Observable mit den gefilterten Werten kann dann zum Beispiel mit *map()* auf ein weiteres Observable mit per Abbildungsfunktion transformierten Werten gebracht werden. Man rechnet mit Observables also nahezu genauso wie mit einfachen Arrays – aber eben sogar mit Werten aus asynchronen Quellen.

Einmal Observable, immer Observable

Einen Unterschied zu den funktionalen Sequenzoperationen gibt es jedoch: Während *map()* und *filter()* üblicherweise Sequenzen zurückgeben, liefert *reduce()* ja einen singulären Wert. Das ist bei Observables jedoch anders. Einzelne Werte



Datenströme aus Arrays und Argumenten (Bild 7)

werden bei Observable einfach als ein Observable mit einem Element gebildet.

Der Vorteil: Man kann die Operationen jederzeit auch auf einzelne Werte anwenden. Aus dieser Eigenschaft folgen zwei Dinge: Es muss Operationen geben, die beliebige JavaScript-Objekte in Observables wandeln, und es muss Möglichkeiten geben, die enthaltenen Werte aus einem Observable wieder herauszubekommen.

Universales Programmierkonzept

Mit der Möglichkeit, neben synchronen Quellen auch asynchrone Quellen durch funktionale Operatoren verarbeitbar zu machen, ist die reaktiv-funktionale Programmierung ein nahezu universales Programmierkonzept. Es spielt keine Rolle, ob man es mit Werten in einem Array, Klick-Events auf einen Button oder den Ergebnissen einer Suchanfrage zu ▶

tun hat – alles lässt sich als Observable repräsentieren und einheitlich mit den Operatoren für Observables kombinieren, konvertieren, filtern und weiterverarbeiten.

RxJS installieren

RxJS ist als NPM-Paket verfügbar und kann folgendermaßen mit dem Node Package Manager installiert werden:

```
npm install rx -save
```

Mit einem Bundler wie beispielsweise Webpack (https://webpack.github.io) oder Browserify (http://browserify.org) kann man das *npm*-Modul auch im Webbrowser einsetzen. Die RxJS-Bibliothek ist selbst minifiziert und gepackt noch ziemlich groß. Wer nicht alle Funktionen benötigt, der kann das Paket rx-lite installieren; eine reduzierte Variante der Bibliothek.

Dennoch bleibt die Größe der Bibliothek ein Kritikpunkt. Glücklicherweise ist die nächste Version (RxJS 5) von Grund auf modular konzipiert, sodass man als Entwickler gezielt jene Teile wählen kann, die man in der eigenen Anwendung wirklich benötigt.

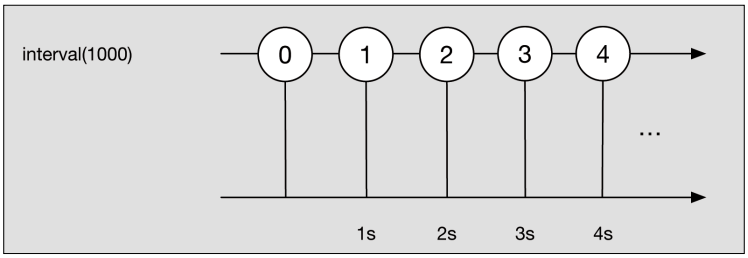
Observables erzeugen

Die Stärke der funktional-reaktiven Programmierung besteht darin, dass man den gesamten Datenfluss der eigenen Anwendung durch asynchrone Datenströme mittels Observables beschreiben kann.

Entsprechend dieser Theorie muss es viele Möglichkeiten geben, um diese Observables aus vorhandenen Objekten und Datenquellen zu erzeugen. Die einfachste Möglichkeit besteht darin, einen Datenstrom aus genau einem beliebigen Objekt mit *Rx.Observable.just* zu erzeugen:

Listing 2: browser.js

```
import * as Rx from 'rx';

let input = document.querySelector('input');

let keyUps = Rx.Observable.fromEvent(input,
"keyup");
let values = keyUps.map((ev)=>ev.target.value);
let distinct = values.distinctUntilChanged();
let debounced = distinct.debounce(500);
let inputs = debounced.startWith("");

debounced.subscribe(v=>{

  let el = document.querySelector('h1');
  el.textContent = v;

});
```



Zeitintervalle als Observable (Bild 8)

```
let eineZahl = Rx.Observable.just(7);
let einString = Rx.Observable.just("Hallo");
let einObjekt = Rx.Observable.just({ a: 1, b: 2 });
```

Bild 6 zeigt, wie diese drei Datenströme visualisiert aussehen. Die Linie ist eine Zeitleiste, auf der unterschiedliche Ereignisse eintreten können: Erscheint ein neuer Wert *(onNext-Ereignis)*, dann wird das durch einen Kreis dargestellt. Ein Fehler *(onError-Ereignis)* wird durch ein X symbolisiert. Observables können potenziell endlos sein oder eben einen Abschluss haben. Dieser wird durch ein *onComplete*-Ereignis signalisiert und im Diagramm als senkrechte Linie dargestellt.

Die Werte der drei einfachen Observables liegen – als Kreise dargestellt – auf ihrer jeweiligen Zeitleiste. Will man aus den Werten eines bestehenden JavaScript-Arrays ein Observable erzeugen, kann man die Methode *Rx.Observable.from* nutzen:

```
let array = [1,2,3,4,5,6];
let einArray = Rx.Observable.from(array);
```

Bild 7 zeigt das Ergebnis: Alle Elemente des Array sind in Reih und Glied auf der Zeitleiste aufgereiht.

Ähnlich arbeitet die Funktion *Rx.Observable.of()*, allerdings wandelt sie die einzelnen übergebenen Argumente in ein Observable:

```
let aufArgumenten = Rx.Observable.of(1,2,3,4,5,6);
```

Alle bislang gezeigten Observables bestehen aus einfachen Objekten oder Arrays. Wirklich spannend wird es mit asynchronen Datenquellen. Ein Observable kann beispielsweise aus einem definierten Zeitintervall bestehen:

```
// Rx.Observable.interval erwartet Millisekunden
// als Parameter
let jedeSekunde = Rx.Observable.interval(1000);
```

Es entsteht ein Observable, dessen erstes Element die Zahl *0* ist und das dann endlos jede Sekunde eine um eins höhere Zahl liefert (Bild 8). Doch was ist, wenn man eigentlich ein abgeschlossenes Observable benötigt? Vielleicht möchte man nach zehn Durchgängen einen Abschluss finden?

Das geht sehr einfach mit den für Observables definierten Operatoren: Mit *take()* kann man beispielsweise eine feste Anzahl an Werten aus einem Observable entnehmen. Das Er-

Listing 3: Webpack-Configuration

```
module.exports = {

  entry: {
    index: "./index",
    browser: "./browser"
  },

  output: {
    filename: "[name].bundle.js",
    chunkFilename: "[id].bundle.js"
  },

  module: {
    loaders: [
      {
        test: /\.jsx?$/,
        exclude: /(node_modules)/,
        loader: 'babel?presets[]=es2015'
      }
    ]
  }

};
```

Listing 4: package.json

```
{
  "name": "rxjs-artikel",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "bundle": "./node_modules/.bin/webpack",
    "test": "echo \"Error: no test specified\"
    && exit 1"
  },
    "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel-core": "^6.3.26",
    "babel-loader": "^6.2.0",
    "babel-preset-es2015": "^6.3.13",
    "webpack": "^1.12.9"
  },
  "dependencies": {
    "rx": "^4.0.7"
  }

}
```

gebnis von *take()* ist ein neues Observable, der seine Werte aus dem ursprünglichen bezieht und nach gegebener Anzahl Elementen ein *onComplete*-Ereignis auslöst:

```
let jedeSekunde = Rx.Observable.interval(1000);
let jedeSekunde10mal = jedeSekunde.take(10);
```

Obwohl man es hier eigentlich mit asynchron generierten Werten zu tun hat, kann man damit genauso arbeiten, als würde man einfach ein Array aus den ersten zehn Elementen eines anderen erzeugen.

Ein ebenso typischer Anwendungsfall asynchron generierter Werte sind Ereignisse wie jene aus dem DOM. RxJS bietet zwei Operatoren *fromEvent* und *fromEventPattern,* mit denen man verschiedene Ereignis-Systeme in Observables überführen kann. Sobald ein Ereignis ausgelöst wird, wird automatisch ein *onNext*-Ereignis auf dem erzeugten Observable generiert. So lassen sich Ereignisse neben beliebigen anderen Observables verwenden und kombinieren.

DOM-Ereignisse verarbeiten

Das Beispiel in Listing 2 erzeugt ein Observable aus Tastatur-Ereignissen eines Textfelds und stellt die jeweils aktuellen Eingabewerte in einem *h1*-Tag dar:

Das Observable *keyUps* wird mit *Rx.Observable.fromEvent()* erzeugt und enthält sämtliche *input*-Ereignisse des Textfelds. Für den Textinhalt des *h1*-Tag benötigt man jedoch nicht das Ereignis, sondern den Inhalt des Textfelds. Das *input*-Element findet man bei einem DOM-Ereignis unter dem Attribut *target*. Damit kann man den aktuellen Eingabewert im Attribut *value* ermitteln. Das Observable *values* basiert auf *keyUps*. Es ist einfach eine Abbildung per *map()* von den Ereignissen zu den jeweils aktuellen Inhalten des Textfelds.

Dubletten vermeiden mit distinctUntilChanged()

Nach der Abbildung auf die Inhalte des Textfelds kann es passieren, dass aus zwei aufeinanderfolgenden Ereignissen dennoch der gleiche Textinhalt vorliegt. Derartige Dubletten kann man ebenfalls mit einem Operator entfernen: Dazu nutzt man bei RxJS den Operator *distinctUntilChanged()*.

Derartige Dubletten erzeugen unnötige DOM-Manipulationen, was in diesem kleinen Beispiel allerdings kaum visuell wahrnehmbar ist. Nutzt man die Werte des Textfelds jedoch für Suchanfragen, merkt man schnell, dass auch Tastaturereignisse, die am eigentlichen Inhalt nichts ändern, zu einem Ereignis im Observable führen. Das Ergebnis wären doppelte Suchen nach dem gleichen Text und damit unnötige Netzwerkanfragen.

Doch selbst wenn man Dubletten verhindert – beim schnellen Tippen kommt es trotzdem oft zu unnötig vielen Ereignissen. Die typische Lösung: ein Debouncing der Ereignisse, um zu schnell aufeinanderfolgende Ereignisse zu ignorieren. Mit *Rx.Observable.debounce()* kann man dies für beliebige Observable-Sequenzen erreichen.

Bevor der Nutzer irgendetwas eingegeben hat, gibt es noch kein Element im Observable. Oft möchte man jedoch einen initialen Zustand bereitstellen, der für die erste Darstellung genutzt werden soll. Dies ist mit der Operation *startWith()* ▶

möglich, welche ein gegebenes Objekt vorne an ein Observable anfügt.

Um nun den Textinhalt des *h1*-Elements mit der Observable-Sequenz *inputs* zu verknüpfen, kann man sich mittels *subscribe()* daran registrieren. Im Beispiel wird dann einfach das *h1*-Element gesucht und der Textinhalt auf das an die Callback-Funktion übergebene Element der Observable-Sequenz angepasst.

Das Bundle bauen

Die Webpack-Configuration in Listing 3 übersetzt die *browser.js* mittels des Babel-Loaders von ES2015 nach ES5 und erzeugt ein Bundle *browser-bundle.js* mit allen notwendigen Abhängigkeiten.

Die Abhängigkeiten werden mit NPM verwaltet und sind in *package.json* (Listing 4) beschrieben.

Mit *npm install* werden alle notwendigen Abhängigkeiten installiert. Anschließend kann das Bundle *browser-bundle.js* mit folgendem Aufruf gebaut werden:

```
npm run bundle
```

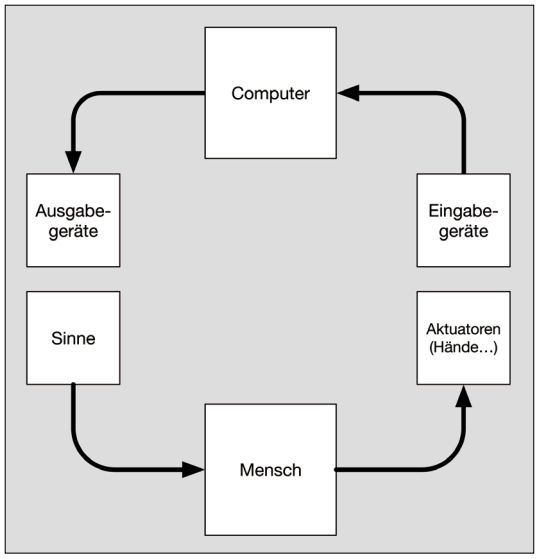
Das zugehörige NPM-Skript startet das unter *node_modules* installierte *webpack*-Kommando und baut so das Bundle. In der HTML-Datei wird dann lediglich das Bundle als Skript eingebunden (Listing 5).

Das Beispiel funktioniert, aber die direkte DOM-Manipulation mit dem leider imperativen DOM-API machen einen Großteil der konzeptionellen Schönheit eines solchen funktionalen Ansatzes wieder zunichte.

Nicht umsonst sehen beispielsweise die React-Entwickler den größten Vorteil ihres Frameworks nicht im virtuellen DOM mit dessen DOM-Diff-Algorithmus, sondern in der Bereitstellung eines funktionalen DOM-API.

Noch besser wäre es jedoch, wenn die Browser selbst einfach bereits ein solches funktionales DOM-API anbieten würden. React und RxJS passen aus diesen Gründen tatsächlich ausgezeichnet zusammen.

Doch auch das kleine JavaScript-Framework Cycle.js von André Staltz vervollständigt RxJS zu einer runden Sache: Es basiert grundlegend auf RxJS und löst das Problem mit der DOM-Manipulation auf seine eigene Weise.



Cycle.js: Mensch-Maschine-Modell (Bild 9)

Modellvorstellung

Der Name Cycle.js deutet bereits an, dass das Framework in irgendeiner Weise einen Kreislauf wiedergibt. Bild 9 zeigt, wie in der Modellvorstellung von Cycle.js die Verbindung von Mensch und Maschine funktionieren soll.

Der Computer hat demnach Eingaben wie Maus- oder Tastenbefehle, die letztlich als Aktionen vom Menschen ausgegeben werden. Aus diesen Eingaben berechnet der Computer Ausgaben, die der Mensch durch seine Sinne als Eingaben wahrnimmt. So entsteht ein Kreislauf.

Jede Komponente mit Eingaben und Ausgaben kann nach dieser Vorstellung als Funktion dargestellt werden:

```
function computer (aktionen) {
  return inhalte;
}

function mensch (sinne) {
  return aktionen;
}
```

André Staltz hatte die Idee, die Eingaben und Ausgaben dieser funktionalen Abbildung der Mensch-Maschine-Schnittstelle mittels Observables zu modellieren. Das vorherige Code-Beispiel unter Anwendung von Cycle.js sieht aus wie in Listing 6 dargestellt.

Konventionen

Anmerkung: Die Cycle.js-Codebeispiele folgen einer in diesem Framework gängigen Konvention, dass alle Variablen, die Observables enthalten, auf *$* enden.

Das Beispiel zeigt: Eine Cycle.js-App ist einfach eine Funktion. Der Parameter *sources* enthält ein Objekt, dessen Attribute aus verschiedenen sogenannten Treibern besteht.

Der einzige im Beispiel vorhandene ist der sogenannte *DOMDriver,* der mittels der Operation *makeDOMDriver("#viewport")* erstellt wurde. Dieser Treiber schafft die Verbin-

Listing 5: browser.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>RxJS Demo</title>
  </head>
  <body>
  <div id="viewport">
    <input type="text">
    <h1></h1>
  </div>
  <script src="browser.bundle.js"></script>
  </body>
</html>
```

dung zu dem Teil-DOM unter dem Element, das per Selektor *#viewport* gefunden werden kann. Der DOM-Treiber bietet dazu eine Funktion *select*, mit der man Elemente im DOM finden kann. Mit *events()* kann man eine Observable-Sequenz der DOM-Ereignisse des Elements erzeugen. Diese wird im Beispiel genauso weiterverarbeitet wie im vorherigen Code.

Der Unterschied besteht jedoch dann in der Manipulation des DOM: Cycle bietet dazu eine Bibliothek @cycle/dom an – eine eigene Virtual-DOM-Implementierung. Das Observable *input$* enthält stets die aktuellen Eingaben im Textfeld und wird mit *map* einfach nur auf einen jeweils zur Eingabe passenden Virtual-DOM-Tree abgebildet. Das Observable dieser *vdom*-Trees wird aus der *main()*-Funktion zurückgegeben. Der Kreis schließt sich wieder.

Debugging erschwert

Programme können mit einer Bibliothek wie RxJS und einem funktionalen Programmierstil durch die deklarative Programmierung leichter nachvollziehbar und und weniger anfällig für Fehler werden. Doch einen Haken hat das Ganze: Die Debugger-Tools der Webbrowser sind letztlich auf imperativen Quellcode ausgelegt.

Breakpoints, die auf einzelnen Zeilen stoppen, nützen wenig, da der Ablauf des Codes nicht mehr Zeile für Zeile nachvollziehbar ist, sondern durch lediglich bei Bedarf ausgewertete Datenströme. Die kommende Version 5 von RxJS soll einige Verbesserungen enthalten, die das Debugging dennoch etwas erleichtern sollen.

Listing 6: Beispiel mit Cycle.js

```
import Cycle from '@cycle/core';
import {div, label, input, h1, makeDOMDriver} from
"@cycle/dom";

function main (sources) {
  let input$ = sources.DOM.select
  (".field").events("input")
  .map((ev)=>ev.target.value)
  .distinctUntilChanged()
  .debounce(500)
  .startWith("");
  let vtree$ = input$.map(text=>
    div([
      input(".field", { attributes:
      {type: 'text'}}),
      h1(text)
    ])
  );
  return {
    DOM: vtree$
  }
}

Cycle.run(main, {DOM: makeDOMDriver("#viewport")});
```

Links zum Thema

- RxJS-Website
  https://github.com/ReactiveX/RxJS
- Interaktive Visualisierung der RxJS-Operatoren
  http://rxmarbles.com
- Cycle.js-Website
  http://cycle.js.org
- Webpack
  https://webpack.github.io
- Browserify
  http://browserify.org

Fazit

Mit Observable-Sequenzen aus RxJS ist es möglich, gesamte JavaScript-Programme mittels asynchroner Datenströme zu beschreiben. Das Prinzip ist mächtig und es vermeidet ganze Klassen von Fehlersituationen. Die Vorteile funktionaler Datenverarbeitung können so auch auf reaktive Systeme angewandt werden. Bedienoberflächen lassen sich mit denselben Mitteln programmieren.

Dass Observable-Sequenzen zumindest als Vorschlag für einen kommenden ECMAScript-Standard eingereicht wurden, kann bedeuten, dass man in Zukunft nicht nur ohne RxJS auf diese Weise programmieren kann, sondern auch, dass zukünftige Entwicklerwerkzeuge passende Werkzeuge bereitstellen, um das Arbeiten mit solchen asynchronen Datenströmen zu erleichtern.

Leider hat es der Vorschlag für Observables noch nicht in die für den ES2016 notwendige Stufe 4 geschafft, weshalb dieses Feature nicht Teil von ES2016 sein wird. Allerdings gilt ECMAScript neuerdings offiziell als lebender Standard – das bedeutet, dass Features automatisch als Teil des kommenden Standards gelten, sobald sie Stufe 4 erreichen. Man kann sich also in Zukunft auch schon vor den Jahres-Releases darauf verlassen, was kommt und was noch nicht.

Nichtsdestotrotz erfreuen sich asynchrone Datenströme nicht nur in Form von RxJS steigender Beliebtheit. Auch andere Bibliotheken wie Bacon.js, Kefir oder Most.js bietet ähnliche Features. Kein Wunder also, dass schon manchmal davon gesprochen wird, dass 2016 das Jahr der asynchronen Datenströme wird. ■

Jochen H. Schmidt
ist als Autor, Berater und Software-Entwickler tätig. Schwerpunkte seiner Aktivitäten sind Webentwicklung und Webtechnologien. Er ist Verfasser von bekannten Fachbüchern zum Thema Common Lisp.

ÜBERBLICK

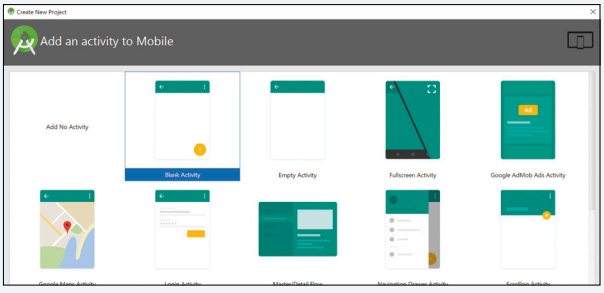
DVD-Highlights

Das PDF-Jahresarchiv 2015 auf der Heft-DVD umfasst 1.776 Seiten im Original-Layout.

Jahresarchive 2013 bis 2015 (5.335 Seiten)

Direkt von der Heft-DVD öffnen Sie die Jahresarchive der letzten drei Jahrgänge der **web & mobile Developer** als PDF-Archiv. Oder Sie kopieren den Ordner *archive* im Verzeichnis *files* auf Festplatte und legen eine Verknüpfung der jeweiligen Datei *start.pdf* in den Jahrgangsordnern auf den Desktop. So haben Sie insgesamt 5.335 Seiten im Original-Layout immer schnell zur Hand. Sie können im Adobe Reader oder einem anderen PDF-Betrachter durch die Seiten blättern und Artikel direkt über das Inhaltsverzeichnis der Ausgaben öffnen. Mit der Tastenkombination [Umschalt Strg F] öffnen Sie die erweiterte Indexsuche im Adobe Reader für schnelle Treffer.

Android Studio Bundle 1.5.1



Android Studio ist eine freie Integrierte Entwicklungsumgebung von Google für Android auf Basis von IntelliJ IDEA. Zudem gibt es Unterstützung für die Entwicklung von Android-Wearables und TV-Apps. Android Studio verwendet ein Build-Management-Automatisierungs-Tool und erlaubt die Ausgabe optimierter Apps für verschiedene Gerätetypen wie Tablets und Smartphones. Android Studio überprüft den Quelltext automatisch und liefert eine Liste von Verbesserungsmöglichkeiten.

Weitere Highlights:

BlueStacks App Player 2.0.8

Mit dem Android-Emulator für den PC laden Sie Android-Apps und -Spiele herunter, installieren sie und nutzen sie dann auf dem Windows-Desktop. Zur Installation von Apps aus dem Play Store wird ein Google-Account benötigt. Zusätzlich lassen sich mit dem BlueStacks App Player auch Apps über ihre APK-Dateien installieren. In den Einstellungen der Software kann man einige bekannte Android-Einstellungen anpassen, Apps verwalten und etwa die Sprachversion einstellen.

XAMPP 7.0.2.1 für Windows und Linux

So mancher wird schon die Erfahrung gemacht haben: Ein Apache-Webserver installiert sich nicht so leicht, Systemkenntnisse zum Einspielen der Pakete sind erforderlich. Noch schwieriger wird es, wenn weitere Pakete wie MySQL, PHP oder Perl dazukommen. XAMPP ist eine Distribution, die es ermöglicht, diese Programme auf sehr einfache Weise zu installieren.

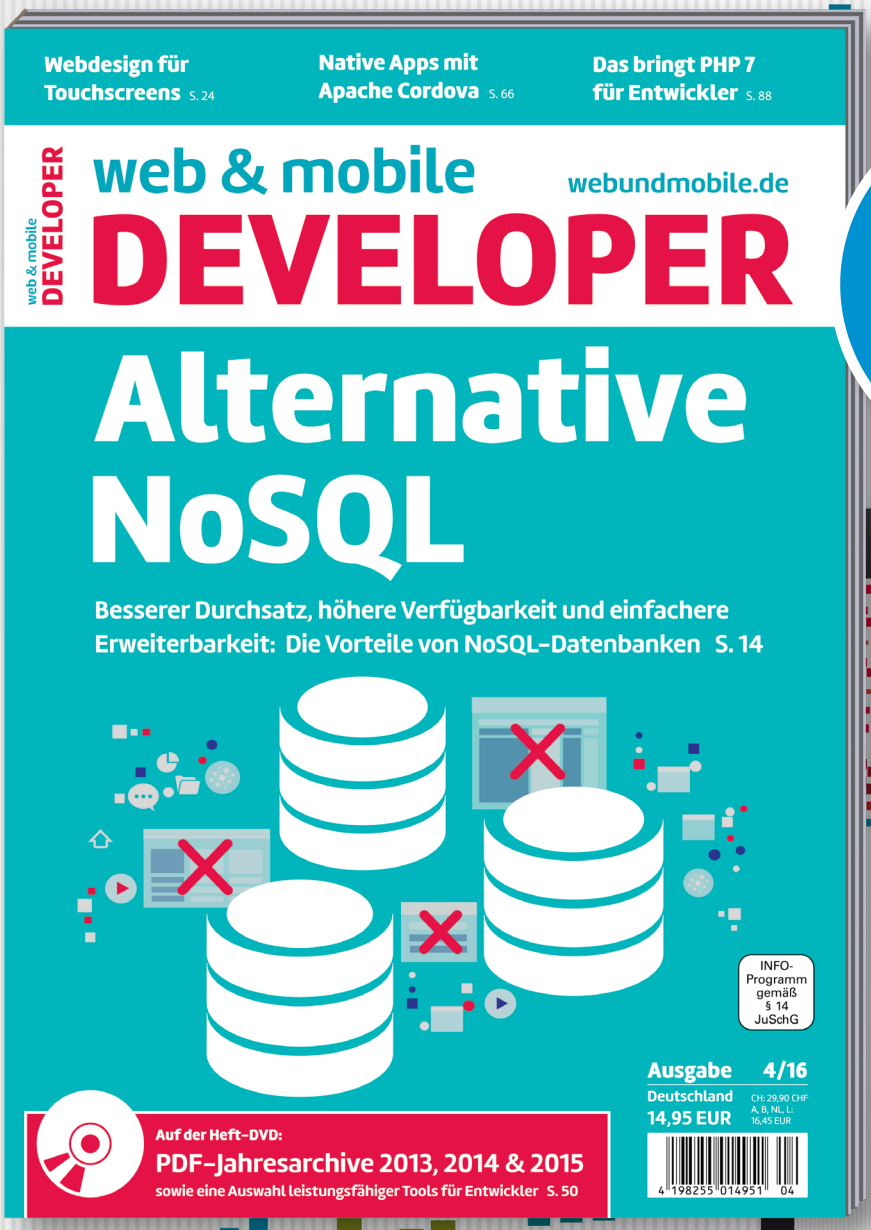
NetBeans IDE 8.1

Die Entwicklungsumgebung wurde hauptsächlich für die Programmiersprache Java entwickelt und unterstützt unter anderem C, C++ und dynamische Programmiersprachen. Die IDE setzt ein Grundverständnis von den verwendeten Programmiersprachen und Bibliotheken oder Frameworks voraus. Das Paket enthält NetBeans Platform SDK, Java SE, JavaFX, Java Web und EE, Java ME, Java Card 3 Connected, Ruby, C/C++, Groovy und PHP. Als Server werden GlassFish Server Open Source Edition und Apache Tomcat mitgeliefert.

Oracle VirtualBox 5.0.14

Mit der kostenlosen Virtualisierungs-Software können Sie auf einem Windows-Host-PC etwa Windows- und Linux-Systeme als Gast-Computer installieren. VirtualBox emuliert einen Rechner mit Hardware-Komponenten wie Prozessor, Arbeitsspeicher, Festplatten, CD/DVD-Laufwerken, Netzwerkkarte und externen USB-Geräten.

Jetzt kostenlos testen!



2 x gratis!

Praxiswissen für Entwickler!

Testen Sie jetzt 2 kostenlose Ausgaben und erhalten Sie exklusiven Zugang zu unserem Archiv.

webundmobile.de/probelesen

DELEGATION IN SWIFT

Aufgaben auslagern

Mittels Delegation lassen sich Aufgaben eines Typs an ein anderes Objekt auslagern.

Delegation ist eines der wichtigsten Patterns in der iOS-Entwicklung. Es kommt in zahlreichen Klassen aus Apples Foundation- und UIKit-Framework zum Einsatz und trägt einen wertvollen Teil dazu bei, Code übersichtlich und wiederverwendbar zu gestalten (Bild 1).

Doch was ist Delegation eigentlich genau? Und wie wird Delegation in der iOS-Entwicklung mit Swift korrekt umgesetzt und implementiert?

Delegation: Ein Anderer macht es

Klären wir zunächst einmal die Frage, was Delegation in der iOS-Entwicklung eigentlich ist. Ganz grundlegend deutet der Begriff selbst bereits an, worum es geht: Wie im richtigen Leben auch, bezeichnet Delegation die Auslagerung von Aufgaben an eine andere Stelle. Sprechen wir von der Delegation in der iOS-Entwicklung, so bedeutet das, dass ein Objekt einzelne Aufgaben an ein anderes Objekt auslagert und dieses andere Objekt sich um die Durchführung und Abarbeitung eben jener Aufgaben kümmert.

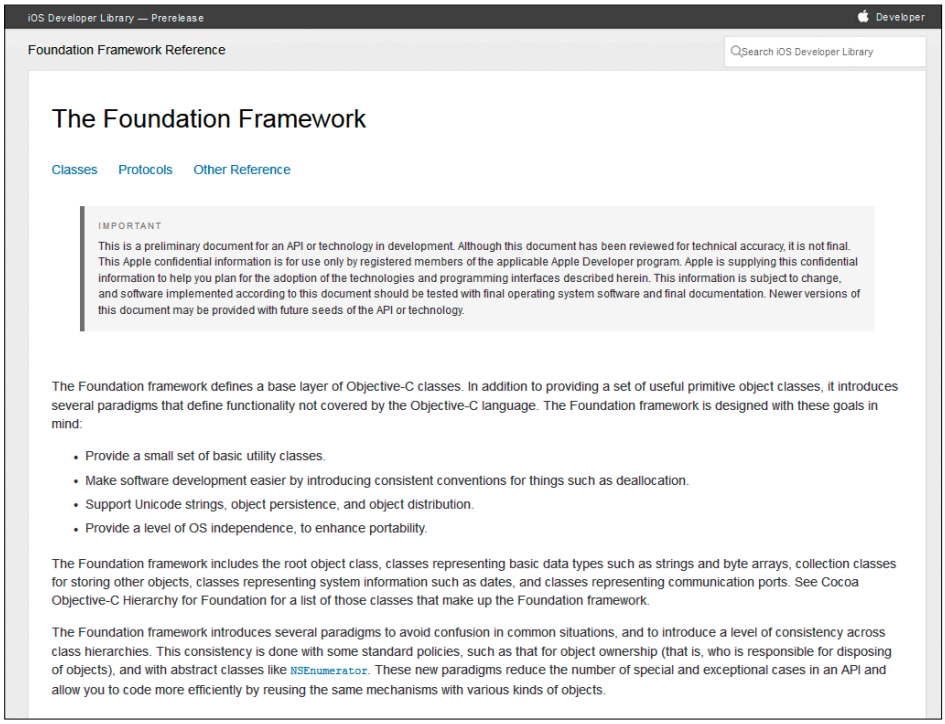
Verwendet wird Delegation vor allen Dingen im Zusammenspiel zwischen View und Controller, zwei der Elemente aus dem Model-View-Controller-Pattern (kurz MVC) (Bild 2).

Die Klasse *UITableView* dürfte dabei unter iOS-Entwicklern zu den bekanntesten Vertretern von Delegation gehören (Bild 3). So definiert eine Table-View selbst nicht, wie viele Zellen sie anzeigt oder wie diese Zellen aussehen. Auch legt sie nicht fest, was geschehen soll, wenn eine Zelle ausgewählt wird.

Diese Aufgaben delegiert die Table-View an ein anderes Objekt. Und das ist auch gut so. Somit kümmert sich die Table-View nur darum, wie sie sich selbst aufbaut und wie sie aussieht. Die Informationen darüber, was sie anzeigt und was sie bei Aktionen wie der Auswahl einer Zelle tut, überlässt sie anderen Objekten, meist Objekten der Klasse *UITableViewController*.

Hier offenbart sich auch direkt einer der größten Vorteile von Delegation: Möchte man beispielsweise das Verhalten einer Table-View verändern, so muss man nicht eine komplett neue Table-View erstellen; stattdessen tauscht man einfach das Delegate-Objekt aus und definiert dort eben die passende Implementierung genau jener Aufgaben, die die Table-View delegiert.

So weit die Theorie, die womöglich zum jetzigen Zeitpunkt noch ein wenig kryptisch klingen mag. Keine Sorge, der Schleier lüftet sich bald. Werfen wir daher im nächsten Schritt einmal einen Blick auf die praktische Umsetzung von Delegation in der iOS-Entwicklung.



Delegation kommt in zahlreichen Klassen aus Apples Foundation- und UIKit-Framework zum Einsatz (Bild 1)

Basis von Delegation: Protokolle

Grundlage der Delegation in iOS sind die sogenannten Protokolle. Protokolle definieren Eigenschaften wie Properties und Methoden, implementieren diese aber nicht. Sie enthalten somit nur die reine Deklaration ohne jegliche Logik. Womöglich fragt sich der ein oder andere, wozu das gut sein soll, doch dazu kommen wir gleich; in der Delegation ist eben genau dieses Verhalten essenziell wichtig.

Um ein Protokoll in Swift zu erstellen, wird dieses mit Hilfe des Schlüsselworts *protocol,* gefolgt vom gewünschten Namen des Protokolls deklariert. Anschlie-

ßend folgt innerhalb von geschweiften Klammern die Deklaration der Eigenschaften dieses Protokolls, ganz ähnlich, wie man es auch von Klassen in Swift kennt.

Nur eben die Implementierung bleibt in Protokollen außen vor. Ein einfaches Beispiel für die Umsetzung eines Protokolls in Swift sieht so aus:

```
protocol Drivable {
  var speed: Int {
    get set
    }
  func startDriving()
  func stopDriving()
}
```

In diesem Protokoll werden insgesamt drei Eigenschaften festgelegt: zwei Methoden und eine Property. Dabei wird insbesondere bei den Methoden deutlich, dass diese über keinerlei Implementierung verfügen, sondern stattdessen nur innerhalb des Protokolls deklariert werden.

Bei der Property *speed* des Protokolls ist das nicht anders, hier sticht jedoch eine Besonderheit ins Auge: Nach Deklaration der Property und Angabe des Typs *(Int)* folgt innerhalb von geschweiften Klammern die Info, dass diese Property über einen Getter und einen Setter verfügt *({ get set })*. Das bedeutet, dass die Property *speed* sowohl gelesen als auch geschrieben werden kann.

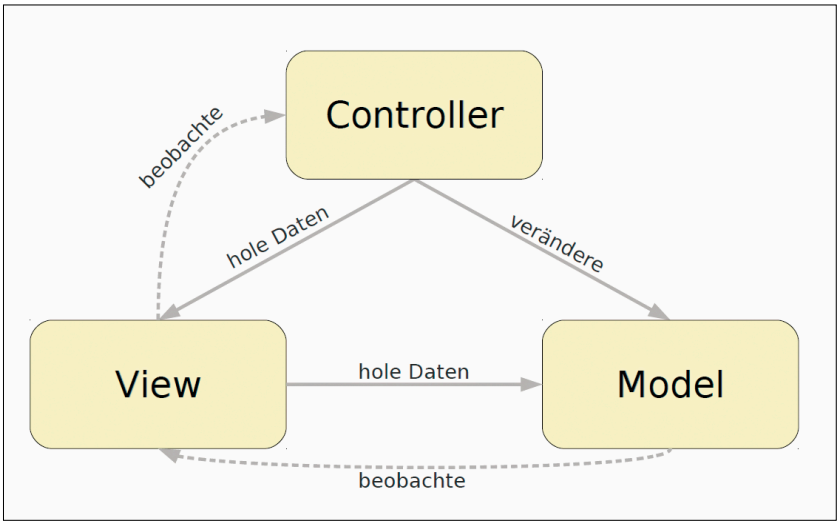
Alternativ gibt es in Protokollen die Möglichkeit, Properties als reine Getter zu deklarieren. Dann müssen diese Properties immer noch einen Wert zurückliefen, es kann ihnen aber kein neuer Wert zugewiesen werden.

Eine solche Property wird mit Hilfe des alternativen Codes *{ get }* anstelle von *{ get set }* in einem Protokoll deklariert. Ein Setter alleine ist nicht möglich und auch wenig sinnvoll, da damit der gesetzte Wert einer Property niemals ausgelesen und verwendet werden könnte.

Eigene Typen

So weit die Deklaration des Protokolls mit dem Namen *Drivable*. Protokolle generieren in Swift – genau wie Klassen – einen eigenen Typ. So, wie auch *String*, *Array* oder *Dictionary* Typen in Swift sind, sorgt die Deklaration eines Protokolls ebenfalls für die Generierung eines entsprechenden Typs.

Somit kann in diesem Beispiel nun *Drivable* als Typ genauso wie jeder andere verfügbare Typ auch verwendet werden. Was das im Detail bedeutet und wofür das gut ist, sehen wir gleich. Zunächst betrachten wir aber einmal das Erstellen einer Klasse, die zu einem Protokoll konform ist.



MVC: Delegation wird vor allen Dingen im Zusammenspiel zwischen View und Controller verwendet (Bild 2)

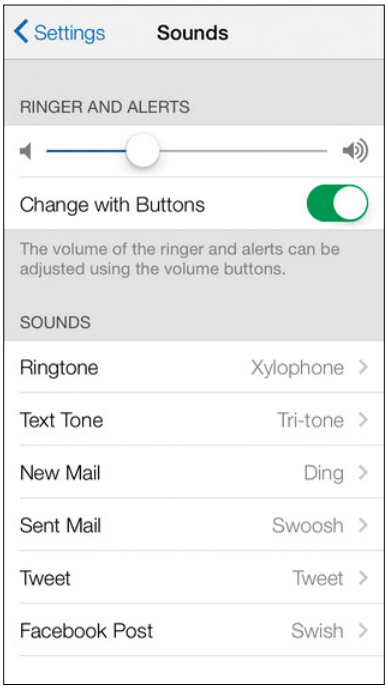
Protokollkonforme Klassen

Wir haben bereits gesehen, dass Protokolle lediglich Deklarationen vorgeben und keine Implementierung zur Verfügung stellen. Sie dienen in gewisser Weise als Blaupausen, die definieren, welche Eigenschaften bestimmten Objekten zur Verfügung stehen. Soll somit eine Klasse die Eigenschaften eines Protokolls implementieren, so spricht man auch davon, dass die entsprechende Klasse konform zum jeweiligen Protokoll sein soll.

Konform bedeutet in diesem Fall, dass die entsprechende Klasse versichert, für alle Eigenschaften des Protokolls eine Implementierung anzubieten. Das folgende Beispiel zeigt dazu einmal die Deklaration und Implementierung einer Klasse *Car*, die konform zum im vorigen Beispiel erzeugten Protokoll *Drivable* ist und für alle Eigenschaften dieses Protokolls eine eigene Implementierung anbietet:

```
class Car: Drivable {
  var manufacturer: String?
  var speed: Int = 0
  func startDriving() {
    print("Car starts driving...")
  }
  func stopDriving() {
    print("Car stops driving...")
  }
}
```

Die Klasse *Car* implementiert alle Eigenschaften, die im Protokoll *Drivable* de- ▶



Die Klasse UITableView gehört bei iOS-Entwicklern zu den bekanntesten Vertretern von Delegation (Bild 3)

finiert sind; die Property *speed* (welche sie mit einem Standardwert von *0* implementiert) sowie die beiden Methoden *startDriving* und *stopDriving*.

Zusätzlich verfügt die Klasse noch über eine eigene Eigenschaft, eine optionale Property *manufacturer*. Doch essenziell ist die Implementierung aller Eigenschaften des *Drivable*-Protokolls. Das liegt daran, dass die Klasse *Car* in ihrer Deklaration festlegt, konform zum Protokoll *Drivable* zu sein:

```
class Car: Drivable
```

Nach dem Namen der Klasse folgt ein Doppelpunkt gefolgt vom Namen des Protokolls. Diese Syntax ist identisch wie die zum Zuweisen einer Superklasse, von der eine Klasse in Swift all ihre Eigenschaften erbt. Und da es sich – wie wir bereits erfahren haben – bei einem Protokoll auch um einen Typ in Swift handelt, lässt sich dieser Typ genauso zuweisen und verwenden wie eben jeder andere Typ auch.

Diese Deklaration der Klasse *Car* verpflichtet die Klasse dazu, alle Eigenschaften des Protokolls *Drivable* zu implementieren. Fehlt auch nur eine der Eigenschaften des Protokolls, wird ein Compiler-Fehler in Xcode gefeuert.

Darüber hinaus zeigt das Beispiel aber auch, dass Klassen, die zu einem Protokoll konform sind, trotzdem natürlich auch noch andere Eigenschaften implementieren können und dürfen (im Beispiel der Klasse *Car* ist das die Property *manufacturer)*. Doch die Klasse muss in jedem Fall mindestens die Eigenschaften implementieren, die in dem Protokoll definiert sind, zu dem sie konform ist.

Sinn und Zweck von Protokollen

Mit diesem ersten Beispiel möchte ich einmal näher auf den Sinn und Zweck von Protokollen eingehen und erklären, wozu sie gut sind und wofür man sie einsetzt.

Zunächst einmal sind Protokolle immer dann sinnvoll, wenn man für bestimmte Klassen gewisse Eigenschaften voraussetzt und erwartet. In solch einem Fall kann es sinnvoll sein, besagte Eigenschaften in einem Protokoll zu deklarieren und alle Klassen konform zu eben diesem Protokoll sein zu lassen.

Damit stellt erstens der Compiler sicher, dass auch tatsächlich alle gewünschten Eigenschaften der Klasse implementiert werden. Darüber hinaus sorgt diese Maßnahme aber auch für eine bessere Strukturierung und Dokumentation des eigenen Codes. Anstatt mehrere verschiedene Klassen zu haben, die mehr oder weniger zufällig ganz oder in Teilen dieselben Eigenschaften besitzen, macht ein zugrunde liegendes Protokoll deutlich, warum sich einzelne Eigenschaften immer und immer wieder in verschiedenen Klassen finden und welchem Zweck sie dienen.

In diesem ersten Fall haben Protokolle somit keine technische Notwendigkeit; sie machen den eigenen Code einfach übersichtlicher und strukturierter. Doch dann gibt es noch einen weiteren Bereich, in dem Protokolle mehr sind als nur ergänzendes und schmückendes Beiwerk. Wie eingangs bereits geschrieben, sind bei der Delegation Protokolle das A und O.

Listing 1: Implementierung der Klasse Truck

```
class Truck {
  var speed: Int = 0
  var weight: Double?
  func startDriving() {
    print("Truck starts driving...")
  }
  func stopDriving() {
    print("Truck stops driving")
  }
}
```

Delegation in der Praxis

Um den Sinn und Zweck von Delegation sowie Protokollen vollends zu verstehen, erweitere ich das bisherige Beispiel bestehend aus *Drivable*-Protokoll und *Car*-Klasse um eine weitere Klasse namens *Driver*:

```
class Driver {
  var name: String
  var vehicle: Car?
  init(name: String) {
    self.name = name
  }
}
```

Die Klasse *Driver* verfügt über zwei Properties, *name* (für den Namen eines Fahrers) und *vehicle*. Letzteres stellt das Fahrzeug dar, mit dem ein Fahrer unterwegs ist. Aktuell ist *vehicle* vom Typ *Car* der gleichnamigen Klasse, was bedeutet, dass wir dieser Property ausschließlich Instanzen vom Typ *Car* zuweisen können.

Das folgende Beispiel zeigt einmal die Erstellung einer neuen *Driver*- und *Car*-Instanz sowie die Zuweisung des *Car*-Objekts zum erzeugten Fahrer:

```
let myCar = Car()
let myDriver = Driver(name: "Thomas")
myDriver.vehicle = myCar
```

Anhand dieser Basis sind wir nun in der Lage, auf die *vehicle*-Property von *myDriver* die Methoden *startDriving* und *stopDriving* aufzurufen sowie den Wert der Property *speed* zu verändern:

```
myDriver.vehicle?.startDriving()
myDriver.vehicle?.speed = 50

myDriver.vehicle?.stopDriving()
myDriver.vehicle?.speed = 0
```

So weit, so gut. Die Klasse *Car* definiert all diese Eigenschaften und somit können wir natürlich über die Property *vehicle* unserer *Driver*-Instanz auf diese zugreifen.

Doch unser Fahrer kann natürlich nicht nur Autos fahren. Listing 1 implementiert eine weitere Klasse namens *Truck*, mit der Lkw-Objekte abgebildet werden sollen.

Die Klasse *Truck* enthält zwei Properties, *speed* und *weight*, um die Geschwindigkeit und das Gewicht eines Lkw abzubilden. Darüber hinaus implementiert es eine *startDriving*- und *stopDriving*-Methode, mit der der Lkw gestartet und angehalten werden soll.

Zu großen Teilen kommen uns die Eigenschaften der Klasse *Truck* vertraut vor. Abgesehen von der Property *weight* sind alle Eigenschaften auch Teil des *Drivable*-Protokolls. Zwar ist die Klasse *Truck* nicht konform zu diesem Protokoll, stellt die entsprechenden Eigenschaften aber bereit.

Dennoch kann aber die Klasse *Truck* nicht als Fahrzeug für unseren Fahrer verwendet werden. Schließlich ist die entsprechende Property *vehicle* vom Typ *Car*, womit ihr auch nur ausschließlich Objekte der Klasse *Car* zugewiesen werden können. Obwohl *Truck* über dieselben Eigenschaften verfügt, die wir auch auf *Car* aufrufen, ist es uns nicht möglich, unserem Fahrer einen Lkw als Fahrzeug zuzuweisen. Und genau an dieser Stelle greift nun das Prinzip der Delegation.

Dynamische Objekte auf Basis von Protokollen

Das von uns definierte Protokoll *Drivable* legt nun ja alle Eigenschaften fest, über die ein Fahrzeug verfügen muss, wenn unser Fahrer damit fahren soll. Und da wir inzwischen ebenso wissen, das Protokolle eigene Typen – genau wie Klassen – in Swift definieren, können wir die Property *vehicle* unserer *Driver*-Klasse statt des statischen Typs *Car* den Typ des Protokolls zuweisen. Der folgende Code zeigt die überarbeitete Implementierung der Klasse *Driver*:

```
class Driver {
  var name: String
  var vehicle: Drivable?
  init(name: String) {
    self.name = name
  }
}
```

Tatsächlich sorgt diese kleine Änderung zunächst einmal für keinerlei verändertes Verhalten unserer gesamten bisherigen Implementierung. Weiter oben haben wir der Property *vehicle* einer *Driver*-Instanz ein *Car*-Objekt zugewiesen, und diese Zuweisung ist auch nach dieser Änderung noch immer korrekt. Das liegt daran, dass die Klasse *Car* konform zum Protokoll *Drivable* ist.

Die Änderung des Typs der Property *vehicle* von *Car* zu *Drivable* hat lediglich bewirkt, das wir *vehicle* nun nur solche Objekte zuweisen können, die eben konform zum Protokoll *Drivable* sind; genau wie dies bei der Klasse *Car* der Fall ist.

Kommen wir nun noch einmal zurück auf unsere Klasse *Truck*. Da diese auch von einem Fahrer genutzt werden soll und alle notwendigen Eigenschaften implementiert, kann und soll sie auch entsprechend konform zum *Drivable*-Protokoll sein. Listing 2 zeigt die zugehörige überarbeitete Version der Implementierung der Klasse *Truck*.

Es bleibt alles gleich, bis auf die Tatsache, dass nun auch *Truck* konform zu *Drivable* ist. Da die Klasse sowieso bereits alle notwendigen Eigenschaften dieses Protokolls implementiert, braucht sonst nichts weiter getan zu werden.

Listing 2: Zuweisen des Drivable-Protokolls

```
class Truck: Drivable {
  var speed: Int = 0
  var weight: Double?
  func startDriving() {
    print("Truck starts driving...")
  }
  func stopDriving() {
    print("Truck stops driving")
  }
}
```

Listing 3: Aufruf von Eigenschaften eines Protokolls

```
myDriver.vehicle = myCar
myDriver.vehicle?.speed = 50
myDriver.vehicle?.startDriving()
myDriver.vehicle?.stopDriving()

myDriver.vehicle = myTruck
myDriver.vehicle?.speed = 30
myDriver.vehicle?.startDriving()
myDriver.vehicle?.stopDriving()
```

Änderung mit großer Wirkung

Doch diese kleine Änderung hat erneut eine große Wirkung. Denn nun können wir ohne Probleme der Property *vehicle* unserer *Driver*-Klasse auch ein *Truck*-Objekt zuweisen. Die Initialisierung und Zuweisung eines solchen *Truck*-Objekts erfolgt so:

```
let myTruck = Truck()
myDriver.vehicle = myTruck
```

Solange die Property *vehicle* vom Typ *Car* war, wäre eine solche Zuweisung unmöglich gewesen; schließlich ist *Truck* ein anderer Typ als *Car*.

Durch das Protokoll *Drivable* wurden nun aber gemeinsame Eigenschaften definiert, die jede der Klassen auf beliebige Art und Weise implementieren kann, solange sie sie nur vollständig implementiert.

Objekte aller Klassen, die konform zum Protokoll *Drivable* sind, können somit der *vehicle*-Property zugewiesen werden. Kommen in unserem Beispielprojekt somit nach und nach möglicherweise noch weitere Fahrzeugklassen hinzu, so können auch diese als Fahrzeug verwendet werden, solange sie konform zum Protokoll *Drivable* sind. ▶

Links zum Thema

- Apple-Developer-Portal
  https://developer.apple.com
- Swift
  https://developer.apple.com/swift
- Model-View-Controller (MVC)
  https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html
- Übersicht iOS
  www.apple.com/de/ios/preview
- How Delegation Works
  https://www.andrewcbancroft.com/2015/04/08/how-delegation-works-a-swift-developer-guide
- Quick Guide to Swift Delegates
  http://useyourloaf.com/blog/quick-guide-to-swift-delegates
- iOS Swift Basics Tutorial: Protocols and Delegates
  https://www.youtube.com/watch?v=9LHDsSWc680

Aufruf von Delegate-Funktionen

Wir haben bisher nun die verschiedenen Grundlagen kennengelernt, die notwendig sind, um Delegation verwenden zu können. Eigenschaften jeder Art kann statt einer Klasse auch ein Protokoll als Typ zugewiesen werden, wodurch es keine Rolle mehr spielt, von welcher Klasse das zugewiesene Objekt ist, solange die Klasse konform zum zugehörigen Protokoll ist.

Betrachten wir noch einmal die *vehicle*-Property unserer *Driver*-Klasse, die dem Typ *Drivable* entspricht. Es können nun problemlos alle Eigenschaften auf dieser Property genutzt und aufgerufen werden, die innerhalb des Protokolls *Drivable* deklariert sind. Listing 3 zeigt einmal verschiedene Beispiele, wie der Aufruf von Eigenschaften eines Protokolls erfolgt.

Dabei wird *vehicle* zunächst wieder das *myCar*-Objekt zugewiesen, und anschließend werden die verschiedenen Eigenschaften verwendet. Dann wird noch einmal das *myTruck*-Objekt für *vehicle* genutzt und wieder die unterschiedlichen Eigenschaften aufgerufen. Sie liefern bei *myCar* und *myTruck* jeweils unterschiedliche Ergebnisse, was nicht weiter verwunderlich ist; schließlich haben die jeweils zugrunde liegenden Klassen *Car* und *Truck* die jeweiligen Eigenschaften des Protokolls unterschiedlich implementiert.

Aber genau diese unterschiedlichen Ergebnisse zeigen die Stärken und den Sinn von Delegation: Ein Protokoll definiert, welche Eigenschaften zur Verfügung stehen, und je nachdem, welche Klasse mitsamt zugehöriger eigener Implementierung als Delegate-Objekt verwendet wird, erhält man ein anderes Ergebnis.

Dabei ist natürlich eines zu beachten: Die Property *vehicle* hat zwar vollen Zugriff auf alle Eigenschaften, die im Protokoll *Drivable* deklariert sind (schließlich ist die Property *vehicle* auch vom Typ *Drivable)*, aber nicht auf sonstige spezifische Eigenschaften, die in den Klassen *Car* und *Truck* deklariert sind.

Nehmen wir als Beispiel die Property *weight* der Klasse *Truck*. Auch wenn *vehicle* eine Instanz der Klasse *Truck* zugewiesen wird, so kann *vehicle* zunächst einmal nur auf die Eigenschaften des Protokolls und nicht der spezifischen Klasse zugreifen. Möchte man spezifische Eigenschaften nutzen, so muss dazu ein entsprechendes Type Casting durchgeführt werden.

Beispiele von Delegation in der iOS-Entwicklung

Delegation findet sich in der iOS-Entwicklung häufig bei verschiedenen UIView-Subklassen wieder. Das wohl bekannteste Beispiel ist die Klasse *UITableView*. Sie verfügt über zwei Properties, die mit Delegation arbeiten: *dataSource* und *delegate*. *dataSource* ist dabei eine Property vom Protokoll-Typ *UITableViewDataSource* und *delegate* eine Property vom Protokoll-Typ *UITableViewDelegate*.

Diese Protokolle definieren, wie eine Table-View aufgebaut ist und wie sie funktioniert. Da typischerweise jede Table-View, die wir in eigenen Apps umsetzen, anders funktioniert, anders aussieht und andere Daten anzeigt, erlauben es die Protokolle, beliebige Klassen mit der zugehörigen Logik zu erstellen und den genannten Properties der Table-View zuzuweisen.

Je nachdem, welche Logik das ist, funktioniert die Table-View anders und sieht anders aus. Einzige Voraussetzung ist eben, dass die entsprechenden Klassen die passenden Protokolle implementieren.

Fazit

Delegation ist eine der wichtigsten und gleichzeitig mächtigsten Technik in der Programmierung mit Swift. Es ist auch ein essenzieller Bestandteil bei der Programmierung von iOS-Apps, bei der sich das Konzept von Delegation in vielen Systemklassen wiederfindet.

Erfreulicherweise ist die Arbeit mit und Verwendung von Protokollen in Swift noch einmal wesentlich leichter und komfortabler gelöst als in Objective-C. Die Tatsache, dass ein Protokoll einen vollwertigen Typ in Swift definiert, macht die Verwendung von Protokollen für Variablen, Konstanten, Properties et cetera einfach und flexibel. Das Beispiel mit der *vehicle*-Property hat gezeigt, wie leicht und übersichtlich sich Delegation in eigenen Projekten umsetzen lässt. ■

Thomas Sillmann
ist iOS-App-Entwickler, Trainer und Autor. Freiberuflich tätig programmiert er für den App Store eigene Apps sowie Apps in Form von Kundenaufträgen. Er ist Autor eines erfolgreichen Fachbuchs und mehrerer Artikel in Fachzeitschriften.

SMART DATA
Developer Conference
Big Data Analytics

- 18.04.2016 – Konferenz
- 19.04.2016 – Workshops

Novotel München City

Die SMART DATA Developer Conference macht Softwareentwickler mit den Herausforderungen von Big Data vertraut.

- In den Konferenz-Sessions erlangen Sie Wissen zu Speicherung, Analyse, Plattformen und Tools.
- Die Workshops bieten intensives Training in den Technologien Multi-Model-Database mit CQRS und OrientDB, Microsoft Azure und Entity Framework sowie Smart Development mit Enapso.

Jetzt anmelden! Mit Code **SMART16dwmd** bis zu 150,– EUR sparen

Online Special

- Kostenfreie Webinare
- Blog
- Konferenz 2015 als Video

smart-data-developer.de **#smartddc** **SMARTDATADeveloperConference**

Präsentiert von: dotnetpro, web & mobile DEVELOPER

Veranstalter: developer media, Neue Mediengesellschaft Ulm mbH

SMART DATA
Developer Conference
Big Data Analytics

Konferenz-Programm: 18. April 2016

09.00 Uhr

Clouds in the Wind – Big und Smart Data in der Cloud und trotzdem beweglich • *Michael Nolting*

Smart und Big Data sind die Trendthemen heutzutage. Aber welche Technologie und welcher Anbieter sind für den speziellen Anwendungsfall am besten geeignet? Die entwickelte Entscheidungsmatrix soll den Zuhörer mündig machen, die im Markt verfügbaren Technologie-Stacks schnell zu vergleichen und die für den eigenen Anwendungsfall am besten geeigneten zu finden.

10.00 Uhr — Kaffeepause

10.30 Uhr

Smart Data mit .NET

Performance trotz Entity Framework
• *André Krämer*

Ist das Entity Framework wirklich langsam? JA! In dieser Session werden wir uns ansehen, was das Entity Framework langsam macht und was wir dagegen unternehmen können!

Best Practices

Datenqualität für Entwickler
• *Werner Keil*

Big Data ohne Datenqualität wird chaotisch und bedeutungslos. Die Session bietet einen Überblick bekannter Standards, sowie deren Unterstützung durch Sprachen, APIs etc.

11.30 Uhr

Smart Data mit .NET

Datenpipeline-Komponenten für die SSIS
• *Thomas Worm* — Blog

Die SQL Server Integration Services ermöglichen die Integration von Datenbanken in vorhandene IT-Landschaften. Die Session behandelt Framework und Eigenheiten der SSIS.

Best Practices

Daten entlang der Seidenstraße
• *Jan Fellien* — Blog

Die Session stellt die Multi-Model-Graphen-Datenbank OrientDB mit node.js vor. Sie kann in Documents wie in Graphen speichern, was die Anwendung im Alltag sehr vereinfacht.

12.30 Uhr — Mittagspause

13.30 Uhr

Smart Data mit .NET

Datenquellen mit F# Data und Deedle
• *Stefan Lieser*

Mit diesen beiden Open-Source-Bibliotheken ist der Zugriff auf Datenquellen sowie ihre Analyse und Aggregation auf einfache Weise möglich.

Best Practices

Das Analytics Framework BRAIN Reflex
• *Rupert Steffner und Jan Lendholt*

Fokus der Session ist das Streaming Analytics Framework und dessen containerized Micro-Service-Architektur. Gezeigt werden Design Patterns und Use Cases.

14.30 Uhr

Smart Data mit .NET

Interaktive Analysen mit Apache Spark
• *Olivia Klose und Sascha Dittmann* — Video

Bei interaktiven Datenabfragen schwächeln die meisten Big Data Technologien. Diese Session zeigt die wichtigsten Grundlagen der In-Memory Technologie anhand von Codebeispielen.

Best Practices

Mehr als SQL Server in the Cloud
• *Constantin Klein* — Video

Gewinnen Sie in dieser Session einen Überblick über die Microsoft Cloud Data Platform und die Möglichkeiten beider Bereiche „Data & Storage" und „Analytics".

15.30 Uhr — Kaffeepause

16.00 Uhr

Smart Data mit .NET

Looking for the meaning to our data
• *Diego Poza*

This talk is going to get deep into real world scenarios using Azure Machine Learning. Build from scratch a predictive model, comparing algrorithms and adding custom modules.

Mensch & Maschine

The UX of Data: Responsive Visualisierung
• *Peter Rozek*

Big Data ist nicht gleich Smart Data. Interaktive und skalierbare Datenvisualisierung ist der nächste Schritt, um Daten für den Nutzer nutzerfreundlich aufzubereiten.

17.00 Uhr bis 18.00 Uhr

Smart Data mit .NET

Hey Cortana - let's talk about Analytics!
• *Olivia Klose, Majid Latif und Marcel Tilly*

Ein Tiefflug über den Cortana Analytics Stack: Wie spielen die einzelnen Dienste, wie EventHub, Stream Analytics, Azure Data Lake, Azure Machine Learning und PowerBI zusammen?

Mensch & Maschine

Ergonomische Datendarstellung
• *Daniel Greitens*

Daten sind nur so gut, wie dem Betrachter verständlich. Dieser Vortrag liefert psychologische Hintergründe und einen strukturierten Weg, wie Daten optimal aufbereitet werden können.

Programmänderung vorbehalten

Workshops: 19. April 2016

Tipp: Kostenfreie Webinare als kompakte Einführung und Kennenlernen der Workshop-Leiter. Weitere Infos auf der Webseite: smart-data-developer.de

Workshop 1 – CQRS und Multi-Model-DB – ein Herz und eine Seele

Jan Fellien Uhrzeit: 09.00 – 18.00 Uhr

Lange habe ich nach einer guten Lösung für eine Datenbank gesucht, die sich ins CQRS Umfeld gut einfügt. Mit der OrientDB scheint die Suche ein Ende zu haben. Hiermit können Graphen und Documents gleichermaßen gespeichert (sprich ein Event Store) und die Read Models passgenau zu den Anforderungen abgelegt werden.

Der Workshop führt Sie zunächst in das CQRS-Paradigma ein und leitet Sie behutsam hinüber in die Datenspeicherung mit node.js und OrientDB.

Workshop 2 – Smart Development mit Enapso

Alexander Schulze Uhrzeit: 09.00 – 18.00 Uhr

Big-Data-Analysen erzeugen Wissen als Basis für smarte Maschinen, für Vorhersagen, für Empfehlungs- und Assistenz-Systeme. Doch smarte Technologien verbessern nicht nur Software als Ergebnis, sondern auch deren Entwicklung.

Dieser Workshop stellt Modelle, Abfragen und Regeln semantischer Ontologien mit OWL2, SPARQL und SWRL vor. Wir erläutern Machine-Learning, BPMN mit Micro-Services in der Cloud und wie intelligente Agenten als Assistenten zur Softwareentwicklung in Enapso Anforderungen und Wissen verbinden.

Workshop 3 – Smart Data with Microsoft Azure

Sascha Dittmann, Olivia Klose Uhrzeit: 09.00 – 18.00 Uhr

In diesem Workshop werden wir anhand eines anschaulichen Szenarios die Analytics-Dienste der Azure-Datenplattform vorstellen und selber aufbauen, mit denen aus der Datenflut hilfreiche Erkenntnisse gewonnen werden können:

Azure Data Factory, HDInsight (Hadoop auf Azure), Stream Analytics und Machine Learning.

Beispielsweise wird erläutert, wie Erkenntnisse zum Einkaufsverhalten sowohl aus historischen Daten als auch in Echtzeit gewonnen und darauf basierend dem Kunden mithilfe von Machine Learning personalisierte Empfehlungen gegeben werden können.

Workshop 4 – Enterpriseanwendungen mit Entity Framework 6

Christian Giesswein Uhrzeit: 09.00 – 18.00 Uhr

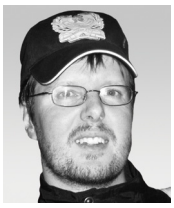
Entity Framework gilt als DIE Lösung im Zusammenhang mit dem .NET-Framework und relationalen Datenbanken. Doch kaum geht eine Anwendung über das beliebte „Hello World"-Projekt hinaus, tauchen die wirklichen Fragen und Probleme erst auf. Plötzlich wird über die Performance gemeckert, oder den Komplexitätsgrad der Abfragen. Denn das Entity Framework ist bei falscher Verwendung alles andere als dankbar.

Anhand von „Best Practices" verschiedener Projekte erfahren Sie, wo die Stolpersteine des Entity Framework liegen, wenn es um die Einbindung in komplexe Enterprise-Anwendungen geht, und erhalten wertvolle Tipps und Anleitungen, um diese zu überwinden.

Die Referenten der SMART DATA Developer Conference (u.a.):

Jan Fellien,
devCrowd GmbH

Christian Giesswein
Giesswein Software-Solutions



Werner Keil,
Creative Arts & Technologies

Constantin Klein,
Freudenberg IT GmbH & Co. KG

Olivia Klose,
Microsoft Deutschland GmbH

Stefan Lieser,
CCD School

Michael Nolting,
Sevenval Technologies GmbH

Jan-Hendrik Lendholt,
Otto (GmbH & Co KG)

Alexander Schulze,
Innotrade GmbH

Marcel Tilly,
Microsoft Research

IOS NSOPERATION UND DISPATCH QUEUE

Parallelbetrieb

Die Ausführung von mehreren Threads in einer App wird mit den Klassen NSOperation–Queue und NSBlockOperation umgesetzt.

Es gibt immer etwas zu tun … – diesen Spruch kennt wohl jeder, und diese Aussage kann man natürlich auch auf iOS anwenden. Aber die anfallenden Arbeiten sollen natürlich am besten gleichzeitig und still und leise im Hintergrund erledigt werden. iOS bietet für diese Anforderung sogar geeignete Mechanismen an.

Die Bearbeitung von Aufgaben, beispielsweise die Interpretation einer Geste auf dem Touchscreen, wird in der Regel aus dem Main-Thread einer App heraus bewältigt. Solange diese Operation andauert, blockiert diese die weitere Verarbeitung und somit auch die komplette App. Soll beispielsweise ein dauerhafter oder lang andauernder Prozess verarbeitet werden, etwa die Wiedergabe von Musik, so ist es besser, wenn eine solche Aufgabe, sofern dies möglich ist, in den Hintergrund verlagert wird.

Die App wird hierdurch nicht komplett blockiert und ein Anwender kann weiter mit ihr arbeiten. Gerade aus diesem Grund ist die Parallelverarbeitung von Aufgaben unter iOS aber auch eine Angelegenheit, die wohlbedacht und geplant eingesetzt werden sollte. Unter iOS gibt es zwei APIs, die für die Verarbeitung solcher Prozesse verwendet werden können: NSOperation und Dispatch Queue (Bild 1).

Zuerst soll die Warteschlangenverarbeitung betrachtet werden. Hinter dem Begriff Grand Central Dispatch, kurz GCD, verbergen sich Funktionen der Sprache Swift sowie Bibliotheken, welche die Ausführung von Code ermöglichen, der gleichzeitig ausgeführt werden soll. Der Code beziehungsweise die Anweisungen werden dabei mit Hilfe sogenannter Queues (Blöcken von Code) verarbeitet.

Die Aufgaben (in Form von Objekten) werden hierbei nach dem FIFO-Prinzip (First In, First Out) abgearbeitet. Also wer zuerst da ist, der mahlt auch zuerst. Die erste Aufgabe, welche in diese Warteschlange gestellt wird, wird auch zuerst aus der Warteschlange entfernt.

Man spricht in diesem Zusammenhang auch von Dispatch Queues. Diese Queues werden zwar aus dem Main-Thread der App heraus gestartet, werden dann aber in einem eigenen, vom Main-Thread unabhängigen Prozess ausgeführt. Für die Verarbeitung werden zwei Queue-Typen unterschieden: die Serial und die Concurrent Queues.

Serielle Queues

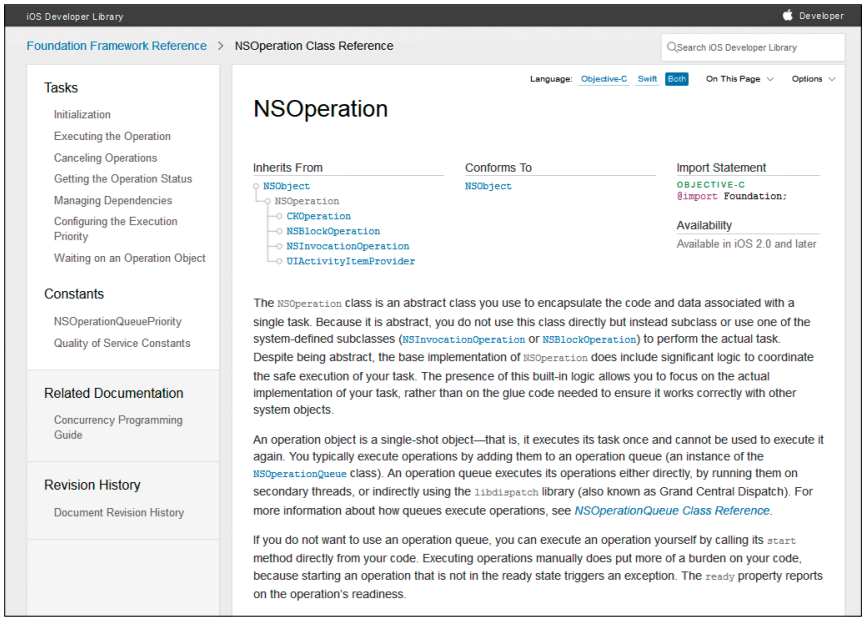
Eine Serial Queue ist eine Warteschlange, die Objekte seriell verarbeitet. Eine seriell arbeitende Queue kann immer nur eine Aufgabe gleichzeitig abarbeiten. Aus diesem Grund werden auch alle Arbeitsaufträge, welche derselben Warteschlange übergeben wurden, nacheinander ausgeführt. Es ist aber möglich, mehrere seriell arbeitende Warteschlangen nebeneinander zu betreiben.

Hat man also zwei Serial Queues, so werden die jeweils zugeordneten Arbeitsaufträge dann gleichzeitig abgearbeitet. Serial Queues eigenen sich beispielsweise sehr gut, um Arbeitsaufträge, welche aus einer gemeinsamen Quelle kommen, abzuarbeiten. Denken Sie beispielsweise an mehrere Kassen vor einem Zoo.

An jeder Kasse kann immer nur ein Kunde eine Eintrittskarte kaufen. Da es aber mehrere Kassen gibt, können auch mehrere Kunden gleichzeitig eine Karte erwerben.

Seriell arbeitende Queues sind immer dann sinnvoll einsetzbar, wenn sichergestellt sein muss, dass die Arbeitsaufträge nacheinander und in einer vorgegebenen Reihenfolge abgearbeitet werden sollen.

Neben den seriell arbeitenden Queues gibt es dann auch noch die, welche



iOS stellt zwei APIs für die Verarbeitung paralleler Prozesse zur Verfügung (Bild 1)

gleichzeitig beziehungsweise konkurrierend arbeiten. Diese werden im Fachjargon als Concurrent Queues bezeichnet.

Der große Unterschied zu den seriell arbeitenden Queues ist, dass die in die Warteschlange eingestellten Arbeitsaufträge nicht nacheinander, sondern parallel abgearbeitet werden. Dabei kann es passieren, dass die Bearbeitung eines Arbeitsauftrags, welcher erst als zweiter in die Warteschlange gestellt wurde, noch vor dem ersten beendet wird.

Wie werden die beiden unterschiedlichen Queue-Typen nun in einer Anwendung verwendet?

Queue im Main-Thread

Einer App werden vom Betriebssystem standardmäßig eine serielle und vier parallel arbeitende Queues zur Verfügung gestellt. Man sollte sich bei der Entscheidung vor Augen führen, dass die seriell arbeitende Queue im Main-Thread zur Verfügung gestellt wird. Diese Queue wird auch verwendet, um das UI zu aktualisieren. Wenn man hier eine Aufgabe einstellt, kann es also dazu kommen, dass das UI blockiert wird.

Aus diesem Grund sollte man zuerst zu den parallel arbeitenden Queues greifen. Vier davon werden wie beschrieben einer App zur Verfügung gestellt. Diese unterscheiden sich in der Höhe der Priorität, mit der die eingestellten Aufgaben bearbeitet werden. Mit dem ersten Parameter, welcher der Funktion *dispatch_get_global_queue* übergeben wird, wird festgelegt, mit welcher Priorität die eingestellte Aufgabe bearbeitet wird:

```
DISPATCH_QUEUE_PRIORITY_HIGH,
DISPATCH_QUEUE_PRIORITY_DEFAULT,
```

▶

Listing 1: Dispatch mit Concurrent Queues

```
import UIKit

class ViewController: UIViewController {

  let urls = ["https://christianbleske.files.
  wordpress.com/2015/12/unterwasser.jpg",
  "https://christianbleske.files.wordpress.com/
  2015/12/wasser.jpg",
  "https://christianbleske.files.wordpress.com/
  2015/12/landschaft.jpg"]

  @IBOutlet var imageView1: UIImageView!
  @IBOutlet var imageView2: UIImageView!
  @IBOutlet var imageView3: UIImageView!

  override func viewDidLoad() {
    super.viewDidLoad()
  }

  override func
    didReceiveMemoryWarning() {
      super.didReceiveMemoryWarning()
  }

  @IBAction func loadButton(sender:
    AnyObject) {

    let queue =
      dispatch_get_global_queue(
        DISPATCH_QUEUE_PRIORITY_DEFAULT,
          0)

    dispatch_async(queue) { () ->
      Void in

      let loader1 =
        ImageLoader.loadImageFromURL(
          self.urls[0])
      dispatch_async(
        dispatch_get_main_queue(), {
          self.imageView1.image =
            loader1
      })
    }

    dispatch_async(queue) { () ->
      Void in

      let loader2 =
        ImageLoader.loadImageFromURL(
          self.urls[1])

      dispatch_async(
        dispatch_get_main_queue(), {
          self.imageView2.image =
            loader2
      })
    }

    dispatch_async(queue) { () ->
      Void in

      let loader3 =
        ImageLoader.loadImageFromURL(
          self.urls[2])

      dispatch_async(
        dispatch_get_main_queue(), {
          self.imageView3.image =
            loader3
      })
    }
  }
}
```

```
DISPATCH_QUEUE_PRIORITY_LOW,
DISPATCH_QUEUE_PRIORITY_BACKGROUND.
```

Die Bezeichnung der Werte zeigt an, wie ein eingestellter Arbeitsauftrag behandelt werden soll.

Queues in Action

Bisher wurde die Thematik nur theoretisch besprochen. Aber nichts geht über die Praxis. In einem kleinen Beispiel sollen einmal die beiden Queue-Typen ausprobiert werden. Im Beispielprojekt *(DispatchQueueBsp)* werden Bilder von einem angebenden URL in eine App geladen und dort dann angezeigt. Insgesamt werden drei Bilder geladen und im Client präsentiert (Bild 2).

Zum Laden der Bilder muss die entsprechend beschriftete Schaltfläche betätigt werden. Innerhalb der dem Button zugeordneten Action befindet sich dann der Code, der verwendet wird, um die Bilder zu laden (Listing 1).

Zu Beginn wird in der Methode *loadButton* eine neue DispatchQueue erzeugt. In diese Queue werden die Aufgaben eingestellt werden. Bei der Anlage der Queue wird der Parameter *DISPATCH_QUEUE_PRIORITY_DEFAULT* übergeben, was zur Folge hat, dass der eingestellte Auftrag mit keiner hohen Priorität bearbeitet wird.

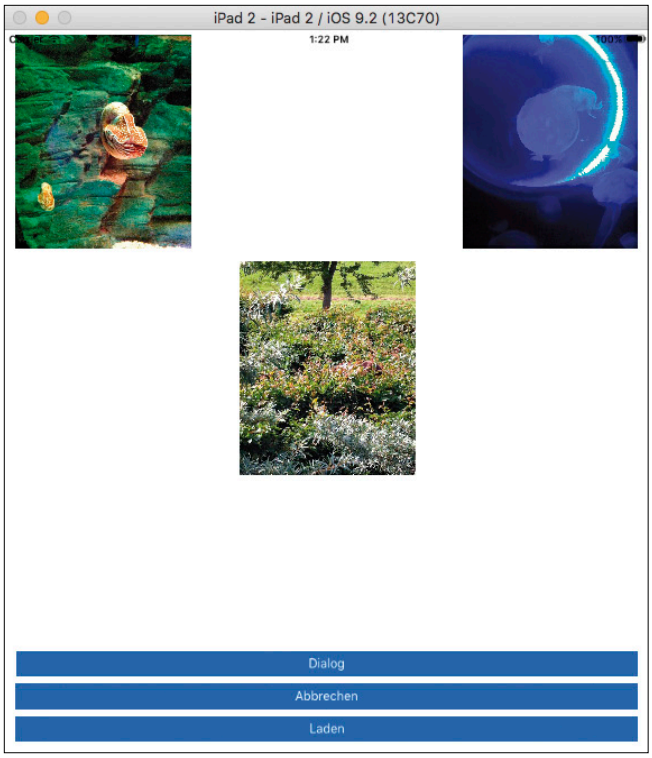
Die erzeugte Queue wird anschließend der Funktion *dispatch_async* übergeben. Innerhalb der Funktion befindet sich der Codeblock, der in der Queue ausgeführt werden soll. Von der Klasse *ImageLoader* wird anschließend eine Instanz erzeugt. Die Klasse hat nur eine (statische) Funktion *loadImageFormURL,* deren einzige Aufgabe es ist, ein Bild von einem als Parameter übergebenen URL zu laden (Listing 2).

Nachdem das Bild geladen wurde, wird dieses als Return-Wert einer lokalen Variable zugewiesen, um anschließend im Main-Thread (der Download wurde ja im Background ausgeführt) einem ImageView-Control zur Anzeige zugewiesen zu werden. Da insgesamt drei Bilder geladen werden sollen, wird auch dreimal eine Aufgabe in die Queue eingestellt. Da die Aufgabe im Hintergrund ausgeführt wurde, ist weiterhin eine Interaktion mit der Oberfläche der App möglich.

Verwendet man statt einer Concurrent Queue eine Serial Queue, so sind zwei Dinge unterschiedlich. Das Laden der

Listing 2: Die Klasse ImageLoader

```
import Foundation
import UIKit

class ImageLoader {
  class func loadImageFromURL(
    url:String) -> UIImage! {
    let nsData = NSData(contentsOfURL:
      NSURL(string: url)!)
    return UIImage(data: nsData!)
  }
}
```



Die Beispiel-App, nachdem die Bilder geladen wurden (Bild 2)

Bilder dauert einen Moment länger, was daran liegt, dass die Bilder nun seriell nacheinander geladen werden. Und noch etwas fällt auf: Die Bilder werden in der Reihenfolge geladen, in der die Aufträge in die Queue geschrieben wurden. Benötigt man also eine festgelegte Reihenfolge bei der Verarbeitung, so ist eine Serial Queue das Mittel der Wahl.

Wie unterscheidet sich die Serial von der Concurrent Queue im Code? Listing 3 zeigt die Antwort. Im Listing ist nicht die komplette Klasse abgebildet, sondern nur die Erzeugung der Queue und der Arbeitsaufträge.

Das Anlegen der seriell arbeitenden Queues unterscheidet sich etwas von der zuvor erzeugten parallel arbeitenden Queue. Eine andere Funktion *dispatch_queue_create* wird verwendet, und auch die verwendeten Parameter unterscheiden sich. Die einzelnen Codeblöcke hingegen sind, mit Ausnahme der übergebenen Queue, identisch zum Concurrent-Queue-Beispiel. Auch in diesem Fall lässt sich zur Laufzeit beziehungsweise während des Ladens der Bilder ein Dialog anzeigen. Somit wird deutlich, dass auch in diesem Fall der UI-Thread nicht blockiert.

NSOperation Queues

Es gibt noch einen weiteren Queue-Typ: Operation Queues. Der große Unterschied ist, das diese Queues auf einer höheren Abstraktionsebene funktionieren und einige Funktionen zusätzlich zur Verfügung stellen. Anzumerken ist, dass Operation Queues nicht nach dem FIFO-Prinzip arbeiten.

Den Aufträgen kann eine Priorität zur Abarbeitung zugeordnet werden, und es ist außerdem auch möglich, Abhängigkeiten zwischen den Aufträgen zu definieren, sodass sichergestellt ist, dass eine bestimmte Aufgabe zuerst und ei-

Listing 3: Serial Queue im Einsatz

```
let serialQueue = dispatch_queue_
create("mySerialQueue",
DISPATCH_QUEUE_SERIAL)
dispatch_async(serialQueue) { () -> Void in
  let loader1 = ImageLoader.loadImageFromURL
    (self.urls[0])
  dispatch_async(dispatch_get_main_queue(), {
    self.imageView1.image = loader1
  })
}
dispatch_async(serialQueue) { () -> Void in
  let loader2 = ImageLoader.loadImageFromURL
    (self.urls[1])
  dispatch_async(dispatch_get_main_queue(), {
    self.imageView2.image = loader2
  })
}
dispatch_async(serialQueue) { () -> Void in
  let loader3 = ImageLoader.loadImageFromURL
    (self.urls[2])
  dispatch_async(dispatch_get_main_queue(), {
    self.imageView3.image = loader3
  })
}
```

ne andere, die davon abhängig ist, erst im Anschluss abgearbeitet wird. Operation Queues sind Instanzen der Klasse *NSOperationQueue*. Außerdem kann für die Ausführung ein Status vergeben werden. Hierfür wird die Enumeration *NSOperationQueuePriority* verwendet. Diese enthält die Stufen: *VeryLow, Low, Normal, High, VeryHigh*.

Außerdem ist es möglich, eine oder auch mehrere laufende Operationen abzubrechen. Hierfür wird die *Cancel*-Methode verwendet. Mittels der Eigenschaften: *finished, cancelled* und *ready* kann der aktuelle Status einer Operation abgefragt werden. Optional ist es auch möglich, einen sogenannten CompetionBlock aufzurufen, wenn eine Operation abgeschlossen wurde. In Listing 4 sehen Sie Beispiele für diese Möglichkeiten. Zu Beginn der Klasse *ViewController* wird eine globale Instanz der Klasse *NSOperationQueue* erzeugt. Auf diese wird beispielsweise zugegriffen, um alle laufenden Operationen abzubrechen. In der Action *cancelButton* wird dies demonstriert, indem dort die Methode *cancelAllOperations* aufgerufen wird. So kann das Laden der Bilder abgebrochen werden. Innerhalb der Action *loadButton* befindet sich (wie zuvor) der Code für das Erstellen der Aufträge.

Im ersten Schritt wird ein neuer Block durch Ableitung der Klasse *NSOperationBlock* erzeugt. In diesem Block befindet sich abermals der Code zum Laden der Bilder. Auffällig ist ▶

Listing 4: NSOperationQueue im Einsatz (Teil 1)

```
import UIKit
class ViewController: UIViewController {

  var nsOperationQueue =
    NSOperationQueue()

  let urls = [Quellcode entfernt]

  @IBOutlet var imageView1:
    UIImageView!
  @IBOutlet var imageView2:
    UIImageView!
  @IBOutlet var imageView3:
    UIImageView!

  @IBAction func cancelButton(sender:
    AnyObject) {
      nsOperationQueue.
        cancelAllOperations()
      print("Operation abgebrochen!")
  }

  @IBAction func dialogButton(sender:
    AnyObject) {
      showAlertViewWithTitle("Info",
        message: "Dies ist ein Dialog")
      print("Dialog wurde aufgerufen!")
  }

  @IBAction func loadButton(sender:
    AnyObject) {
      nsOperationQueue =
        NSOperationQueue()

      let nsBlockOperation1 =
        NSBlockOperation(block: {
          let loader1 = ImageLoader.
            LoadImageFromURL(
              self.urls[0])

        NSOperationQueue.mainQueue().
          addOperationWithBlock({
            self.imageView1.image =
              loader1
          })
        })
```

hier, dass der Code für die Zuweisung des Bildes in einer separaten zusätzlichen Methode *addOperationWithBlock* durchgeführt wird. Wird die Operation beendet, so soll diese ausgegeben werden. Hierzu wird ein neuer *CompletionBlock* erzeugt, in dem dann eine Ausgabe erfolgt. Zuletzt wird der definierte Block mittels der Methode *addOperation* der *NSOperationQueue* hinzugefügt. Durch die Verwendung von *NSOperationQueue* stehen mehr Möglichkeiten zur Verfügung, um auf die Verarbeitung Einfluss zu nehmen.

Fazit

iOS bietet dem Entwickler unterschiedliche Wege, Aufgaben parallel zu erledigen. Wenn die möglichen Techniken bekannt sind, kann die eigene App sehr simpel um entsprechende Funktionen erweitert werden. ■

Links zum Thema

- Framework Reference
  https://developer.apple.com

Christian Bleske
ist Autor, Trainer und Entwickler mit den Schwerpunkten Client/Server und mobile Technologien. Sein Arbeitsschwerpunkt liegt auf Microsoft-Technologien.
cb.2000@hotmail.de

Listing 4: NSOperationQueue im Einsatz (Teil 2)

```
nsBlockOperation1.
  completionBlock = {
    print("Laden von Bild 1
      durchgeführt = \
        (nsBlockOperation1.cancelled)")
}
nsOperationQueue.addOperation(
  nsBlockOperation1)

let nsBlockOperation2 =
  NSBlockOperation(block: {
    let loader2 = ImageLoader.
      LoadImageFromURL(
        self.urls[1])
  NSOperationQueue.mainQueue().
    addOperationWithBlock({
      self.imageView2.image =
        loader2
  })
})

nsBlockOperation2.
  completionBlock = {
    print("Laden von Bild 2
      durchgeführt = \
        (nsBlockOperation2.cancelled)")
}
nsOperationQueue.addOperation(
  nsBlockOperation2)

let nsBlockOperation3 =
  NSBlockOperation(block: {
    let loader3 = ImageLoader.
      LoadImageFromURL(
        self.urls[2])
  NSOperationQueue.mainQueue().
    addOperationWithBlock({
      self.imageView3.image =
        loader3
  })
})

  nsBlockOperation3.
    completionBlock = {
      print("Laden von Bild 3
        durchgeführt = \
          (nsBlockOperation3.cancelled)")
  }
  nsOperationQueue.addOperation(
    nsBlockOperation3)
}

func showAlertViewWithTitle(
  title:String, message:String) {

  let alertController =
    UIAlertController(
      title: title, message: message,
        preferredStyle: .Alert)

  let OKAction = UIAlertAction(
    title: "OK", style: .Default)
    { (action) in
    }

  alertController.addAction(
    OKAction)

  self.presentViewController(
    alertController, animated: true)
  {
  }
 }
}
```

Jetzt kostenlos testen!

Das Fachmagazin für IT-Entscheider

2 Ausgaben kostenlos testen. Mit exklusivem Zugang zu unseren Digitalausgaben. Business-Newsletter inklusive.

www.com-magazin.de/gratis

MOBILE APPS MIT CORDOVA

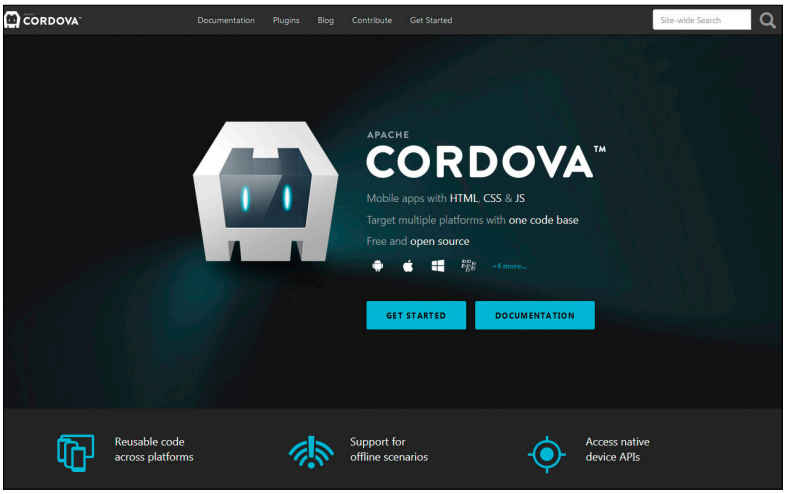
Hybride Applikationen

Mit dem Cordova-Framework lassen sich hybride Applikationen in HTML5, CSS und JavaScript für mobile Endgeräte erstellen.

Ursprünglich 2009 als PhoneGap von der Firma Nitobi auf den Markt gebracht, wurde das Framework mittlerweile von Adobe aufgekauft und als Adobe Cordova (http://cordova.apache.org) unter die Open-Source-Lizenz Apache 2.0 gestellt (Bild 1).

Unter http://phonegap.com findet sich zudem eine Distribution von Cordova mit dem Namen PhoneGap, die – bis auf wenige Ausnahmen – nahezu identisch zu Cordova ist. Unterschiede sind in der Lizenz und im Betreiber zu finden, denn PhoneGap wird von Adobe betrieben. Wir verwenden aber im nachfolgendem Artikel Cordova, da es die Basis darstellt.

Mit Hilfe von Cordova kann ein auf Basis von HTML5, CSS und JavaScript entwickelter Code auf zahlreichen nativen Plattformen lauffähig gemacht werden. Zurzeit werden die folgenden Plattformen unterstützt: iOS, Android, webOS, Windows Phone >8.0, BlackBerry10, Amazon Fire OS, Tizen und Ubuntu.



Projekt: Die Projektseite von Cordova (Bild 1)

Zugriff auf interne Funktionen

Dabei ist man aber nicht nur auf die nativen HTML5-Fähigkeiten begrenzt. Mit Hilfe spezieller Bibliotheken (APIs) kann auch auf die Hardware und interne Funktionen der mobilen Geräte zugegriffen werden, wie Beschleunigungssensor, Kamera, Kompass, Kontakte, Dateien, Geolocation, Medien, Netzwerk, Benachrichtigungen und den Speicher. Allerdings unterstützt nicht jede Plattform alle Funktionen – eine Übersicht darüber finden Sie unter https://cordova.apache.org/docs/en/latest/guide/support (Bild 2).

Je nachdem, welche Zielplattform Sie wählen, benötigen Sie ein bestimmtes Setup.

Für iOS etwa müssen Sie unter Mac OS X entwickeln, da die dazu notwendige Plattform Xcode nur unter diesem Betriebssystem zur Verfügung steht. Den Download finden Sie unter https://developer.apple.com/xcode.

Für die Windows-Phone-Entwicklung wiederum müssen Sie auch unter Windows arbeiten, da hierfür Microsoft Visual Studio benötigt wird. Dieses finden Sie unter https://www.visualstudio.com. Alle anderen Plattformen wiederum können auf beiden Betriebssystemen oder sogar unter Linux bedient werden – denn hierfür wird (meist) Eclipse verwendet, das selbst eine Java-Applikation ist, die plattformübergreifend lauffähig ist. Speziell für die Android-Entwicklung können Sie das ADT (Android Developer Tools) verwenden, das Sie unter der Adresse http://developer.android.com/tools/help/adt.html finden.

Platform Support

The following shows the set of development tools and device APIs available for each mobile platform. The device APIs listed here are provided by the core plugins, additional APIs are available via third-party plugins. Column headers display the CLI's shorthand names.

	amazon-fireos	android	blackberry10	Firefox OS	ios	Ubuntu	wp8 (Windows Phone 8)	windows (8.0, 8.1, 10, Phone 8.1)
cordova CLI	✔ Mac, Windows, Linux	✔ Mac, Windows, Linux	✔ Mac, Windows	✔ Mac, Windows, Linux	✔ Mac	✔ Ubuntu	✔ Windows	✔
Embedded WebView	✔ (see details)	✔ (see details)	✘	✘	✔ (see details)	✔	✘	✘
Plug-in Interface	✔ (see details)	✔ (see details)	✔ (see details)	✘	✔ (see details)	✔	✔ (see details)	✔
Platform APIs								
Accelerometer	✔	✔	✔	✔	✔	✔	✔	✔
BatteryStatus	✔	✔	✔	✔	✔	✘	✔	✔ * Windows Phone 8.1 only
Camera	✔	✔	✔	✔	✔	✔	✔	✔
Capture	✔	✔	✔	✘	✔	✔	✔	✔
Compass	✔	✔	✔	✘	✔ (3GS+)	✔	✔	✔
Connection	✔	✔	✔	✘	✔	✔	✔	✔
Contacts	✔	✔	✔	✔	✔	✔	✔	partially

Übersicht: Welche Plattform welche Funktionen unterstützt (Bild 2)

Die Installation von Cordova wird per Kommandozeile durchgeführt. Vorher müssen Sie aber noch folgende weitere Tools installieren: Node.js (https://nodejs.org/en), Git (http://git-scm.com) und – wenn Sie unter Android entwickeln wollen – Apache Ant (http://ant.apache.org). Nun öffnen Sie ein Terminal-Fenster und tippen folgenden Befehl ein:

```
$ sudo npm install -g cordova
```

Sollten Sie eine andere Plattform als Mac OS X einsetzen, finden Sie unter https://cordova.apache.org/docs/en/latest/guide/plat

forms die passende Anleitung zur Installation.

Hier wird mit Root-Rechten der Node Package Manager mit der Option zum (globalen) Installieren des Pakets Cordova aufgerufen. Um unter Mac OS X den iOS Simulator auch von der Kommandozeile aus bedienen zu können, installieren wir diesen analog:

```
$ sudo npm install -g ios-sim
```

Wir erstellen nun einer erste App, um den Ablauf kennenzulernen. Dafür setzen wir auf der Kommandozeile folgenden Befehl ab:

```
$ cordova create hello
com.example.hello helloworld
Creating a new cordova project.
$ cd hello
```

Der Name des Projektverzeichnisses ist *hello,* als Namespace verwenden wir *com.example.hello* und als Namen für die App *helloworld.* Anschließend wechseln wir in das neu angelegte Verzeichnis.

Ist das Grundgerüst einer Cordova-App einmal angelegt, müssen Sie die gewünschten Zielplattformen hinzufügen, damit Cordova weiß, für welches System es die App später übersetzen soll. Ein Liste aller möglichen Plattformen finden Sie unter https://cordova.apache.org/docs/en/latest/guide/platforms/index.html.

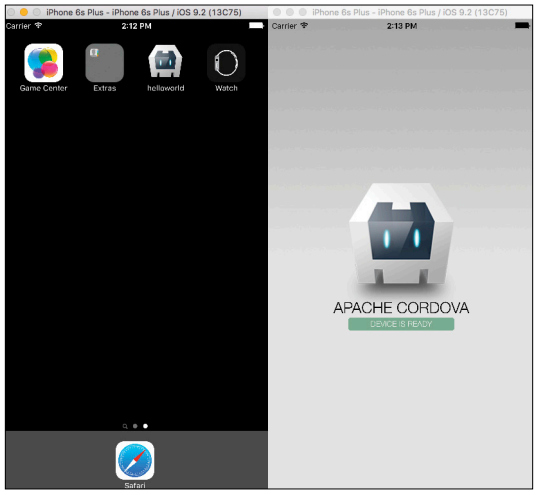
Wir wollen zunächst zwei Plattformen hinzufügen: einerseits iOS und andererseits den Browser an sich. Dies ist über den Befehl *platform add* möglich. Über *platform list* können zudem alle verfügbaren und installierten Plattformen angezeigt und per *platform rm* auch wieder gelöscht werden:

```
$ cordova platform add browser
$ cordova platform add ios8
$ cordova platform ls
Installed platforms: browser
4.0.0, ios 4.0.1
Available platforms: amazon-
fireos, android, blackberry10,
firefoxos, osx, webos
```

Im nächsten Schritt wollen wir die App erzeugen lassen. Dafür verwenden wir den Befehl *build.* Ohne Parameter werden alle Plattformen generiert:

```
$ cordova build
```

Das Ergebnis lässt sich im Unterverzeichnis *platforms* betrachten. Hier gibt es ja je zugefügter Plattform ein eigenes Verzeichnis und nun, nach dem *build*-Befehl, ein Verzeichnis *build* mit dem Inhalt des erzeugten Packages.



Im Emulator: Die helloworld-App startet (Bild 3)

Um das Ergebnis im Emulator zu betrachten, kann man den *emulate*-Befehl verwenden. Nachfolgend wollen wir den iOS-Emulator mit der App starten, die von Cordova dorthin deployt wurde:

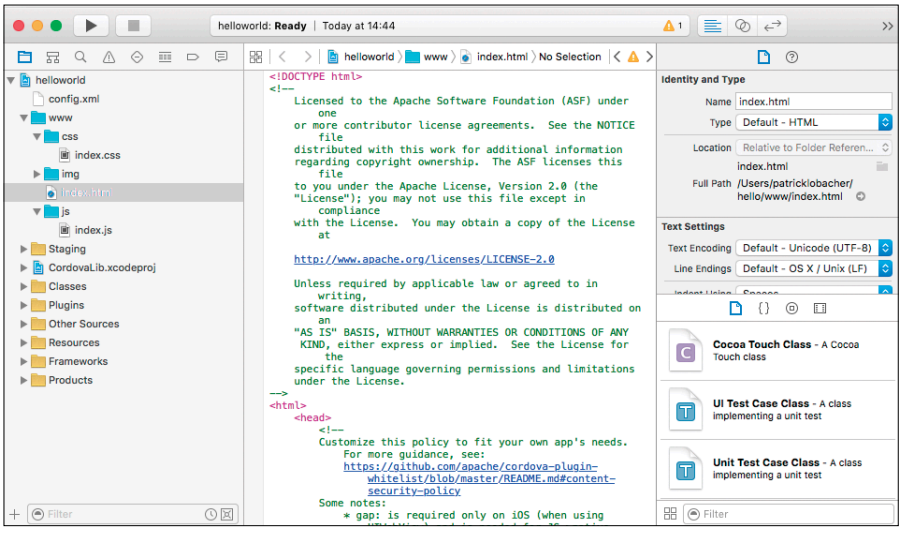
```
$ cordova emulate ios
```

Nach kurzer Wartezeit startet der Emulator mit der *helloworld*-App (Bild 3). Nun können Sie das Projekt auch direkt in Xcode öffnen. Dazu navigieren Sie zum Unterverzeichnis *platforms/ios.* Dort gibt es die Datei *helloworld.xcodeproj,* die sie per Doppelklick öffnen können. Daraufhin öffnet sich – vorausgesetzt, Sie haben dies vorher installiert – Xcode mit Ihrem Projekt (Bild 4).

Projektstruktur

Sehen wir uns jetzt die erhaltene Verzeichnisstruktur in einem Projekt etwas genauer an:

- **config.xml:** Dies ist die globale Konfigurationsdatei des Projekts. Im Schlüssel <*content*> wird zum Beispiel die Datei angegeben, die im Unterverzeichnis *www* als Startdatei aufgerufen wird – hier *index.html.*
- **hooks:** Es gibt eine Vielzahl von Ereignissen (zum Beispiel wenn die Batterie leer ist, wenn der Build-Prozess abgeschlossen ist et cetera). Skripts, die auf diese Hooks reagieren, befinden sich in diesem Verzeichnis.
- **plattforms:** Sämtliche Plattformen, auf denen der Code lauffähig sein soll, werden hier abgelegt.
- **plugins:** Cordova wird mit einem sehr schlanken Kern ausgeliefert – alle weitere Funktionen (zum Beispiel der Zu- ▶



Xcod: Das Projekt lässt sich auch direkt in Xcode öffnen (Bild 4)

griff auf die Hardware des Geräts) wird in Plug-ins ausgelagert, die man in diesen Verzeichnis installieren kann.

- **www:** Hier befindet sich die Applikation an sich. Dort gibt es die Startdatei *index.html* und die Verzeichnisse *css* (für die Stylesheets), *js* (für die JavaScripts) und *img* (für die Bilder).

Als allgemeiner Startpunkt dient die Datei *index.html*. Sehen wir uns diese einmal genauer an (Listing 1).

Im Header wird hier eine entsprechende CSS-Datei eingebunden und am Ende des Bodys zwei JavaScript-Dateien. Dabei existiert *cordova.js* scheinbar gar nicht – zumindest nicht im *www*-Verzeichnis. Hier ist wichtig zu wissen, dass es für jede Plattform eine eigene, speziell angepasste Cordova-JavaScript-Basis-Bibliothek gibt, die also erst im Verzeichnis *platforms/[platform]/www* auftaucht.

Das navigator-Objekt

Bei einer Cordova-App stehen Ihnen innerhalb von JavaScript die meisten Methoden zur Verfügung, die sich auch beim Programmieren einer normalen Webseite verwenden lassen.

So können Sie beispielsweise die DOM-Objekte verwenden und dabei manchmal auf spezifische Cordova-Plug-ins verzichten. Legen wir hierzu eine Datei *js/custom.js* mit dem folgenden Inhalt an (Bild 5):

```
function navigatorobj() {
  document.getElementById
  ('content').innerHTML =
  '';
  for (i in navigator) {
    document.getElement
    ById ('content').
    innerHTML+= i + ': '
    + navigator[i] +
    '<br />';
  }
}
```

JavaScript: Bei einer Cordova-App stehen die meisten JavaScript-Methoden zur Verfügung (Bild 5)

Der Aufruf der Methode wiederum erfolgt in der Datei *js/index.js:*

```
var app = {
  ...
  receivedEvent: function(id) {
    ...
    console.log('Received Event: ' + id);
    // Eigener Code ab hier
    navigatorobj();
  }
};
```

Nun müssen wir noch die Datei *index.html* anpassen:

```
<div id="deviceready">
  <p class="event listening">Connecting to Device</p>
  <p class="event received" id="content">
    <!-- Eigener Code -->
  </p>
</div>
```

Im Nachfolgenden wollen wir mit einem Plug-in von Cordova arbeiten, um beispielsweise die Geolokalisierung anzusprechen.

Geolokalisierung

Die Plug-ins finden Sie auf der Cordova-Seite unter dem Link https://cordova.apache.org/plugins.

Geben Sie dort in die Suchmaske ganz oben *geolocation* ein und klicken Sie auf den Link zu *cordova-plugin-geolocation*. Dort finden Sie auch einen Hinweis zur Installation. Die-

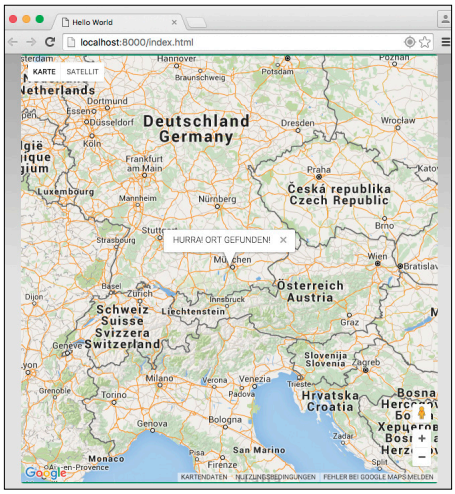
Listing 1: Startdatei index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Security-Policy"
    content="default-src 'self' data: gap: https://
    ssl.gstatic.com 'unsafe-eval'; style-src 'self'
    'unsafe-inline'; media-src *">
    <meta name="format-detection"
    content="telephone=no">
    <meta name="msapplication-tap-highlight"
    content="no">
    <meta name="viewport" content="user-scalable=no,
    initial-scale=1, maximum-scale=1,
    minimum-scale=1, width=device-width">
    <link rel="stylesheet" type="text/css"
    href="css/index.css">
    <title>Hello World</title>
  </head>

  <body>
    <div>
      <div id="deviceready" class="blink">
        <p class="event listening">Connecting to
        Device</p>
        <p class="event received">Device is Ready</p>
      </div>
    </div>
    <script type="text/javascript" src="cordova.js">
    </script>
    <script type="text/javascript"
    src="js/index.js"></script>
  </body>
</html>
```

se wird immer über den Befehl *plugin add* durchgeführt:

```
$ cordova plugin add
cordova-plugin-geolocation
```

Wir brauchen noch entsprechenden Code in der Datei *custom.js* (Listing 2). In der Datei *js/index.js* muss zudem noch der Aufruf von *navigatorobj()* durch *geo()* ersetzt werden. Die Funktion *geo()* überprüft zunächst, ob es das Objekt *navigator.geolocation* gibt. Ist dies der Fall, werden die Geokoordinaten über *getCurrentPosition()* ermittelt. Im Fehlerfall wird die Fehlerfunktion *geoError()* aufgerufen, bei Erfolg *geoSuccess()*. Hier werden alle verfügbaren Daten ausgelesen und angezeigt.



Google Maps: Die aktuelle Position auf einer Google Map anzeigen (Bild 6)

Jetzt, da wir die Geokoordinaten ermitteln können, ist es sinnvoll, diese dazu zu benutzen, um die aktuelle Position auf einer Google Map anzuzeigen. Hierfür benötigen wir wieder eine Anpassung der Datei *custom.js* (Listing 3).

Und in der *index.html* müssen wir nun noch das Google-Maps-API einbinden (Bild 6).

Kontakte ansprechen

Als weiteres Beispiel sehen wir uns an, wie wir so mit der Kontakte-Applikation auf dem Smartphone kommunizieren können, dass es uns möglich ist, einen neuen Kontakt dort anzulegen.

Das zugehörige Plug-in ist *cordova-plugin-contacts,* das wir analog dem vorigen Beispiel über die Kommandozeile installieren. In der Datei *index.html* platzieren wir ein Formular:

```
...
<div class="event received" id="content">
  <!-- Eigener Code -->
  <h4>Kontakt zufuegen:</h4>
  <form id="adresse">
    <label>ID</label>
    <input type="text" id="id" />
    <br />
    <label>Anzeigename</label>
    <input type="text" id="name" />
    <br />
  </form>
  <button id="create">
    Kontakt erstellen!
```

Listing 2: custom.js für Geolokalisierung

```
function geo() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition
    (geoSuccess, geoError);
  } else {
    document.getElementById('content').innerHTML =
    "Kein Geolocation möglich!";
  }
}
function geoSuccess(position) {
  document.getElementById('content').innerHTML =
    'Inhalt navigator.geolocation:<hr />' +
    'Breitengrad: ' + position.coords.latitude +
    '<br />L&auml;ngengrad: ' +
    position.coords.longitude +
    '<br />&Uuml;bergangsh&ouml;he: ' +
    position.coords.altitude +
    '<br />' + 'Genauigkeit: ' +
    position.coords.accuracy +
    '<br />' + 'Genauigkeit &Uuml;bergangsh&ouml;
    he: ' + position.coords.altitudeAccuracy +
    '<br />' + 'Steuerkurs: ' +
    position.coords.heading + '<br />' +
    'Geschwindigkeit: ' + position.coords.speed +
    '<br />' +
    'Zeitstempel: ' + position.timestamp;
}
function geoError(msg) {
  document.getElementById('content').innerHTML =
  "Fehler bei der Lokalisierung:<br /> " + msg;
}
```

Listing 3: custom.js für Google Maps

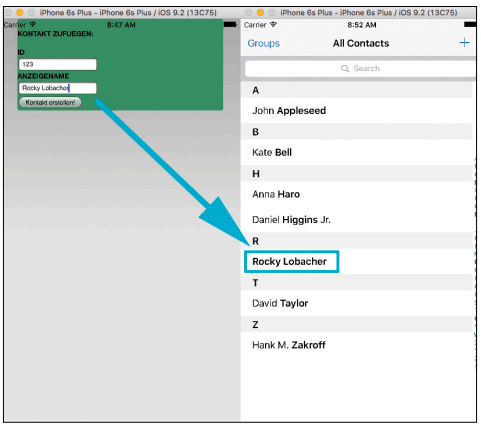
```
...
function geoSuccess(position) {
  var map = new google.maps.Map
  (document.getElementById('map'), {
    center: {lat: -34.397, lng: 150.644},
    zoom: 6
  });
  var infoWindow = new google.maps.InfoWindow
  ({map: map});

  var pos = {
    lat: position.coords.latitude,
    lng: position.coords.longitude
  };
  infoWindow.setPosition(pos);
  infoWindow.setContent('Hurra! Ort gefunden!');
  map.setCenter(pos);
}
...
```

```
  </button>
</div>
...
```

Nun benötigen wir noch entsprechendes JavaScript, mit dem wir das Kontakt-Objekt ansprechen können. Dies wird wie bisher in der Datei *custom.js* eingebracht:

```
function contacts() {
  document.getElementById
  ("create").
  addEventListener('click',
  function() {
    var myContact = navigator.
    contacts.create({
      "id":document.getElementById("id").value,
      "displayName": document.getElementById("name").
      value});
      myContact.save();
  });
}
```

Wir haben lediglich eine *id* (Identifier) und den *displayName* (angezeigter Name) abgefragt und verarbeitet. Natürlich können Sie auch alle anderen Felder verwenden. Einen Überblick über die in diesem Zusammenhang möglichen Felder finden Sie unter https://www.npmjs.com/package/cordova-plugin-contacts.



Kontakte: Auch die Kontakte aus dem Device lassen sich ansprechen (Bild 7)

Schließlich müssen Sie noch den Aufruf *contacts();* in die Datei *index.js* einfügen (Bild 7).

Icon und Splashscreen

Bislang startet die App immer mit dem Cordova-Splashscreen und dem Cordova-Logo. Dies wollen wir nun ändern. Dazu legen Sie innerhalb des Verzeichnisses *www* die Verzeichnisse *res/ios/* an. Dorthinein werden die Icons (für iOS) gelegt.

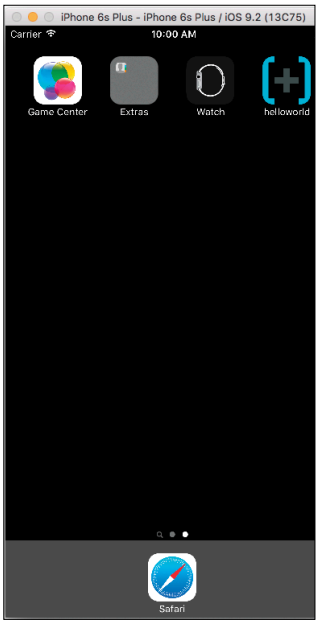
Zuständig für für die Einstellungen von Icons und Splashscreen ist die Datei *config.xml.* Hier kann man den Abschnitt aus Listing 4 einbringen, um die Icons zu definieren. Die Erkennung des richtigen Icons erfolgt aufgrund der Display-Größe des Geräts. Den Dateinamen können Sie ebenfalls aus der Liste entnehmen.

Als Beispiel haben wir auf einem iPhone 6s Plus ein neues Icon *icon-60@3x.png* in den Pfad *www/res/ios/* gelegt. Adäquate Anweisungen für die anderen Plattformen finden Sie unter https://cordova.apache.org/docs/en/latest/config_ref/images.html (Bild 8). Der Austausch des Splashscreens erfolgt analog (Listing 5). Hier legen wir einen eigenen Splashscreen für ein iPhone 6s Plus im Verzeichnis *www/res/screen/ios/* unter dem Namen *Default-736h.png* ab (Bild 9).

Um die App wirklich vollumfänglich testen zu können, ist es wichtig, dies nicht nur im Emulator zu tun, sondern auf ei-

Listing 4: Icons

```
<platform name="ios">
  <!-- iOS 8.0+ -->
  <!-- iPhone 6 Plus  -->
  <icon src="www/res/ios/icon-60@3x.png" width="180"
  height="180" />
  <!-- iOS 7.0+ -->
  <!-- iPhone / iPod Touch  -->
  <icon src="www/res/ios/icon-60.png" width="60"
  height="60" />
  <icon src="www/res/ios/icon-60@2x.png" width="120"
  height="120" />
  <!-- iPad -->
  <icon src="www/res/ios/icon-76.png" width="76"
  height="76" />
  <icon src="www/res/ios/icon-76@2x.png" width="152"
  height="152" />
  <!-- iOS 6.1 -->
  <!-- Spotlight Icon -->
  <icon src="www/res/ios/icon-40.png" width="40"
  height="40" />
  <icon src="www/res/ios/icon-40@2x.png" width="80"
  height="80" />
  <!-- iPhone / iPod Touch -->

  <icon src="www/res/ios/icon.png" width="57"
  height="57" />
  <icon src="www/res/ios/icon@2x.png" width="114"
  height="114" />
  <!-- iPad -->
  <icon src="www/res/ios/icon-72.png" width="72"
  height="72" />
  <icon src="www/res/ios/icon-72@2x.png" width="144"
  height="144" />
  <!-- iPhone Spotlight and Settings Icon -->
  <icon src="www/res/ios/icon-small.png" width="29"
  height="29" />
  <icon src="www/res/ios/icon-small@2x.png" width="58"
  height="58" />
  <!-- iPad Spotlight and Settings Icon -->
  <icon src="www/res/ios/icon-50.png" width="50"
  height="50" />
  <icon src="www/res/ios/icon-50@2x.png" width="100"
  height="100" />
</platform>
```



Icon: Apps lassen sich mit individuellen Icons ausstatten (Bild 8)

Splashscreen: Auch individuelle Splashscreens sind möglich (Bild 9)

nem physikalischen Gerät. Für Apple/iOS müssen Sie zunächst Mitglied im Apple Developer Program werden: https://developer.apple.com/programs. Dazu klicken Sie rechts oben auf *Enroll* und wählen dann aus, ob Sie ein einzelner Entwickler oder eine Organisation sind. Die Kosten betragen hierbei 99 Euro pro Jahr.

Fügen Sie anschließend mit Hilfe des Unterpunkts *Devices* Ihre Geräte hinzu, auf denen Sie die App später testen wollen. Dazu müssen Sie entweder jedes der Geräte kurz per USB-Kabel an Ihren Rechner anschließen oder den Identifier (UDID) eingeben.

Als Nächstes benötigen Sie eine AppID. Wählen Sie hier *hello* als *Product Name* und *com.example* als *Company Identifier*. Nun benötigen Sie ein iOS Provision Profile, das Sie unter https://developer.apple.com/account/overview.action erstellen können.

Hierfür gibt es einen sogenannten Development Provisioning Assistant. Wählen Sie zunächst *Ad Hoc* als Distributions-Art. Sobald das Provision-Profil erstellt ist, können Sie dieses herunterladen und anschließend per Doppelklick auf Ihrem Rechner installieren

Als letzten Schritt müssen Sie in Xcode dafür sorgen, dass die App das Provision-Profil verwendet. Dafür gehen Sie in den Organizer und verknüpfen die App mit dem Profil.

Eine gute Anleitung hierfür finden Sie unter http://codewithchris.com/deploy-your-app-on-an-iphone. Für die anderen Betriebssysteme können Sie sich hier orientieren: https://cordova.apache.org/docs/en/3.3.0/guide/platforms.

Listing 5: Splashscreen

```
<platform name="ios">
  <!-- images are determined by width and height.
  The following are supported -->
  <splash src="www/res/screen/ios/Default~iphone.
  png" width="320" height="480"/>
  <splash src="www/res/screen/ios/Default@2x~iphone.
  png" width="640" height="960"/>
  <splash src="www/res/screen/ios/Default-
  Portrait~ipad.png" width="768" height="1024"/>
  <splash src="www/res/screen/ios/Default-
  Portrait@2x~ipad.png" width="1536" height="2048"/>
  <splash src="www/res/screen/ios/Default-
  Landscape~ipad.png" width="1024" height="768"/>
  <splash src="www/res/screen/ios/Default-Landscape
  @2x~ipad.png" width="2048" height="1536"/>
  <splash src="www/res/screen/ios/Default-
  568h@2x~iphone.png" width="640" height="1136"/>
  <splash src="www/res/screen/ios/Default-667h.png"
  width="750" height="1334"/>
  <splash src="www/es/screen/ios/Default-736h.png"
  width="1242" height="2208"/>
  <splash src="www/res/screen/ios/Default-Landscape-
  736h.png" width="2208" height="1242"/>
</platform>
```

Links zum Thema

- Apache-Cordova-Website
  https://cordova.apache.org
- Apache-Cordova-Tutorial
  https://ccoenraets.github.io/cordova-tutorial
- Video2brain über Apache Cordova
  https://www.video2brain.com/de/videotraining/mobile-apps-mit-cordova
- PhoneGap
  http://phonegap.com

Fazit

Nun kann es losgehen mit den übrigen Funktionen eines Smartphones: Kompass, Beschleunigungssensor, Kamera, Batteriestatus, Videos, Audio, Browser und viele weitere mehr. Und das stets plattformunabhängig und am Ende als native App. ■

Patrick Lobacher
ist Digital-Native, Entwickler, Berater, Coach und Autor zahlreicher Fachbücher und Fachartikel. Er ist Vorstandsvorsitzender der Pluswerk AG, die digitale Kommunikationslösungen konzipiert, umsetzt und betreut.

XTEXT, XTEND: DOMAIN-SPECIFIC LANGUAGE FÜR C++ / QT (TEIL 5)

Cross-Plattform-Generator mobaDSL

Die DSL bietet Vorteile bei der Entwicklung von mobilen Apps für mehrere Plattformen.

In der vorangegangenen Folge habe ich einen groben Überblick über mobaDSL gegeben und gezeigt, was in den Xtext/Xtend-basierten Modellen so alles modelliert werden kann. Bevor wir uns der Generierung im Detail zuwenden, möchte ich in diesem Artikel auf einen wichtigen Aspekt eingehen: die Vorteile einer DSL bei der Entwicklung von mobilen Businessanwendungen für mehrere Plattformen.

In den ersten drei Folgen haben wir ja schon am Beispiel von BlackBerry-10-Apps gesehen, was mit einer einfachen DTO-DSL möglich ist. mobaDSL geht darüber hinaus und generiert nicht nur DTOs, sondern auch Entities, den modularen Aufbau von Apps (NFC, Bluetooth, Push, Headless et cetera), Aufrufe von REST-Services und die Verwaltung von Offline-Persistierung (JSON, SQLite).

Keine Apps im eigentlichen Sinne

Eine Klarstellung ist hier wichtig: Es werden keine Apps im eigentlichen Sinne generiert, sondern nur die gesamten Patterns und alles andere, was sonst mühsam per Copy and Paste aus anderen Apps kopiert wird. mobaDSL ist eine Toolbox, die das Entwickeln komplexer mobiler Apps von Ballast befreit, sodass man sich auf die eigentliche Architektur und die Gestaltung der Oberfläche konzentrieren kann.

Es gibt viele Projekte und Frameworks, die beim Entwickeln von Cross-Plattform-Anwendungen zum Einsatz kommen, und jedes hat seine Fans und seine Vor- oder Nachteile. In einem anderen Artikel außerhalb dieser Serie werde ich darauf noch einmal genauer eingehen – jetzt möchte ich aber erläutern, warum ich mich für Qt als erste weitere Zielplattform neben BlackBerry 10 entschieden habe. Dann sehen wir uns an, wie man Qt installiert, um damit für Android und iOS Apps zu schreiben, und zu guter Letzt erkläre ich, wie mobaDSL installiert wird. Danach geht es dann in den nächsten Artikeln direkt in den Generator, die Xtend-Templates und den für mehrere Plattformen generierten Code.

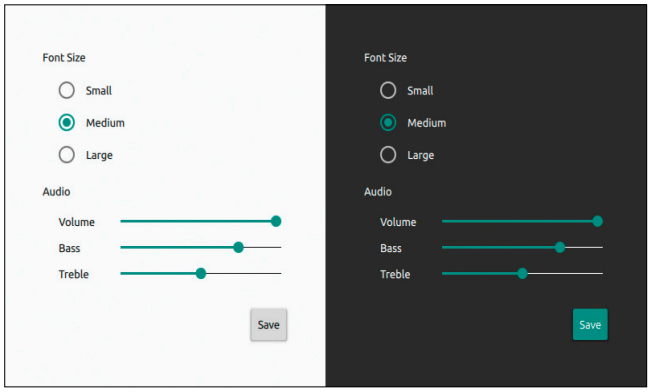
Qt 5.6/5.7 – ein solides Fundament

Qt verspricht schon seit längerer Zeit (seit Qt 5.0), dass damit auch Apps für Android und iOS erzeugt werden können. Aus meiner Sicht sprachen aber bisher einige schwerwiegende Dinge dagegen.

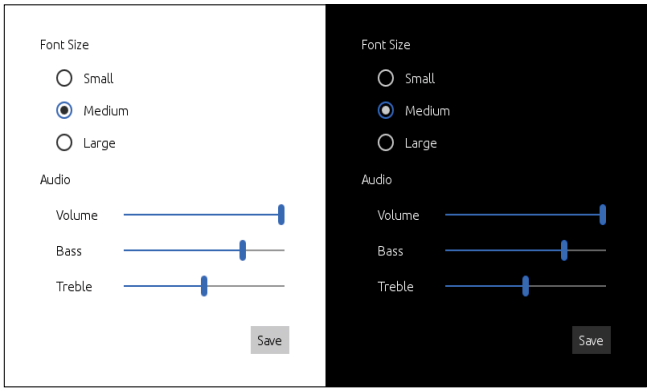
Es gab keinerlei Unterstützung für die Darstellung auf Geräten mit unterschiedlicher Density, was aber grundlegend für heutige mobile Apps ist und von Android, iOS und auch BlackBerry 10 schon seit längerer Zeit unterstützt wird. Ich hatte wirklich keine Lust, das alles selbst skalieren zu müssen.

Der zweite große Kritikpunkt waren die fehlenden UI-Elemente, um die Oberfläche mobiler Apps zu gestalten. Es gab zwar mit den QtQuick-Controls elementare UI-Controls, aber die sahen doch eher nach Desktop-Controls als nach moderner mobiler Oberfläche aus. Außerdem fehlten grundlegende Controls und Animationen, um durch eine mobile App navigieren zu können.

Als Drittes sprach die Lizenzierung mit einem hohen monatlichen Lizenzbetrag (300 Euro) für einen Entwickler wie



Material Design ist angelehnt an das Design-Prinzip von Googles Material Design (Bild 1)



Universal Design ist angelehnt an das Design-Prinzip von Microsofts Universal Design (Bild 2)

mich gegen den Einsatz von Qt als Framework. Das ist jetzt aber in Bewegung gekommen: Qt hat erkannt, dass es dort einen günstigen Einstieg geben muss. Mit dem Erscheinen von Qt 5.6 hat sich alles grundlegend geändert.

High-Density-Support in Qt 5.6

Qt kommt ja in vielen Umgebungen zum Einsatz: Desktop (OS X, Windows, Linux), Embedded, Mobil (Android, iOS, Windows 10), QNX, Smart TVs und mehr. Nicht überall benötigt man die Unterstützung von unterschiedlicher Density wie in mobilen Apps. Darum ist dies in den Apps beim Start der Anwendung jeweils durch einen einfachen Schalter individuell konfigurierbar:

```
QGuiApplication::setAttribute
(Qt::AA_EnableHighDpiScaling);
```

Dieser Schalter führt zu einer automatischen Anpassung von Texten, UI-Controls und Positionierungen auf dem Bildschirm. Es können hier aber weitere Konfigurationsparameter gesetzt werden.

High Density wird von den Qt-Quick-Controls und -Layouts unterstützt, nicht aber von den Qt-Widgets, die vielfach im Desktop genutzt werden. In den meisten Fällen kann man sich auf die automatische dynamische Skalierung verlassen. Wer möchte, kann da aber eingreifen und eigene Logiken – abhängig von der jeweiligen Bildschirm-Density – implementieren:

```
Image {
  source: "artwork.png"
}
```

Dieses Image würde jetzt bei einem Retina-Display automatisch versuchen, das *artwork@2x.png* darzustellen. Da kann man aber eingreifen:

```
Image {
  source: {
    if (Screen.PixelDensity < 40)
      "image_low_dpi.png"
    else if (Screen.PixelDensity > 300)
      "image_high_dpi.png"
    else
      "image.png"
  }
}
```

Die ersten High-DPI-Implementierungen von Qt 5 betrafen nur OS X und Windows – mit Qt 5.6 werden die mobilen Varianten ebenfalls unterstützt.

UI-Controls für mobile Apps in Qt 5.6

Die fehlenden UI-Controls waren der Haupt-Hinderungsgrund für mich, da ich wirklich nicht alles selbst nachbauen wollte, was man so in einer mobilen App an Controls und für die Navigation benötigt.

Listing 1: SwipeView und PageIndicator

```
SwipeView {
  id: view
  currentIndex: 1
  anchors.fill: parent
  Item {
    id: firstPage
  }
  Item {
    id: secondPage
  }
  Item {
    id: thirdPage
  }
}
PageIndicator {
  id: indicator

  count: view.count
  currentIndex: view.currentIndex

  anchors.bottom: view.bottom
  anchors.horizontalCenter:
  parent.horizontalCenter
}
```

Das hat sich jetzt in Qt 5.6 durch die *labs.controls* geändert. Achtung: Diese Controls sind noch als technical Preview deklariert – das API kann sich also bis Qt 5.7 noch ändern. Es gibt nun neben einem Default Style noch zwei spezielle Styles für mobile Apps:

- Material Design, angelehnt an Googles Material Design,
- Universal Design, angelehnt an Microsofts Universal Design.

Bild 1 im Material Design und Bild 2 im Universal Design zeigen den Unterschied. Wie man sieht, werden auch *dark*- oder *light*-Themes unterstützt. Diese Controls sind keine nativen Controls, die aus dem darunterliegenden OS kommen, sondern Qt-Controls, die in C++/QML eingesetzt werden und über die Plattformen hinweg ein identisches Aussehen versprechen.

Neutrales Qt-Styling

Neben diesen beiden Designs gibt es dann noch ein neutrales Qt-Styling – *default* genannt. Gerade im Businessumfeld ist es ein sinnvolles Feature, unternehmensinterne Apps immer identisch aussehen zu lassen, egal ob sie auf Android oder iOS oder Windows laufen.

Da kann es durchaus gewünscht sein, beispielsweise alle unternehmensinternen Apps im Material Style zu entwickeln, unabhängig davon, ob sie später auf Android-, iOS- oder Windows-Geräten laufen. Der Helpdesk hat es dann einfacher beim Support der User, und die Aufwendungen für Dokumentation sind geringer. ▶

Bei normalen Consumer-Apps wird man dagegen versuchen, Apps der Plattform anzupassen, also Material für Android und Universal für Windows zu nutzen. Was es derzeit bei Qt 5.6 noch nicht gibt, ist ein spezifischer iOS-Style.

Um zwischen den Styles zu wechseln, genügt wiederum ein einfacher Schalter:

```
QT_LABS_CONTROLS_STYLE=universal
```

Und schon wird ein Button im entsprechenden Style dargestellt – da muss man sich um nichts mehr kümmern:

```
import Qt.labs.controls 1.0
ApplicationWindow {
  visible: true
  Button {
    text: "Hello World!"
  }
}
```

Das *ApplicationWindow* als *root*-Element hat Kenntnis über Bildschirmgröße, Density und Skalierungen. Die Designs lassen sich auch mit Akzentfarben et cetera konfigurieren:

```
Button {
  text: "Stop"
  highlighted: true
  Material.accent: Material.Red
  Material.theme: Material.Dark
}
```

Neben den UI-Controls bietet Qt 5.6 aber auch alles, um sich durch eine App zu navigieren:

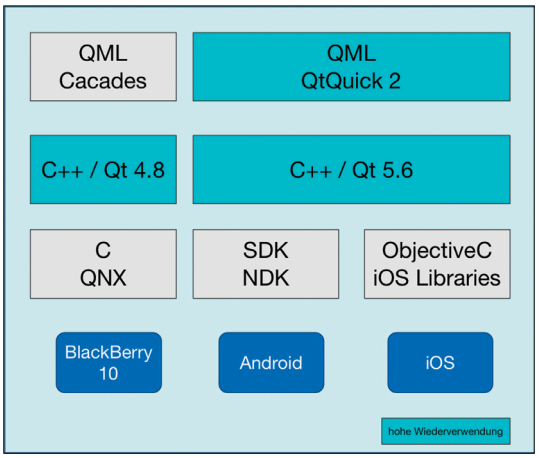
- Drawer,
- Tab Bar,
- Swipe Views,
- Stack Views.

Listing 1 zeigt ein Beispiel, wie die *SwipeView* und ein *PageIndicator* in QML beschrieben werden. Auf alles kann aber auch aus C++ zugegriffen werden – das ist ebenfalls neu. Vorher wurden UI-Controls nur in QML definiert. So weit ein kurzer Blick auf die neuen Controls.

Die Lizenzierung ist bei Qt nicht ganz einfach zu verstehen, wenn man sich das erste Mal damit beschäftigt. Es gibt zwei grundlegende Varianten:

- Open-Source-Lizenz (kostenlos),
- kommerzielle Lizenz (kostenpflichtig).

Qt unterscheidet dabei nicht nach den Plattformen, auf denen es eingesetzt wird. Eine Lizenz gilt immer für alles. Und das ist das Problem: Aufgrund der Vielzahl an Plattformen ist die kommerzielle Version mit 300 Euro monatlichen Lizenzkosten nicht gerade ideal, wenn man vielleicht nur 99-Cent-Apps baut.



Wiederverwendung von Code über Plattformgrenzen hinweg (Bild 3)

Neue Lizenz angekündigt

In einem neuen Blog (siehe Linkliste) wird aber eine neue Lizenz für kleine Start-ups und Entwickler angekündigt, was dann speziell in der Entwicklung mobiler Apps von Vorteil ist.

Die Open-Source-Version (LGPL2 und LGPL3) erfordert nicht, dass die eigene App ebenfalls Open Source ist – kommerzielle Nutzung ist kein Problem. Die Lizenz erfordert aber, dass der Anwender die enthaltenen Qt-Libraries austauschen kann, wenn er das möchte. Das mag für den Embedded-Einsatz passen, aber nicht, wenn Apps über Google Play oder den Apple Store vermarktet werden.

Es gibt zwar für Android eine Dokumentation, wie man die Open-Source-Anforderungen einhält, aber irgendwie passt das aus meiner Sicht dennoch nicht so richtig. Bei Apple ist es zudem nicht möglich, Apps mit Libraries, die dynamisch eingebunden werden, in den Store zu stellen. Daher wird die neue, günstigere Lizenz sicher dazu beitragen, dass weitere Entwickler sich einmal anschauen werden, wie man mit Qt mobile Apps baut.

Bei Businessanwendungen im Enterprise-Bereich ist es kein Problem, mit der Open-Source-Variante zu arbeiten, da der Kunde ja ohnehin die volle Kontrolle über die App hat. Zu den genauen Details der neuen Lizenzierung gibt ein Blog Auskunft (siehe Linkliste)

Vorteile von Qt für mobile Apps

Kommen wir jetzt zu den Gründen, warum ich mir von Qt 5.6 Vorteile verspreche. Wer meinen Artikeln folgt, der weiß, dass ich vorrangig native Apps für BlackBerry 10 entwickle. BlackBerry 10 setzt auf Qt 4.8 auf und hat das durch ein eigenes UI-Framework (Cascades) erweitert, das mit QML definiert wird, wobei die UI-Controls aber speziell für BlackBerry 10 entwickelt wurden und nichts mit den UI-Controls von Qt zu tun haben – selbst wenn in beiden Fällen QML benutzt wird, um die Oberfläche zu beschreiben.

Qt 5 ist eine Weiterentwicklung von Qt 4.8 – es sind aber viele Aspekte sehr ähnlich – sei es, dass es um HTTP-Requests geht oder um QObject-Datenobjekte und um das Event Handling (Signals und Slots).

In diesem Bereich der Businesslogik und anderem, was in C++/ Qt entwickelt wurde, werde ich etwa 80 Prozent wiederverwenden können. Das UI ist allerdings zwischen BlackBerry 10 und Qt for Mobile (Android, iOS, Windows) unterschiedlich.

Wenn ich mir dann aber das UI für Qt 5.6 ansehe, ist es für alle weiteren Plattformen zu 80 bis 90 Prozent identisch.

Ich kann also zwischen BlackBerry 10 und Qt 5.6 vieles in C++/Qt wiederverwenden, und ich kann im UI zwischen Android, iOS und Windows vieles gemeinsam nutzen. In der Summe betrachtet, habe ich kein anderes Framework gefunden, das mir das erlaubt – jedenfalls, wenn ich nicht den HTML5/Hybrid-Weg gehen möchte.

Qt ist von der Performance her für mich vergleichbar mit einer nativen App, da Qt ja direkt auf dem darunterliegenden OS (Android, iOS, Windows) aufsetzt. Mit den neuen UI Controls erhalte ich zudem noch ein plattformnahes UI und UX.

Folgendes ist noch zu beachten: Sollte es bestimmte Funktionalitäten nicht in den Qt-Libraries geben oder es sich um ganz spezielle Features eines mobilen OS handeln, dann kann dies beliebig gemixt werden: Ich kann beispielsweise auch aus Qt/C++ heraus direkt Methoden in Android Java aufrufen, wenn ich das möchte. Auch bei iOS ist gemeinsame Nutzung von C++- und Objective-C-Code möglich. Bild 3 gibt einen Überblick über die Wiederverwendung von Code über Plattformgrenzen hinweg.

Qt 5.6 installieren

Zum Zeitpunkt der Erstellung des Artikels ist die Qt 5.6 Beta verfügbar – die finale Version wird in Kürze erwartet. Um Qt für Android zu installieren, benötigt man:

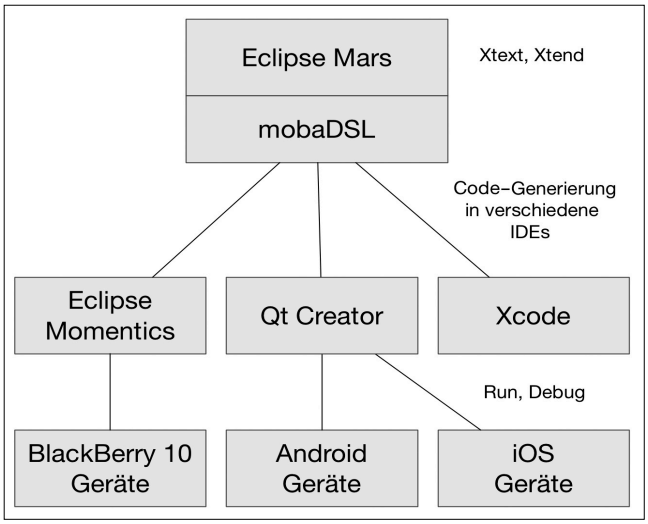
- Android SDK,
- Android NDK,
- ANT,
- Java JDK 6 oder höher.

Es wird also nicht explizit eine Android IDE benötigt – sie ist aber sicher sinnvoll.

Um Qt für iOS zu installieren, muss Xcode in der neuesten Version vorhanden sein. Eine detaillierte Anleitung findet sich in der Linkliste zum Artikel. Das Deployment auf die entsprechenden Geräte erfolgt direkt aus der Qt IDE (Qt Creator), die man zusammen mit dem Qt-5.6-Download erhält.

Ich habe die Installation von Qt 5.6 Beta für Android und iOS gemäß Anleitungen durchgeführt und konnte nach wenigen Minuten direkt aus dem Qt Creator eine Beispiel-App auf das PRIV (BlackBerry Android Smartphone) deployen und ausführen.

mobaDSL benötigt Eclipse Mars – und dort die Edition für RCP und RAP. In der Eclipse IDE ist folgende Software-Update-Seite hinzuzufügen: http://lun.lunifera.org/downloads/p2/mobadsl. Damit sind wir dann gerüstet, um mobaDSL im Einsatz zu erleben. Bild 4 gibt einen Überblick über den Einsatz der verschiedenen IDEs.



Überblick über den Einsatz der verschiedenen IDEs (Bild 4)

Fazit

Mit den Zielplattformen BlackBerry 10 und Qt 5.6 werden Android, BlackBerry10, iOS und Windows 10 abgedeckt. mobaDSL ermöglicht es dann, aus einem Modell heraus Code für diese Zielplattformen zu generieren. ■

Links zum Thema

- mobaDSL bei GitHub
  http://github.com/florianpirchner/mobadsl
- mobaDSL-Dokumentation
  http://github.com/florianpirchner/mobadsl/tree/master/org.mobadsl.docu
- Qt-High-DPI-Dokumentation
  http://doc.qt.io/qt-5/highdpi.html
- Qt Labs Controls
  http://doc-snapshots.qt.io/qt5-5.6/qtlabscontrols-index.html
- Qt-Lizenzierung
  http://blog.qt.io/blog/2016/01/13/new-agreement-with-the-kde-free-qt-foundation/
- Qt Android installieren
  http://doc-snapshots.qt.io/qt5-5.6/androidgs.html
- Qt iOS installieren
  http://doc-snapshots.qt.io/qt5-5.6/ios-support.html#getting-started
- Eclipse Mars installieren
  www.eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/mars2-rc2

Ekkehard Gentz
ist Autor, Trainer und Speaker auf Konferenzen. Er entwickelt als Independent Software Architect mobile Anwendungen für internationale Kunden, ist BlackBerry Elite Member und bloggt unter:
http://ekkes-corner.org

AMAZON-ALTERNATIVEN ZU GOOGLE-DIENSTEN

Glücklich ohne Google

Entwickler von Android-Apps integrieren Google-Dienste oft ohne großes Nachdenken.

Entwickler von Android-Apps integrieren in der Regel Google-Dienste in ihre Applikationen. Dass für die meisten Angebote von Google Alternativen von Amazon zur Verfügung stehen, wird in der Praxis nur wenig berücksichtigt.

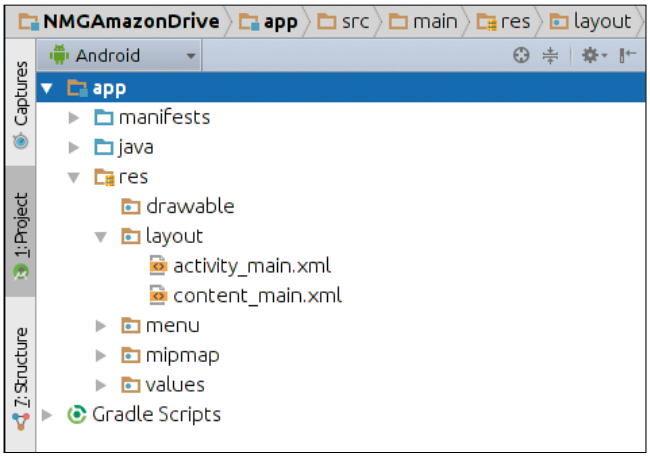
Das ist in mehrerlei Hinsicht schade. Erstens ist der Amazon Marketplace nach wie vor weniger überlaufen als der Play Store, und zweitens veranstaltet Amazon immer wieder Wettbewerbe, die teilnehmende Entwickler mit diversen Preisen und Auszeichnungen belohnen. Kurz gefasst: Android geht auch ohne Google. Dieser Artikel zeigt die Vorgehensweise bei der Nutzung von Amazon-Services.

Als Erstes sollten Sie sich dazu das notwendige SDK besorgen. Öffnen Sie https://developer.amazon.com/public/resources/development-tools/sdk in einem Browser Ihrer Wahl. Der große orangefarbene Button *Download SDK* liefert ein 111 MByte großes Archiv mit dem Gutteil der Amazon-Angebote. Extrahieren Sie es an einen gut zugänglichen Ort im Dateisystem Ihrer Workstation.

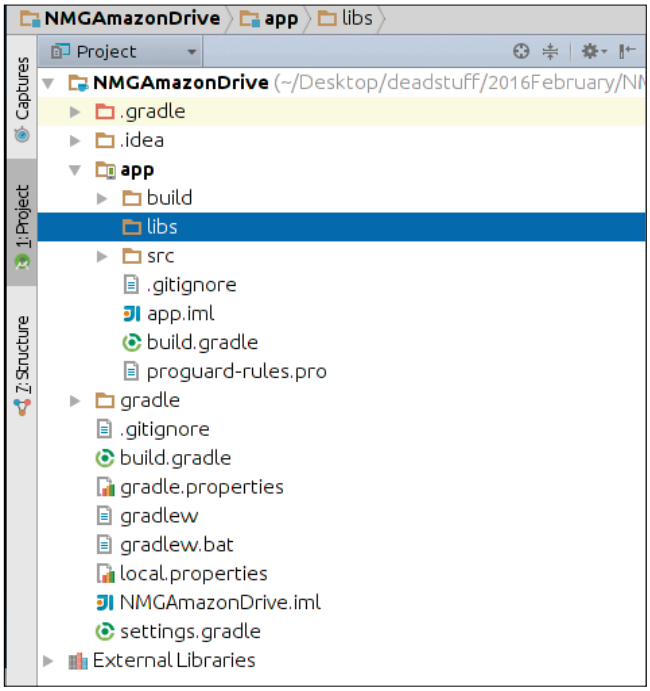
Erstellen Sie im nächsten Schritt ein neues Android-Projekt unter Verwendung einer aktuellen Version von Android Studio. Kopieren Sie die gewünschte Bibliothek im nächsten Schritt in das Unterverzeichnis */libs*. Leider ist das Verzeichnis von Haus aus nicht sichtbar. Bild 1 zeigt, wie die IDE in der Standardeinstellung aussieht.

Klicken Sie zur Lösung dieses Problems auf die in der Oberseite des Projektfensters eingeblendete Checkbox, und wählen Sie die Option *Project* aus. Android Studio reagiert daraufhin mit der Anzeige der kompletten Baumstruktur, in der das *libs*-Verzeichnis sichtbar ist (Bild 2).

Je nach Alter des verwendeten Projektgenerators müssen Sie die Datei *build.gradle* noch um eine Passage erweitern,



Das libs-Verzeichnis ist nicht sichtbar (Bild 1)



Bei der richtigen Einstellung wird das *libs*-Verzeichnis sichtbar (Bild 2)

die das Laden der im *libs*-Ordner liegenden *.jar*-Dateien anweist:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  testCompile 'junit:junit:4.12'
  compile 'com.android.support:appcompat-v7:23.1.1'
  compile 'com.android.support:design:23.1.1'
}
```

Als eine praktische Anwendung wollen wir Amazon Cloud Drive einsetzen. Der kostenlose Teil des Dienstes wurde im Jahr 2015 eingestellt. Wer seine Daten beim Buchhändler unterstellen möchte, muss einen Obolus entrichten. Amazons Bezahldienste sind vor allem deshalb interessant, weil sie kostenlose Uploads einer unbegrenzten Menge von Fotos erlauben. Wer sein Bilderarchiv in Sicherheit bringen möchte, muss sich um die Datenmenge keine Gedanken machen.

Daten in der Wolke

Nach der Erstellung eines Projektskeletts müssen die Dateien *Android/LoginWithAmazon/lib/login-with-amazon-sdk.jar* und *Android/AmazonCloudDrive/1.0.0/lib/amazonclouddri*

ve-1.0.0.jar in das */libs*-Verzeichnis wandern. Das Cloud-Drive-SDK setzt das Vorhandensein des Jackson-Core-Moduls voraus. Da die betreffenden Dateien im Gradle-Repository zur Verfügung stehen, genügt folgende Modifikation in *build.gradle*:

```
dependencies {
  compile 'org.codehaus.jackson:jackson-core-asl:1.9.9'
  compile fileTree(dir: 'libs', include: ['*.jar'])
  ...
}
```

Für das Login-Verfahren setzt Amazon voraus, dass eine bestimmte Activity vorhanden sein muss. Sie lässt sich durch das Einfügen eines vorgefertigten Snippets einpflegen. Achten Sie nur darauf, den Wert von *android:host* durch den kompletten Paketnamen der Applikation zu ersetzen (Listing 1).

Als letzte Vorbereitungshandlung müssen Sie die Manifestdatei um die folgenden beiden Permissions erweitern:

```
<uses-permission android:name=
"android.permission.INTERNET" />
<uses-permission android:name=
"android.permission.ACCESS_NETWORK_STATE" />
```

Für den Zugriff auf die im Cloud Drive liegenden Informationen muss sich der Benutzer bei Amazon anmelden und die zu verwendende Applikation autorisieren. Da unser Beispiel keine weite Verbreitung erreichen wird, wollen wir uns nicht mit dem Implementieren eines brandingkonformen Buttons aufhalten. Wer sein Programm an Endkunden verteilt, sollte die unter https://developer.amazon.com/public/apis/engage/login-with-amazon/content/button.html bereitstehenden

Listing 1: Acitvity

```
<activity
  android:name=
  "com.amazon.identity.auth.device.authorization.
  AuthorizationActivity"
  android:theme="@android:style/Theme.NoDisplay"
  android:allowTaskReparenting="true"
  android:launchMode="singleTask">
  <intent-filter>
    <action android:name=
    "android.intent.action.VIEW" />
    <category android:name=
    "android.intent.category.DEFAULT" />
    <category android:name=
    "android.intent.category.BROWSABLE" />
    <data
      android:host="com.tamoggemon.nmgamazondrive"
      android:scheme="amzn" />
  </intent-filter>
</activity>
```

Listing 2: Anfordern eines Zugriffstokens

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  try{
    myAuthManager = new AmazonAuthorizationManager
    (this, Bundle.EMPTY);
    myAuthManager.getToken(APP_AUTHORIZATION_SCOPES,
    new GetTokenListener());
  } catch (IllegalArgumentException e) {
    Log.e("NMG", "APIKey is incorrect or does not
    exist.", e);
    finish();
  }
```

Ressourcen verwenden, um Problemen mit Amazon vorzubeugen. Im Rahmen von *onCreate()* müssen wir eine Instanz der Klasse *AmazonAuthorizationManager* erstellen. Sie handhabt das eigentliche Anmelden bei der Amazon-Cloud. Der dazu notwendige Code beginnt mit der Deklaration der als Scopes bezeichneten Strings, die die gewünschten Zugriffsberechtigungen beschreiben:

```
public class MainActivity extends AppCompatActivity {

  AmazonAuthorizationManager myAuthManager;
  private static final String[]
  APP_AUTHORIZATION_SCOPES = {
    "clouddrive:read_all",
    ApplicationScope.CLOUDDRIVE_WRITE,
    "profile"};
```

OnCreate belebt die Membervariable mit einer Instanz des *AmazonAuthorizationManagers* und versucht bei dieser Gelegenheit gleich das Anfordern eines Zugriffstokens (Listing 2). *GetTokenListener* ist dabei eine Hilfsklasse, die vom Amazon-SDK bei Änderungen am Zustand des Tokens aufgerufen wird. Sie enthält drei Methoden, die unsere Applikation über Erfolg oder Scheitern des Authentifizierungsprozesses informieren. Unsere nächste Aufgabe soll dem Benutzer eine Möglichkeit zum Einloggen schaffen:

```
Button workBtn=(Button)findViewById(R.id.CmdLogin);
workBtn.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    myAuthManager.authorize(APP_AUTHORIZATION_SCOPES,
    Bundle.EMPTY, new LoginListener());
  }
});
```

Damit können wir uns der Realisierung der beiden Event Handler zuwenden, die wir aus Bequemlichkeitsgründen ▶

Listing 3: Event Handler

```
public class MainActivity extends AppCompatActivity {
  private class GetTokenListener implements
  AuthorizationListener {
    @Override
    public void onCancel(Bundle bundle) { }
    @Override
    public void onSuccess(Bundle bundle) {
      if (bundle.getString
      (AuthzConstants.BUNDLE_KEY.TOKEN.val) != null)
      {
        startDataAcquisition();
      }
    }
    @Override
    public void onError(AuthError authError) { }
  }
  private class LoginListener implements
  AuthorizationListener {
    @Override
    public void onCancel(Bundle bundle) { }
    @Override
    public void onSuccess(Bundle bundle) {
      startDataAcquisition();
    }
    @Override
    public void onError(AuthError authError) {}
  }
```

als innere Klassen anlegen. Der Korpus der beiden Methoden zeigt Listing 3.

Interaktion mit Serverdaten

Cloudspeicher verhalten sich nur in den seltensten Fällen wie normale Dateisysteme. Die meisten Anbieter nutzen eine aus dem Big-Data-Bereich abgeleitete Speicherstruktur. Amazon arbeitet mit Nodes: Die Daten des Nutzers werden in Form eines Baums abgebildet. Unser Programm reagiert auf das erfolgreiche Anmelden mit dem Aufruf der Methode *startDataAquisition*, die die Amazon-Server nach dem Inhalt des Wurzel-Nodes befragt:

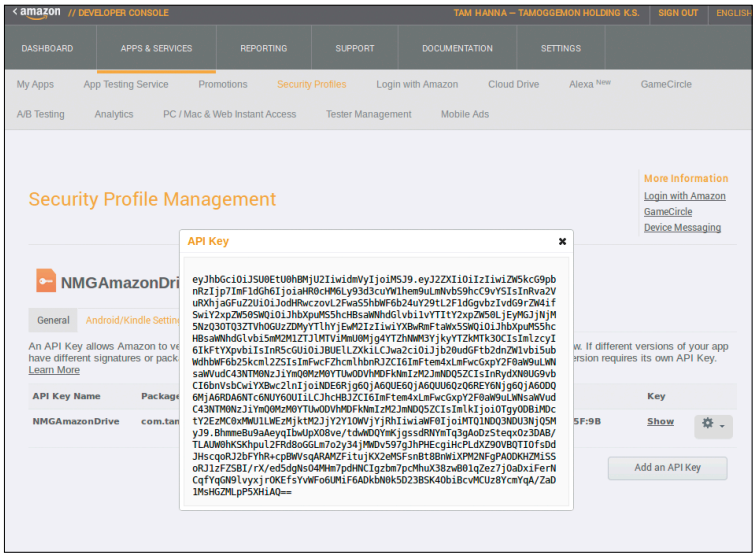
```
public void startDataAcquisition() {
  AmazonCloudDriveClient aClient = new
  AmazonCloudDriveClient(
    new AccountConfiguration(new
    AmazonAuthorizationConnectionFactory
    (myAuthManager, APP_AUTHORIZATION_SCOPES)),
    new ClientConfiguration("ExampleAgent/1.0")
 );
```

Zur Kommunikation zwischen Amazon-Server und Telefon ist eine Instanz der Klasse *AmazonCloudDriveClient* erforderlich. Sie nimmt die vom *AuthenticationManager* zurückgegebenen Tags zurück und nutzt sie zur Erzeugung von neuen, vertrauenswürdigen Verbindungen.

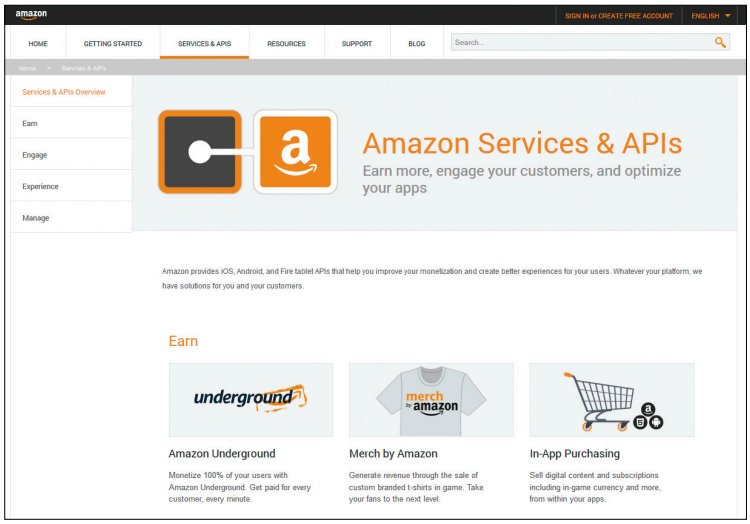
Listing 4: Request

```
String nextToken = null;
try {
  do {
    ListNodesRequest listNodesRequest =
    new ListNodesRequest();
    listNodesRequest.setStartToken(nextToken);
    ListNodesResponse response =
    aClient.listNodes(listNodesRequest);
    nextToken = response.getNextToken();
    List<Node> nodes = response.getData();
    for(Node e : nodes)
    {
      Log.v("NMG", e.getName() + " ist ein " +
      e.getKind());
    }
  }
  while (nextToken != null);
}
catch (Exception ex){}
}
```

Im nächsten Schritt folgt das Absetzen des eigentlichen Requests. Amazon bezeichnet den hier vorgeführten Prozess als *paged request*. Es handelt sich dabei um ein Kommando, das von Seiten des Servers aus Performance- und Effizienzgründen nicht in einem Durchlauf abgearbeitet werden kann (Listing 4). *setStartToken* setzt im Rahmen des ersten Durchlaufs den Mater-Node: nach dem Aufruf von *getNextToken* lädt die Klasse eine Gruppe von Nodes herunter, die über *getData* ab-



Das Amazon-Backend hat unserem Programm eine Signatur zugewiesen (Bild 3)



Amazon bietet Entwicklern ein vollständiges Ökosystem von Produkten an (Bild 4)

gerufen werden können. *GetNextToken()* liefert einen Steuer-Node weiter, der im nächsten Schritt an *setStartToken* übergeben wird – das Ende des Prozesses wird durch Retournieren von null angezeigt.

Freigabe einholen

Cloud-Anbieter streben in der Regel danach, einen genauen Überblick über die ihre Dienste nutzenden Applikationen zu behalten. Im Fall von Amazon erfolgt dies durch eine verpflichtende Registrierung, die zur Laufzeit vom SDK überprüft wird.

Loggen Sie sich zur Beschaffung dieser ID unter https://developer.amazon.com/iba-sp/overview.html mit Ihrem Entwicklerkonto ein. Klicken Sie anschließend auf den Button *Create a New Security Profile* und geben Sie die vom Assistenten abgefragten Informationen ein. Wechseln Sie danach in die Rubrik *Android, Kindle Settings*, woraufhin sich ein Dialog öffnet. Das Feld *API Key Name* wird normalerweise mit dem Namen der Applikation versehen. *Package* nimmt das Paket der Applikation auf – im Fall unseres Beispiels würde der String *com.tamoggemon.nmgamazondrive* lauten. Signature nimmt die Signatur des zum Signieren der Test-Builds verwendeten *keystore*-Datei entgegen.

Suchen Sie dazu das zum Signieren verwendete File *debug.keystore*. Auf der Maschine des Autors findet sich die Datei im versteckten Verzeichnis */home/tamhan/.android*. Schicken Sie die Datei im nächsten Schritt unter Verwendung des Standardpassworts *android* durch *keytool* – der einzige für Sie interessante Wert ist der MD5-Fingerprint:

```
tamhan@TAMHAN14:~/.android$ keytool -list -v
-keystore debug.keystore

Enter keystore password:

...
Serial number: 117a777a
Valid from: Fri Nov 07 20:24:14 CET 2014 until:
Sun Oct 30 20:24:14 CET 2044
Certificate fingerprints:
  MD5: 41: <xxxxx> :9B
  SHA1: ...
```

Nach dem erfolgreichen Ausfüllen der Daten lässt sich der Schlüssel wie in Bild 3 gezeigt durch Anklicken des *Show*-Links auf den Bildschirm holen.

Erstellen Sie unter */src/main* ein Verzeichnis namens *assets*, das mit einer Textdatei namens *api_key.txt* befüllt wird. Kopieren Sie den Schlüssel in diese Datei, um die Konfiguration des Projektskeletts abzuschließen.

Aufgrund der Sensibilität der in Cloud Drive gelagerten Daten verlangt Amazon von Seiten der Entwickler eine zusätzliche Verifikation. Öffnen Sie den URL https://developer.amazon.com/cd/sp/overview.html und klicken Sie auf den Button *Whitelist Security profile*. Warten Sie nach dem Ausfüllen des Formulars, bis die Verifikations-E-Mail eingeht. Amazon zeigt sich hier normalerweise relativ flott. Nach dem Eingang der Nachricht sollten Sie aus Sicherheitsgründen allerdings einige Stunden warten, bis sich der Server aktualisiert hat. Berücksichtigen Sie beim Test zudem, dass die Bibliothek mit den von Google bereitgestellten Emulatoren nicht sonderlich gut funktioniert.

Fazit

Amazon bietet Entwicklern ein vollständiges Ökosystem von Produkten an: Wir konnten hier aus Platzgründen auf interessante Dienste wie In-App-Käufe, den Gamification-Client und den Push-Messaging-Service nicht eingehen (Bild 4). Wer seine Applikation ohne Google aufbauen möchte, muss nicht auf Komfort verzichten – Online-Services lassen sich auch beim Buchhändler zukaufen. ■

Links zum Thema

- Amazon Services und APIs
  https://developer.amazon.com/public/apis
- Amazon SDK
  https://developer.amazon.com/public/resources/development-tools/sdk

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Es lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter
www.tamoggemon.com

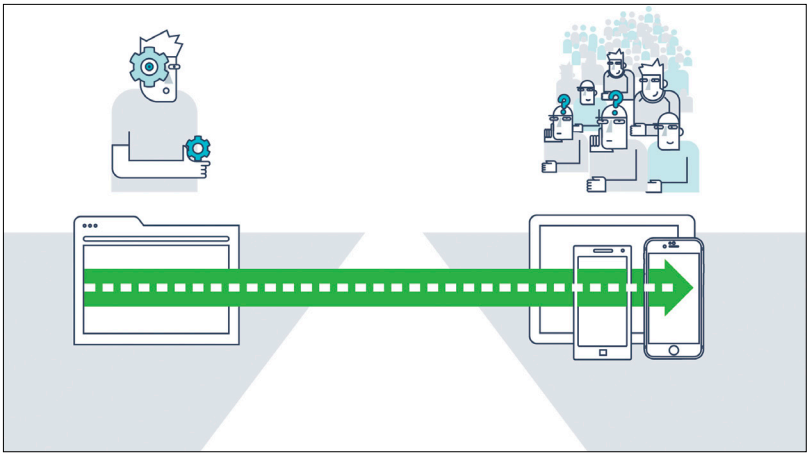
MOBILE APPS OHNE PROGRAMMIERUNG ERSTELLEN

Low Code – no Code

Wie lassen sich mobile Apps am besten implementieren?

Einerseits wollen Unternehmen vorhandene Applikationen für den mobilen Einsatz fit machen und andererseits neue Apps möglichst rasch fertigstellen. Die Lösung: Technisch versierte Benutzer aus den Fachbereichen können mobile Apps mit nur wenig oder ganz ohne Programmierung selbst erstellen.

Je größer ein Unternehmen, desto umfangreicher der Einsatz mobiler Endgeräte und Apps. So ist es heute durchaus üblich, dass in großen Firmen parallel an einer Reihe mobiler Applikationsprojekte gearbeitet wird – und zwar in nahezu allen Fachabteilungen, nicht nur in der IT. Gleichzeitig sind die Programmierer mit der Pflege und Weiterentwicklung einer Vielzahl von Applikationen befasst und ausgelastet. Die Teams kommen kaum mit der Unterstützung der Softwarebestände nach, und neue Projekte müssen warten (Bild 1).



Quelle: Progress

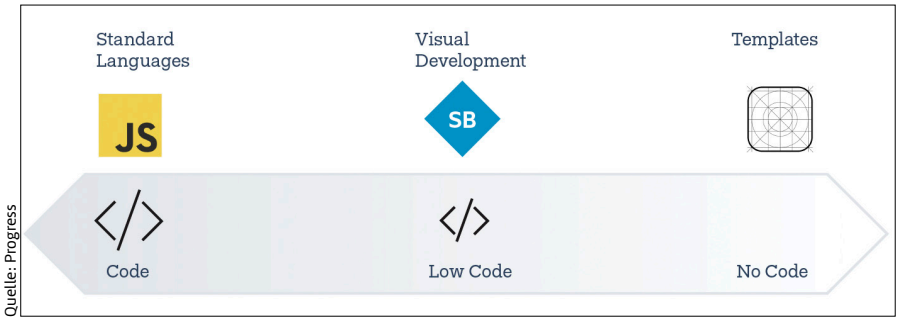
Die Software-Entwicklung sollte heute keine Einbahnstraße mehr sein, bei der die Fachbereiche zu lange auf die Fertigstellung neue Applikationen warten müssen (Bild 1)

In einer solchen Lage ist es verständlich, dass Mitarbeiter aus den Fachbereichen zur Selbsthilfe greifen und ihre Apps eigenständig erstellen. Das geschieht im Marketing, Vertrieb, Einkauf und in der Personalabteilung. In fast allen Fällen geht es dabei um eine schnellere und effizientere Kommunikation mit Geschäftspartnern, Kunden und Lieferanten. Technisch versierte Benutzer in den Fachabteilungen verwenden dazu sogenannte Low-Code-Plattformen, die eine zügige Entwicklung und Implementierung von Applikationen innerhalb von Tagen und mit einer minimalen Menge handgeschriebener Programmierzeilen ermöglichen.

Natürlich ruft die wachsende Verbreitung mobiler Apps, die mit wenig oder ganz ohne Code-Programmierung erstellt wurden, in den IT-Abteilungen Besorgnis hervor. Denn es stellt sich schon die Frage, welche Rolle die nicht hauptberuflichen Entwickler dann im Rahmen der Software-Erstellung in einem Unternehmen und bei der Umsetzung der digitalen Transformation spielen. Das kann dazu führen, dass die IT erst dann zu Rate gezogen wird, wenn die Dinge aus dem Ruder zu laufen drohen und es gilt, Fehlentwicklungen wieder in geregelte Bahnen zurückzuholen (Bild 2).

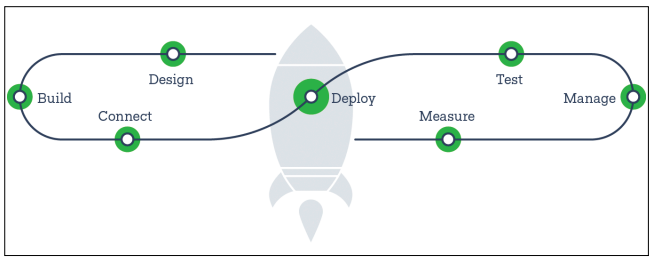
Vom Türsteher zum Katalysator

Gleichzeitig ändern sich die Erwartungen an die IT und damit an die Entwickler. Sie sollen nicht mehr als Kontrolleure wirken, sondern als Katalysator, der Neues fördert. Es gibt eine wachsende Bewegung in Unternehmen, die davon ausgeht, dass die Applikationsentwicklung jedermann angeht. Ursächlich dafür ist die steigende Zahl von Millennials unter den Mitarbeiter. Sie sind technikaffin und erwarten eine Mitsprache bei den Technologien und Endgeräten, die sie beruflich einsetzen. All diese Trends stellen Unternehmen vor beachtliche Herausforderungen. Statt Einschränkungen sollten sie darin Chancen erkennen und die Möglichkeiten der App-Entwicklung durch technisch versierte Anwender in den Fachbereichen nutzen, um die hauptberuflichen Programmierer zu entlasten. Diese können sich dann voll und ganz auf Innovation konzentrieren, die dem Unternehmen Wettbewerbsvorteile verschaffen (Bild 3).



Quelle: Progress

Moderne, offene Applikationsplattformen beschleunigen den Entwicklungszyklus neuer Anwendungen (Bild 2)



Quelle: Progress

Technische versierte Benutzer aus den Fachabteilungen sollten in der Lage sein, mobile Apps mit nur wenig oder ganz ohne Programmierung zu erstellen zu können (Bild 3)

Dafür wird eine Applikationsplattform benötigt, die offenen Standards folgt, sich an Open-Source-Prinzipien ausrichtet und reibungslos mit anderen in einem Unternehmen vorhandenen Systemen zusammenarbeitet. Ein Beispiel dafür ist die Telerik Platform von Progress, mit der Entwickler Web-, native und hybride Apps für alle Plattformen erschaffen und technische versierte Benutzer aus den Fachabteilungen mobile Apps mit nur wenig oder ganz ohne Programmierung erstellen können. Sie können beispielsweise Bildschirme mit Hilfe einer Auswahl vordefinierter Templates konfigurieren und sie einfach und sicher mit Unternehmensdaten verbinden, ohne dafür eine Zeile Programmcode schreiben zu müssen. Das reicht für viele Apps bereits aus, die für eine schnelle Lösung aktueller Anforderungen in den Fachbereichen benötigt werden.

Produktivität und Innovation fördern

Positive Auswirkung hat ein solches Vorgehen auch für die Mitarbeitermotivation, denn eine aktive Beteiligung fördert die Kreativität, Produktivität und Innovation. Die Benutzer der Apps kennen die fachlichen Anforderungen am besten und sind in der Lage, diese zügig mit einer Lösung umzusetzen, die sie dabei unterstützt. Wichtig dabei ist, dass die hauptberuflichen Entwickler mit entsprechenden Rahmenbedingungen für die notwendige Sicherheit, Verlässlichkeit, Skalierbarkeit und Flexibilität sorgen. Von diesen Vorgaben profitieren alle Beteiligten. Die hauptberuflichen Entwickler erhalten den Freiraum, sich um die strategischen Projekte zu kümmern, und die fachlich versierten Benutzer aus den Fachabteilungen müssen nicht mehr abwarten, bis die IT ihre Anforderungen abgearbeitet hat. Stehen dringend benötigte neue Apps schneller zur Verfügung, so profitiert davon letztlich auch das gesamte Unternehmen. ■

Olf Jännsch
ist Regional Vice President, Central and Eastern Europe bei Progress.
www.progress.com

Foto: iStockphoto / erhui979

KOSTEN-NUTZEN-ANALYSE

Cloud-Strategie

Empfehlungen für eine erfolgreiche Cloud-Strategie von Unternehmen.

Selber machen oder auslagern? Für immer mehr Unternehmen ist mittlerweile klar: Eigene Server lohnen sich eigentlich kaum mehr – und es werden im großen Stil Server-Ressourcen und Anwendungen in der Cloud gemietet. Viele Analysten zeigen ja auch auf, dass die Betriebskosten in der Cloud weit niedriger ausfallen als bei der Nutzung von hauseigenen Ressourcen. Kein Wunder also, dass der Cloud-Markt auch in Deutschland trotz Bedenken, etwa in puncto Datensicherheit, weiter ungebremst wächst.

Zu den Vorteilen einer Cloud wie Flexibilität und einfache Skalierbarkeit kommt für viele Unternehmen die Aussicht auf Kostensenkungen als wichtiger Grund für den Wechsel in die Wolke hinzu. Doch ist das wirklich so? Spart die Auslagerung der IT in die Cloud automatisch Geld?

Und falls ja, wie sieht eine geeignete Cloud-Strategie aus? Wie entwickelt ein Unternehmen einen tauglichen Einsatzplan für den Weg in die Cloud?

Folgendes Szenario kommt so oder ähnlich in vielen Unternehmen vor: Ein Abteilungsleiter benötigt einen Server. Er fragt in der IT-Abteilung an, die einen Server aufsetzen kann. Wartezeit: eine Woche. Kosten: mehrere Hundert Euro pro Jahr. Das sorgt bei dem Abteilungsleiter für Verwunderung. Privat bekommt er bei Hosting-Anbietern für wenige Euro pro Monat einen Cloud-Server. Wartezeit: keine.

Seinen Ursprung hat der Cloud-Erfolg im Consumer-Bereich – durch hohe Standardisierung kostengünstige Out-of-the-Box-Lösungen wie Dropbox und Co. lösten einen Boom aus. Die Erwartungshaltung ist entsprechend: Ein Wechsel in die Cloud muss schnell gehen und kostengünstig sein.

Für den Unternehmenseinsatz eignen sich solche Out-of-the-Box-Fertiglösungen jedoch nur selten. Meist müssen auf die jeweiligen IT-Voraussetzungen zugeschnittene, das heißt stark angepasste Cloud-Lösungen her. Eine Kosten-Nutzen-Analyse der Cloud ist für Unternehmen gar nicht so einfach. Die Frage, ob die Total Cost of Ownership (TCO), also die Gesamtbetriebskosten der Wolke, geringer sind als bei der Nutzung eigener Server, lässt sich nicht pauschal beantworten und muss in jeder Firma einzeln betrachtet werden.

Das IT-Beratungsunternehmen Saugatuck Technologies hat sich das Kosten-Nutzen-Verhältnis der Cloud näher angesehen. Das Ergebnis: Es hängt maßgeblich von drei Faktoren ab, ob man mit der Cloud tatsächlich Geld spart: der Art des IT-Workloads, der Größe des IT-Workloads und – der wohl wichtigste Faktor – der Effizienz der hauseigenen IT. Je effizienter die hauseigene IT arbeitet, desto geringer fällt der Kostenvorteil der Cloud aus. Unternehmen sollten daher für jeden einzelnen Bereich der IT genau kalkulieren, ob sich ein Wechsel in die Cloud lohnt.

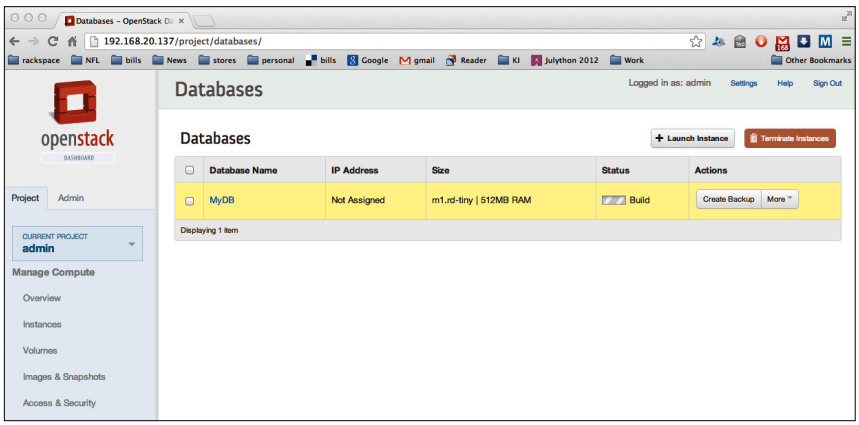


Foto: openstack.org

Gegen den Vendor-Lock-in: Unternehmen setzen vermehrt auf die Open-Source-Lösung OpenStack. Damit lassen sich Private und Public Clouds aufbauen (Bild 1)

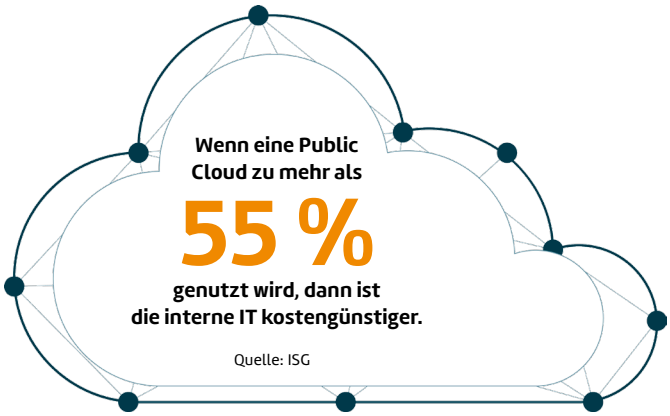
Nur wenige Unternehmen wie Start-ups sind in der komfortablen Situation, dass sie mit der Cloud bei null anfangen können und keinen Berg an Unternehmens- und Kundendaten mit in die Wolke umziehen müssen. Viele Firmen haben im Lauf der Jahre eine Vielzahl an Servern für die unterschiedlichsten Zwecke in Betrieb genommen, auf die sich die Daten jetzt verteilen.

Tim Minahan, Chief Marketing Officer bei SAP Cloud, warnt Firmen davor, sich beim Cloud-Umstieg in der TCO-Debatte zu verstricken. Auch wenn die Verlagerung der Geschäftstätigkeit in die Cloud meist mit erheblichen Kosteneinsparungen einhergehe, liege – so Tim Minahan – der eigentliche Nutzen der Cloud in zwei anderen Faktoren, nämlich der Innovation und der Flexibilität.

Die einfache Konfiguration und die Erweiterbarkeit der Cloud bieten Unternehmen die Flexibilität, die sie benötigen, um ihre Geschäftsprozesse den dynamischen Marktbedingungen schnell und einfach anzupassen.

Darüber-hinaus-Lösung

Nicht wenige Unternehmen, die sich für den Umstieg in die Cloud interessieren, schrecken schließlich doch davor zurück, weil Systemhäuser und Cloud-Anbieter oft den Komplettumstieg in die Wolke empfehlen. Nur so ließe sich, so die Begründung, von den Vorteilen der neuen Technologie voll und ganz profitieren. Doch dieser Ansatz ist nicht weit genug gedacht. Laut SAP-CMO Minahan ist die Cloud keine Entweder-oder-Lösung und auch keine Und-Lösung. Die Cloud sei vielmehr eine »Darüber hinaus«-Lösung: Zufriedenstellend laufende On-Premise-Lösungen ließen sich durch passende Cloud-Angebote erweitern.

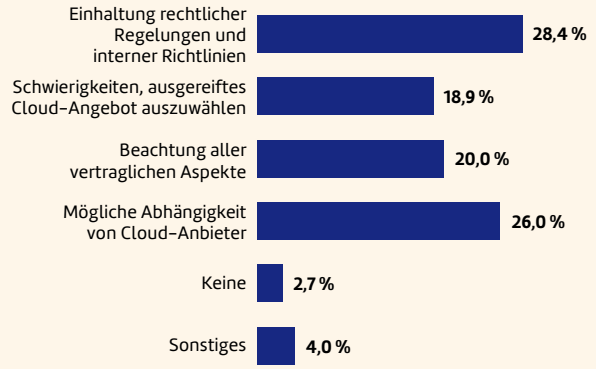


Ein Beispiel für eine solche »Darüber hinaus«-Lösung ist das Unternehmen T-Mobile. Der Mobilfunkanbieter hat seine vorhandene CRM-Lösung um eine Cloud-Anwendung für den Kundenservice ergänzt. Damit reagiert T-Mobile auf Kundenanfragen – außer über die herkömmlichen Kanäle wie Telefon und E-Mail – nun auch über soziale Medien wie Facebook oder Twitter. Damit soll die Lösungsrate beim ersten Auftreten eines Problems deutlich verbessert worden sein und die Kundenzufriedenheit und die Zahl der Vertragsverlängerungen sollen spürbar gestiegen sein.

Cloud first

Trotz aller Kostenerwägungen und auch Nachteilen, die Cloud-Lösungen fraglos haben, geht ohne die Wolke wohl bald nichts mehr: »Cloud first« wird, so die Analysten von ▶

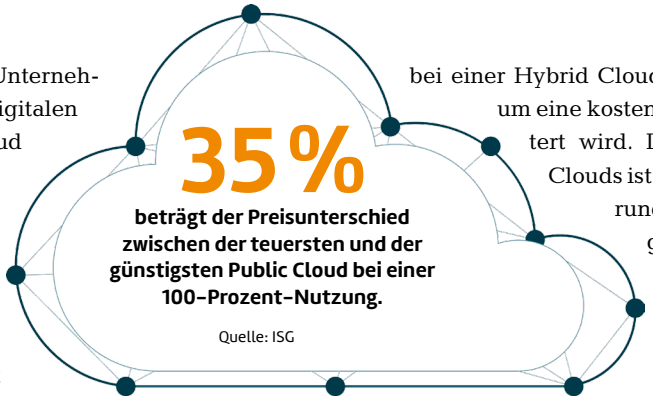
Hürden beim Wechsel in die Cloud



Angst vor der Abhängigkeit ist für Firmen die größte Hürde beim Wechsel in die Cloud.

web & mobile developer 4/2016 Quelle: DsiN Cloud-Scout Report 2015, Unternehmen ab 250 Mitarbeiter

IDC, zum neuen Mantra der Unternehmens-IT. Das Fortschreiten der digitalen Transformation ist ohne die Cloud kaum denkbar. Laut der IDC-Studie »Hybrid Cloud in Deutschland 2015/16« nutzen oder implementieren bereits rund sechs von zehn Unternehmen Cloud-Dienste. Dabei bleiben Private Clouds mit 57 Prozent das bevorzugte Modell. Auf Public Clouds setzen 27 Prozent der Unternehmen.



Bei der Public Cloud teilen sich alle Kunden die Server-Ressourcen des Cloud-Anbieters. Im Gegensatz zu einer Private Cloud – hier nutzt ein Unternehmen seine eigene Cloud-Infrastruktur. Diese kann im eigenen Unternehmen stehen oder bei einem Hosting-Anbieter.

Externe Cloud-Dienste

Die Analysten von IDC gehen davon aus, dass 2017 bereits 60 Prozent der Unternehmen Public Clouds nutzen werden. Die Entscheidung »Selber machen oder auslagern?« fällt immer häufiger zugunsten externer Cloud-Dienste aus.

Hybrid-Trend: Die Frage, ob Public Cloud oder Private Cloud, stellt sich für viele Unternehmen oft nicht mehr: Der Trend geht zu Mischformen. Hybrid Clouds kombinieren verschiedene Cloud-Modelle. Typischerweise handelt es sich bei einer Hybrid Cloud um eine Private Cloud, die um eine kostengünstige Public Cloud erweitert wird. Die Verbreitung von Hybrid Clouds ist im Vergleich zum Vorjahr um rund ein Drittel auf 20 Prozent gestiegen.

Das Beispiel zeigt, dass sich Unternehmen trotz Cloud-Hype nicht gezwungen fühlen sollen, etablierte Systeme zu ersetzen. So ist es durchaus sinnvoll, erfolgreich laufende On-Premise-Lösungen weiterhin einzusetzen und die Cloud als nützliche Erweiterung für einzelne Problemlösungen zu verwenden.

Fahrplan

Die Auswahl geeigneter Cloud-Varianten und der richtigen Anbieter kosten Unternehmen viel Zeit. Oft fehlt bei den IT-Mitarbeitern bislang die Erfahrung mit Cloud-Diensten und vor allem mit dem Wechsel-Prozedere. Umso wichtiger ist es, die verschiedensten Stolpersteine rechtzeitig zu erkennen und über sie hinwegzusteigen.

Es empfiehlt sich, eine Art Fahrplan in die Cloud zu entwickeln: Viele Unternehmen fangen mit ein oder zwei neuen Anwendungen in der Cloud an. Bei erfolgreichem Einsatz kommen weitere neue Apps hinzu, und irgendwann wagt man sich dann an die Migration bestehender Anwendungen.

Kostenvergleich: Public Cloud vs. interne IT

Eine Studie des Marktforschungs- und Beratungshauses Information Services Group (ISG) kommt zu dem Ergebnis, dass sich Infrastructure as a Service (IaaS) in der Public Cloud nur dann rechnet, wenn man die Ressourcen in der Cloud nur zu gut der Hälfte nutzt. Ansonsten ist die Nutzung von Rechenleistung in den eigenen vier Wänden preiswerter.

Der Grund liegt in der nutzungsabhängigen Preisberechnung. So rechnen die gängigen IaaS-Anbieter wie Amazon, Google und Microsoft die Server-Nutzung nach Minuten oder Stunden ab. Wenn eine Firma die externen Cloud-Ressorcen umfangreich nutzt, dann übersteigen die Nutzungsgebühren schnell die Kosten, die für eine interne IT anfallen würden.

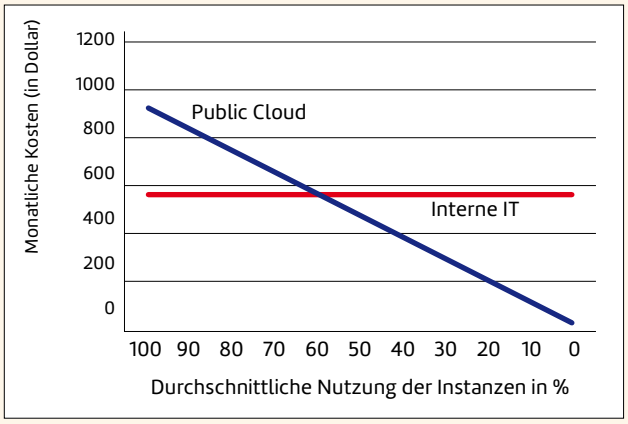
Die Analysten von ISG haben die Kosten für die Nutzung der Public-Cloud-Dienste Amazon Web Services, Google Cloud Platform, Microsoft Azure und IBM SoftLayer anhand einer Musterkonfiguration den Kosten einer internen IT gegenübergestellt. Als Beispiel dienten vier Anwendungs-Server, zwei Datenbank-Server sowie ein Storage-Server.

Das Ergebnis: Zu 100 Prozent auf die Cloud zu setzen, wäre in diesem Beispielszenario keineswegs die günstigste Option. Die Cloud käme das Unternehmen um 32 Prozent teurer als eigene Server. Bis zu einer Nutzungsquote von 54 Prozent wäre die Cloud günstiger als die interne IT, ab 55 Prozent wäre sie teurer.

Im Szenario von ISG gilt demnach die Faustregel: Je weniger ein Public-Cloud-Dienst genutzt wird, desto lukrativer ist er preislich.

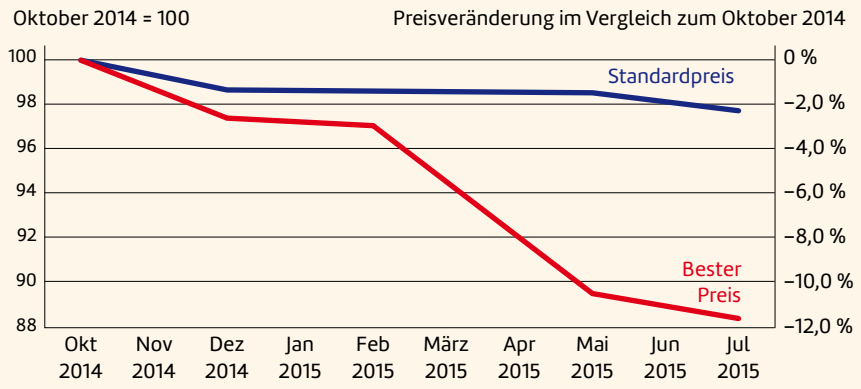
Die Studie berücksichtigt nicht die Kosten für die Datenübertragungen von und zur Public Cloud sowie für die interne IT. Da die Cloud-Dienste die übertragenen Daten meist zusätzlich in Rechnung stellen, dürfte die Beispielrechnung sogar noch stärker zugunsten der internen IT ausfallen.

Übrigens: Was die einzelnen Cloud-Anbieter kosten, ist sehr unterschiedlich. Bei einer 100-Prozent-Nutzung der Cloud-Dienste kostet laut ISG der teuerste Anbieter 35 Prozent mehr als die kostengünstigste Option.



Break-even bei 55 Prozent: Bei reger Nutzung soll eine interne IT günstiger sein als die Cloud, hat ISG Research berechnet

Cloud Price Index: So ändern sich die Preise



Die günstigsten Preise für Public-Cloud-Dienste in den USA sind zwischen Oktober 2014 und Juli 2015 um 12 Prozent gefallen.

web & mobile developer 4/2016 Quelle: 451 Research

Vor dem Wechsel steht ein Cloud-Readiness-Check an – eine Bestandsaufnahme: Welche Anwendungen sind im Einsatz und welche taugen auch für den Betrieb in der Cloud? Ein solcher Check beleuchtet nicht nur, ob Anwendungen implementierungstechnisch cloudtauglich sind, sondern auch, ob die Unternehmensorganisation einen Cloud-Wechsel zulässt.

In nächsten Schritt erstellt man einen Katalog mit den technischen Anforderungen an die Cloud: Welche Software und welche Techniken muss der Cloud-Dienst unterstützen? Dabei sollte man auch künftige Erweiterungen beachten. Unterstützt der Anbieter zum Beispiel APIs, um später etwa Amazon Web Services (AWS) zur Architektur hinzuzufügen?

Abteilungen einbeziehen

Wer die Cloud in erster Linie als zusätzliche Möglichkeit betrachtet, um Investitions- und Betriebskosten einzusparen, hat eine zu eingeschränkte Sichtweise auf das Thema.

Um Cloud-Dienste nachhaltig in die Unternehmensarchitektur zu integrieren und von allen Vorteilen zu profitieren, ist es wichtig, dass man bereits in der frühen Konzeptionsphase alle Fachabteilungen mit ins Boot holt und den Fahrplan in die Wolke gemeinsam erstellt.

Von Anfang an sollte definiert werden, welche Anwendungen irgendwann den Weg in die Cloud antreten sollen. So hat man nicht nur den aktuellen Bedarf im Blick, sondern kann auch zukünftige Entwicklungen miteinbeziehen. Folgende Fragen sollten zwischen der IT-Abteilung und den Fachabteilungen geklärt werden:

- Welches Ziel hat der Wechsel einer Anwendung in die Cloud?
- Wie wichtig ist die Verfügbarkeit einer Anwendung?
- Welche Folgen hat die Nichtverfügbarkeit einer Anwendung und welche Kosten entstehen bei einem Ausfall?
- Wie ist die erwartete Nutzungsdauer der Anwendung, die in die Cloud soll? Wie oft muss die Anwendung aktualisiert werden?
- Innerhalb welches Zeitraums soll eine Anwendung in die Cloud wechseln?

Vor allem sollte man klären, welche rechtlichen Anforderungen die Anwendung in der Cloud zum Beispiel in Bezug auf die Datensicherheit erfüllen muss – Stichwort Safe Harbor. Es ist unerlässlich, den Rat eines darauf spezialisierten Juristen einzuholen.

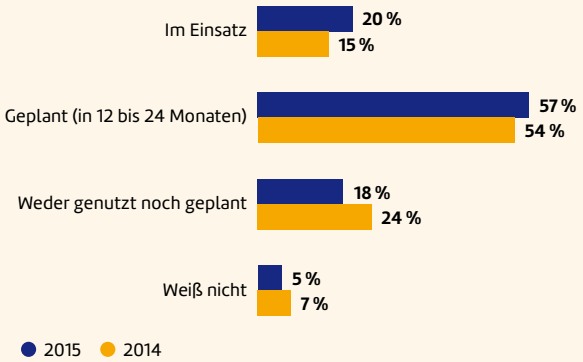
Falls die Übernahme einer anderen Firma oder eine Fusion geplant ist: Wie wirken diese sich auf die Nutzung der Anwendungen in der Cloud aus?

Ein Nachteil der Cloud: Die Anwendungen sowie die Daten befinden sich in fremden Händen. Das Unternehmen ist also der Infrastruktur und der Ausfallsicherheit des Cloud-Anbieters ausgeliefert. Wenn der Cloud-Dienstleister insolvent ist oder aus anderen Gründen den Geschäftsbetrieb einstellt, dann steht es häufig schlecht um die Unternehmensdaten.

Daher sollte man in jedem Fall bereits im Vorfeld eine mögliche Exit-Strategie mit seinem Cloud-Anbieter abklären: Wie sieht es mit der Interoperabilität aus? Besteht jederzeit die Möglichkeit, auf die Rohdaten in der Cloud zuzugreifen? Der Wechsel eines Cloud-Anbieters ist normalerweise nicht so ohne Weiteres möglich. Vielfach sind die Dienste der Anbieter untereinander inkompatibel. Es besteht also die Gefahr des Vendor-Lock-ins, der Abhängigkeit von einem Hersteller.

Die Abhängigkeit von einem Anbieter ist auch die größte Befürchtung vieler Unternehmen beim Wechsel in die Wolke: Laut dem Cloud-Scout Report 2015 der Intitiative Deutschland sicher im Netz e.V. wird unter den Hürden für den Umstieg auf die Cloud die mögliche Abhängigkeit von einem Cloud-Dienstleister bei Unternehmen jeder Größe mit Abstand am häufigsten genannt.

Hybrid Clouds sind im Kommen



Trend zur Hybrid Cloud: 2015 nutzten 20 Prozent der Unternehmen eine Hybrid Cloud, 2014 waren es erst 15 Prozent.

web & mobile developer 4/2016 Quelle: IDC, Oktober 2015

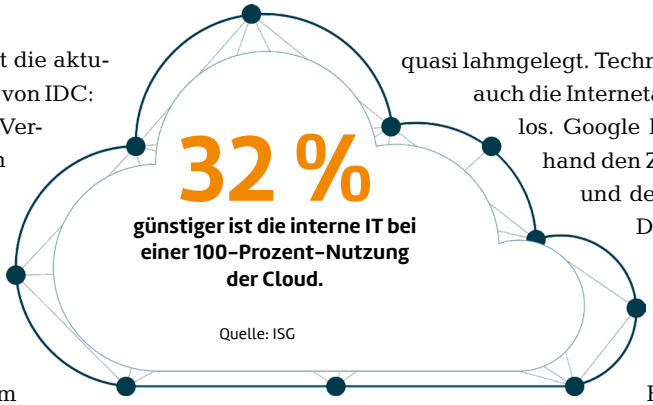
Public, Private oder Hybrid: Welches Cloud-Modell eignet sich für was?

Welche Cloud für was? Zum Beispiel eignen sich tagtäglich genutzte Anwendungen ohne sensible Daten für die kostengünstige Public Cloud.

Standard-Anwendungen Produktivanwendungen, die täglich in Gebrauch sind	• **Standardisierte Funktionalität** • **Weniger sensible Daten** • **Geringe Auslastung: unvorhergesehener Bedarf**	**Public Cloud**
Strategische Anwendungen Hauptanwendungen, die Nutzer für das Unternehmen bewirken	• **Zentrale Bedeutung für den Geschäftsbetrieb** • **Kontrolle über Datensicherheit** • **Hoher Nutzungsgrad und unvorhersehbare Auslastung**	**Private oder Hybrid Cloud**
Pilotanwendungen Anwendungen, die Innovationen vorantreiben	• **Ermöglicht kurze Entwicklungszyklen** • **Bietet einen einfachen Weg, um zu Hauptanwendungen überzugehen** • **Bewältigt große Schwankungen bei Bedarf und Größenordnung**	**Public, Private oder Hybrid Cloud**

web & mobile developer 4/2016 Quelle: Intel

Zu demselben Ergebnis kommt die aktuelle Hybrid-Studie der Analysten von IDC: Für 36 Prozent der CIOs ist die Vermeidung einer Abhängigkeit von einem einzigen Hersteller sehr wichtig. »CIOs wollen sich nicht in der Sackgasse des Vendor-Lock-ins wiederfinden«, so Matthias Kraus, Analyst bei IDC.



Das ist auch der Grund, warum sich immer mehr Unternehmen im Rahmen ihrer Cloud-Strategie für Open-Source-Lösungen entscheiden. So ist bereits bei fast einem Drittel der Unternehmen OpenStack im Einsatz (Bild 1). OpenStack ist eine Open-Source-Software für den Betrieb von Private und Public Clouds. Eine solche Open-Source-Lösung vereinfacht bei einem Wechsel des Cloud-Anbieters die Mitnahme der Unternehmensdaten.

Die Verbindung zur Cloud erfolgt in den meisten Unternehmen über das Internet. Fällt die Internetverbindung aus, aus welchem Grund auch immer, dann ist kein Arbeiten in der Cloud mehr möglich. Daher sollte man sich unbedingt Gedanken darüber machen, wie das Geschäft in einem solchen Fall weitergeht.

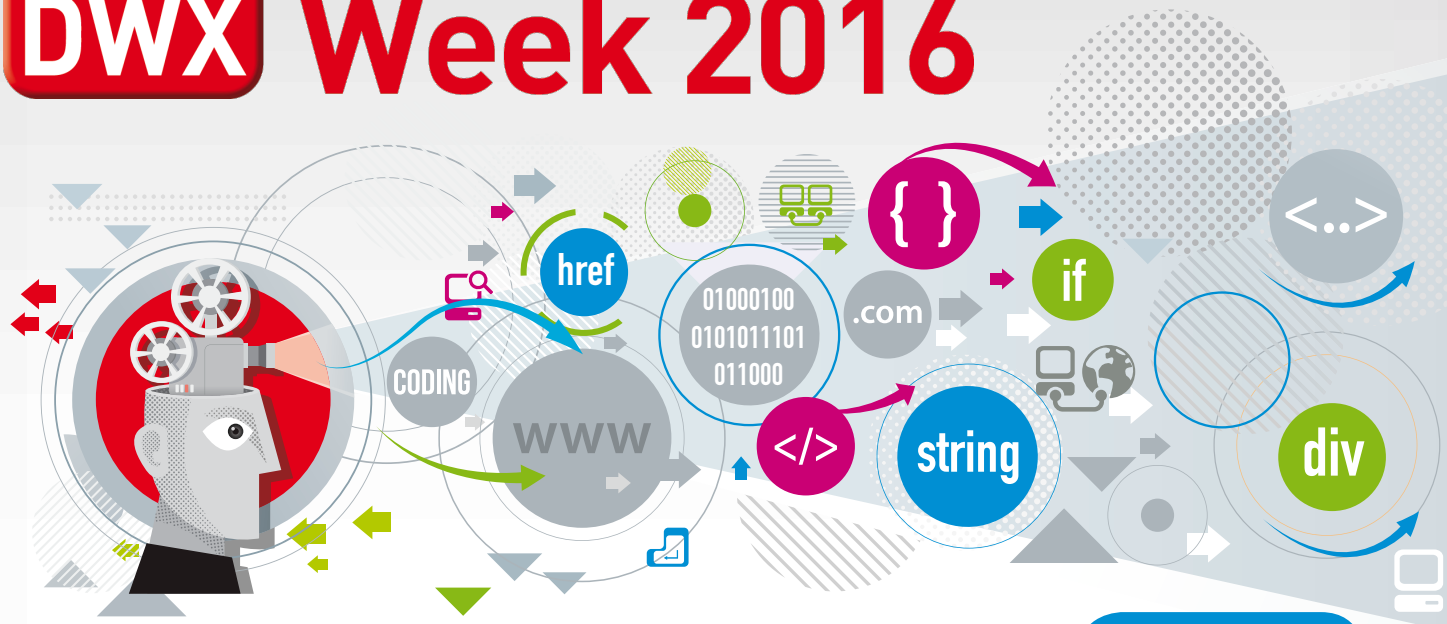
Oft kann übrigens schon ein kleiner Fehler zu großen Ausfällen und Problemen führen. Ein Beispiel ist die Webseite Radio.de. Vor einigen Jahren war das Unternehmen 48 Stunden lang ohne Daten und ohne Bürosoftware – und damit quasi lahmgelegt. Technisch war alles in Ordnung und auch die Internetanbindung arbeitete reibungslos. Google hatte Radio.de jedoch kurzerhand den Zugang zu seiner Bürosoftware und den abgelegten Daten gesperrt. Der Grund: Ein Fehler im Bezahlsystem von Google hatte dafür gesorgt, dass ein für ein Unternehmen eigentlich läppischer Betrag von ein paar Hundert Euro nicht abgebucht worden war.

Welchen Kostenvorteil bietet die Cloud konkret für Ihr Unternehmen? Der TCO-Kalkulator der Analysten der Experton Group unter www.business-cloud.de/calculator gibt Ihnen einen groben Anhaltspunkt, wie viel Sie mit Ihrer vorhandenen IT-Landschaft bei einem Wechsel in die Cloud sparen können. ■

Konstantin Pfliegl
ist Redakteur bei der Zeitschrift com! professional. Er hat mehr als fünfzehn Jahre Berufserfahrung als Redakteur für verschiedene Print- und Online-Medien.
www.xing.com/profile/Konstantin_Pfliegl

DWX Developer Week 2016

20.–23. Juni 2016,
Messe Nürnberg



Mit Code
DWX16wmd
€ 200,–
sparen!

Das Event 2016 mit neuen Specials:

Thema Angular 2:

- **Speed-up AngularJS – Hochperformante Webanwendungen bauen**
  Timo Korinth
- **Live-Coding mit Angular 2**
  Johannes Hoppe
- **Angular 2 Change Detection Explained**
  Pascal Precht
- **Schnellstart mit Angular 2**
  Johannes Hoppe, Gregor Woiwode
- **Angular 2 Upgrade**
  Manfred Steyer

Thema Industrie 4.0:

- **Smart Development in the Industry 4.0 – Best Practices and Prospects**
  Alexander Schulze
- **IoT Ultimate Edition**
  Damir Dobric
- **Innovationsmotoren für IoT**
  Patrick Lobacher
- **Wie hardwarelastige Mittelständler ihre Prozesse und Methoden anpassen um Industrie 4.0-Lösungen für Ihre Kunden und sich zu liefern**
  Lars Roith
- **(UML) Modellbasiert Code generieren und debuggen – ja es funktioniert wirklich!**
  Daniel Sigl

developer-week.de **DeveloperWeek**

Aussteller & Sponsoren: adorsys, AIT, brainLight. LIFE IN BALANCE, Fast Reports Reporting must be Fast!, GFU Cyrus AG, MATHEMA, MAXIMAGO, SDX, SPARX SYSTEMS, TEXT CONTROL

Veranstalter: developer media, Neue Mediengesellschaft Ulm mbH

Präsentiert von: dnc dotnetpro, web & mobile DEVELOPER

WEB- UND SOFTWARE-ENTWICKLUNG

HHVM vs. PHP 7

Facebook animiert durch seine Aktivitäten die PHP-Entwickler zu Höchstleistungen.

Mit Hack, einer quelloffenen Sprache, möchte Facebook viele der Unzulänglichkeiten von PHP beheben und einen Beitrag zur Entstehung robuster(er) Webapplikationen beisteuern. Seit PHP in der finalen neuen Major-Version 7.0 verfügbar ist, hat sich der Optimismus in der PHP-Gemeinde verstärkt, und man spürt sogar zu Recht eine wohlverdiente Aufbruchsstimmung.

PHP ist die populärste Web-Entwicklungssprache, die manchen Schätzungen zufolge für 80 Prozent aller Websites verantwortlich zeichnet (Bild 1).

Traditionsreich: Das offizielle PHP-Logo blieb unverändert, während sich unter der Haube eine ganze Menge geändert hat (Bild 1)

Zu den aktiven Nutzern von PHP-Anwendungen zählte seit seiner Entstehung unter anderem das soziale Netzwerk Facebook (Bild 2). Um die eigenen Webhosting-Kosten zu senken, hat Facebook eine mit PHP interoperable Entwicklungssprache namens Hack Lang erfunden und eine quelloffene Laufzeitumgebung namens HHVM, kurz für HipHop Virtual Machine, entworfen, die sowohl Hack Lang als auch PHP unterstützt (Bild 3).

Die letzte Hauptversion, PHP 5, wurde im Jahre 2004 vorgestellt. PHP 7 erblickte erst am 3. Dezember 2015 das Licht der Welt, also mehr als ein Jahrzehnt später. Insofern ist es nicht weiter erstaunlich, dass man im Facebook-Hauptquartier das Duo aus Hack/HHVM zum PHP-Nachfolger küren wollte.

Stärkere Performance

Das Duo aus Hack/HHVM konnte gegenüber PHP 5.x mit deutlich stärkerer Performance und niedrigerer Ressourcenanforderungen durch eine effizientere Speichernutzung auftrumpfen. Doch die PHP-Gemeinde wollte keinesfalls das Handtuch werfen und hat sich stattdessen tapfer der Herausforderung gestellt.

Da Facebook mit Hack/HHVM eine konkurrenzfähige Alternative zu PHP 5.6 bieten konnte, schien eine gründliche Umarbeitung von PHP unvermeidlich.

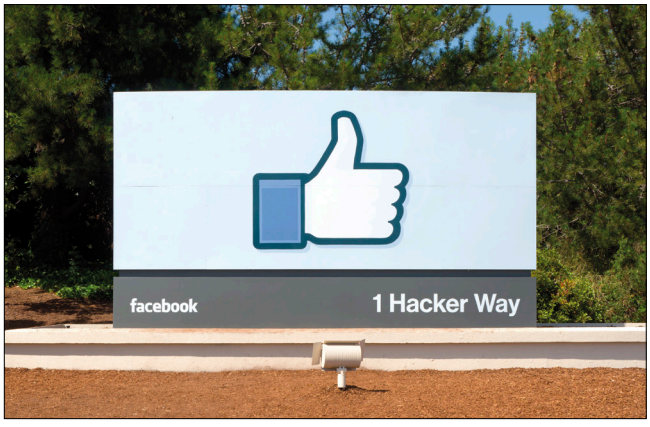
Nach zwei Jahren intensiver Entwicklung und acht Release-Candidate-Versionen haben Rasmus Lerdorf, der Erfinder von PHP, und sein PHP-Entwicklerteam die Ziellinie nun endlich erreicht: PHP 7 liegt in einer finalen Version vor. Die Zend-Engine, der Compiler und die Laufzeitumgebung von PHP wurden in PHP 7 komplett überarbeitet.

»Bei der vorigen, stabilen PHP-Version handelte es sich um den 5.6-Zweig«, sagt Rasmus Lerdorf. Da die PHP-Entwickler zuvor an einem Unicode-basierten PHP-6-Nachfolger gearbeitet hatten und damit kaum vom Fleck kamen, beschloss das PHP-Core-Team, sich davon zu distanzieren und mit PHP 7 einen frischen, neuen Anfang zu wagen, um vor allem Verwechslungen und Missverständnissen vorzubeugen.

Laut Heroku, einem Anbieter diverser Cloud-basierter Produkte, haben sich interessanterweise in kürzester Zeit über 50 Prozent aller PHP-Entwickler auf der Heroku Platform aus Performancegründen bereits für PHP 7 entschieden.

Doch auch Facebook blieb nicht stehen. Neuerdings hat Facebooks HipHop Virtual Machine (HHVM) in der stabilen Version 3.11.0 die Kompatibilität zu PHP 7 (vorerst noch unvollständig) eingeführt, und so geht das Rennen zwischen der Hack Lang (und HHVM) und PHP 7 wieder einmal in die nächste Runde.

Mit der Vorstellung von PHP 7 konnte die Gemeinde das Schlimmste abwenden und der beliebten Websprache auf absehbare Zeit eine solide Zukunft sichern. Dennoch ist die Existenz von Hack Lang und HHVM aus der IT-Landschaft der Web-Anwendungsprogrammierung nicht mehr wegzu-



Like it or not: Das schlichte Schild mit traditionsreichem Like-Button am Eingang des Facebook-Hauptquartiers in Menlo Park (Bild 2)

Der Herausforderer: Das offizielle Logo der HHVM sieht genauso kantig aus, wie die Absichten von Facebook aggressiv sind beziehungsweise waren (Bild 3)

denken. Es stellen sich zahllose Fragen nach den Vor- und Nachteilen der beiden Sprachen und deren Interoperabilität in der Praxis.

Dieser Bericht nimmt beide Sprachen, PHP in der Version 7 und Hack, unter die Lupe und beleuchtet, welche Vorteile sie jeweils haben und für welche Einsatzszenarien sie sich am besten eignen.

Die Evolution von Hack Lang und HHVM

Bei Hack Lang handelt es sich um eine mit PHP interoperable Sprache für HHVM. Hack Lang möchte das schnelle Entwicklungstempo einer dynamisch typisierten Sprache mit der Disziplin einer statischen Typisierung mit einer sofortigen Typprüfung verbinden. Zu den besonderen Schmankerln von Hack Lang zählen Features wie Type Annotations, parametrisierte Klassen und Methoden, Type Aliases, Lambdas und andere.

Im Herzen von HHVM werkelt ein JIT-Compiler (Just in time). Anstatt den Code direkt zu interpretieren oder ihn nach C++ zu cross-kompilieren, kompiliert ihn HHVM in einen Intermediate-Bytecode. Im nächsten Schritt wird der Intermediate-Bytecode zur Laufzeit in 64-Bit-Intel-Maschinencode übersetzt. Dieser zweite Kompilierungsvorgang erlaubt es, eine Reihe von Optimierungen durchzuführen, die sich in statisch kompilierten Binärdateien nicht umsetzen ließen.

Um die Motivation von Facebook für die Entwicklung von HHVM zu verstehen, ist es wichtig, die Entstehung von Hack Lang im zeitlichen Kontext der Entwicklung der PHP-Sprache zu betrachten.

Facebook hatte im Jahr 2009 ein derart starkes Wachstum erfahren, dass es dieses nur mit Mühe und Not handhaben konnte. Die Mitgliederzahlen wuchsen schneller, als Facebook die zusätzlichen Server-Kapazitäten bereitstellen konnte. Noch im Februar 2009 hatte Facebook 175 Millionen Mitglieder; bis zum September 2009 hat das soziale Netzwerk die 300-Millionen-Mitglieder-Marke überschritten. Mit bloßem Geld ließ sich ein derlei explosives Wachstum sicherlich nicht handhaben. Auf die Dauer war es aber vor allem technisch nicht tragbar. Die Facebook-IT konnte gar nicht schnell genug neue Hardware in Betrieb nehmen wie der Bedarf nach mehr Rechenleistung stieg explosionsartig an.

Anfang 2010, als PHP noch in der vergleichsweise langsamen Version 5.3 ein stilles Dasein fristete, während PHP 5.4 noch in weiter Ferne lag und das bevorstehende Fiasko mit der Unicode-basierten sechsten Generation von PHP noch gar nicht absehbar war, entstand bei Facebook unter der Leitung von Drew Paroski die HHVM (Bild 4).

Facebook setzte damals den PHP-zu-C++-Compiler HPHPc und den Interpreter HPHPi (zur Ausführung des Codes im Entwickler-Modus ohne das Kompilieren) ein. Es handelte sich hierbei um zwei voneinander unabhängige Werkzeuge, die sich leider durch eher unbeabsichtigte, also subtile Unterschiede auszeichneten. Die Arbeit am Interpreter hatte sich nicht wirklich einfach gestaltet, denn die Resultate in HPHPc waren nicht hinreichend vorhersehbar. Laut dem Facebook-Team rund um Josh Watzman gab es mit diesen beiden launischen Tools immer wieder unliebsame Überraschungen. Eine Zusammenführung des Interpreters und des Compilers war zwar unabdingbar, aber die Aufgabe fiel in die Kategorie »leichter gesagt als getan«.

Die bloße Zusammenführung des Interpreters und des Compilers in HHVM hätte das Problem mit der Unvorhersehbarkeit der Resultate jedoch nicht automatisch gelöst. Ein weiteres, nicht minder wichtiges Kriterium war die Gewährleistung der Zukunftssicherheit der so geschaffenen Entwicklungsplattform.

Und Tschüss: Drew Paroski war noch bis zum März 2015 als Software-Entwickler maßgeblich an der Entwicklung von HHVM und Hack beteiligt; in Ermangelung einer neuen Herausforderung bei Facebook widmet er sich nun beim Start-up MemSQL seiner Leidenschaft für schnelle Datenbanken (Bild 4)

Das Hauptziel von HHVM bestand darin, das akute Geschwindigkeitsproblem von PHP für Facebook zu lösen. Die Geschwindigkeit von PHP 5.3 war ja, gelinde gesagt, nicht gerade berauschend. Daher war das HHVM-Projekt für Facebook aus damaliger Perspektive äußerst sinvoll. Das Ziel bestand darin, sowohl dynamisch erzeugte Websites schnell genug zu erzeugen als auch Facebooks stark steigende Server-Belastung kosteneffizient zu handhaben. Hierzu musste PHP JIT das Webportal Facebook.com mindestens so effizient handhaben wie HPHPc, Facebooks eigene Ausfüh- ▶

Der DJ der HipHop VM: Jay Parikh, der Vizepräsident der Entwicklungsabteilung bei Facebook und inoffizieller HHVM-Evangelist, gibt bei HHVM den Ton an (Bild 5)

rungsengine für Hack Lang und PHP. Darüber hinaus galt es, HPHPi, den Interpreter, den die Facebook-Entwickler damals in ihrer täglichen Arbeit nutzten, durch etwas Kosteneffizienteres (also bei vergleichbarer Hardware deutlich Schnelleres) ersetzen.

Die Kompilierung von PHP und Hack Lang fand bei Facebook an jedem Wochentag statt und nahm im Schnitt 20 Minuten in Anspruch, das heißt, die Just-in-Time-Kompilierung musste mindestens genauso schnell wie bisher ablaufen, vorzugsweise natürlich schneller. Sollte die Erstellung von HHVM überhaupt Sinn ergeben, so musste dieser Geschwindigkeitsvorteil nicht nur erhalten bleiben, sondern auch deutlich verbessert werden.

Die Stealth-Attacke auf PHP

Facebooks Hack schien sich in der jüngeren Vergangenheit PHP zu einer ernsthaften Bedrohung zu entwickeln, während Beschwerden um Sicherheitsprobleme rund um Altlasten-Features von PHP laut wurden.

Der Wettbewerb von seitens HHVM hat die PHP-Entwicklergemeinde offenbar aber dazu angespornt, der Performance und den Hardwareanforderungen der beliebten Websprache eine besondere Aufmerksamkeit zuteilwerden zu lassen.

Der Gedanke, dass das Facebook-Team rund um Josh Watz mit Hack und Jay Parikh mit HHVM (Bild 5) das geschätzte PHP zum alten Eisen degradieren könnte, hat offenbar so viele PHP-Entwickler um den Schlaf gebracht, dass sich die PHP-Gemeinde voller Energie und Schwung darangemacht hat, die drohende Stealth-Übernahme abzuwenden.

Die Erleichterung der PHP-Entwicklergemeinde über die finale Verfügbarkeit von PHP 7 könnte wohl kaum größer sein.

Überzeugend: Andi Gutmans war in seiner Rolle bei Zend Technologies maßgeblich an der Erstellung der neuen PHP-Engine (PHPNG) beteiligt (Bild 6)

Neue Features in PHP 7 gibt es viele, allen voran eine neue Engine PHPNG, kurz für »PHP next generation«. Diesen hochperformanten Unterbau, der von Dmitry Stogov, Xinchen Hui und Nikita Popov entwickelt wurde, hat Zend Technologies (einer der wichtigsten Software-Infrastrukturanbieter mit Lösungen rund um PHP) beigesteuert, um die Performance von PHP-Erweiterungen zu verbessern (Bild 6). Somit wurde der Kern von PHP 7 de facto ausgetauscht, auf durchaus ähnliche Weise, wie man etwa den Motor in einem Auto auswechseln kann.

Laut Andi Gutmans, einem Mitgründer und dem einstigen Geschäftsführer der PHP-Softwareschmiede Zend Technologies (die im Übrigen unlängst durch eine Akquisition in Rogue Wave Software aufging), habe PHP in der siebten Generation einen deutlichen Performance-Boost erhalten. Diese Feststellung ließe sich als eine Untertreibung einstufen. Das Resultat in Real-World-Benchmarks ist ein Leistungssprung, der sich nur als spektakulär bezeichnen lässt.

PHP 7 mit Zend Engine 3

Nicht nur die Zend-Technologieplattform, sondern auch das ganze PHP-Ökosystem wären in Mitleidenschaft gezogen worden, wenn sich Facebook mit HHVM und Hack durchgesetzt hätte. Die Zend-Entwickler haben daher die Ärmel hochgekrempelt und entschlossen mit angepackt.

Die Zend-Entwickler konnten den wichtigsten Ursachen von Performance-Engpässen mit Hilfe von Benchmarks auf die Spur kommen. Unter Verwendung des synthetischen Benchmarks *bench.php* ließen sich zwischen den Versionen 5.6 und den Vorläufern des PHP-7-Interpreters beachtliche Leistungsverbesserungen nachmessen. Je näher man an die finale PHP-7-Fassung rückte, desto deutlicher waren die Performancesprünge messbar.

Der tatkräftige Einsatz der Zend-Entwickler und die Entschlossenheit der PHP-Gemeinde haben dem PHP-Interpreter einen beachtlichen Leistungsschub verleihen können.

Laut Rasmus Lerdorf (Bild 7) dürften die meisten heutigen PHP-Applikationen eine hundertprozentige Beschleunigung alleine durch die Umstellung auf PHP 7 erfahren. Benchmarks scheinen seine Auffassung zu belegen (Bild 8).

Möglich wurde der Performance-Boost in PHP 7 durch das Abschneiden alter Zöpfe, die konsequente Umstellung auf 64-Bit und einen sehr sparsamen Umgang mit dem Speicher. Außerdem habe sich Angaben von Lerdorf zufolge der Chiphersteller Intel für die PHP-7-Entwicklung tief in die Karten schauen lassen und den PHP-Core-Entwicklern Einblicke in die optimierte Nutzung von Hash-Linien und modernen CPU-Features wie zum Beispiel Registern gewährt.

Die ineffiziente Speicherverwaltung des PHP 5.x-Kerns von Grund auf neu zu erfinden zählte zu den schwierigsten Herausfor-

derungen. Der Speichermanager von PHP 5.x verbraucht beim Betrieb der Referenzanwendung WordPress satte 20 Prozent der CPU-Zyklen selbst (Bild 9). Mit einem derart verschwenderischen Umgang mit der verfügbaren Rechenzeit ist jetzt in PHP 7 endgültig Schluss.

Die ineffiziente Speicherzuweisungsfunktion *dlmalloc* wurde durch eine andere *malloc*-Implementierung nach dem Vorbild von *jemalloc* ersetzt. Anstatt verknüpfte Listen zu bewandern, führt der Speichermanager nun endlich *biteset*-Operationen durch. Dank einer neuen Implementierung von Hash-Tabellen konnten zahlreiche Zuteilungsvorgänge eingespart werden.

Als Resultat dieser Optimierungen fiel der Bedarf des Speichermanagers an CPU-Zyklen von erheblichen 20 Prozent auf bescheidene 5 Prozent. Die neue Implementierung verträgt sich nebenbei auch wesentlich besser mit dem CPU-Cache.

Bei all den Änderungen blieb das Verhalten des PHP-Interpreters gegenüber dem Entwickler der PHP-Webanwendung größtenteils unverändert. Von den Leistungsoptimierungen einmal abgesehen wurde dem Kern keine neue Funktionalität verliehen.

Trendsetter: Rasmus Lerdorf, der Erfinder von PHP (Bild 7)

Gründliche Modernisierung

Die Strategie des PHPNG-Redesigns durch Zend Technologies bestand darin, den schon betagten Quellcode des PHP-Cores vom Hauptentwicklungsbaum abzusplitten, um ihn einer gründlichen Modernisierung zu unterziehen (Stichwort: Refactoring).

Das Resultat ist eine PHP-Performance, die sich im Durchschnitt in etwa verdoppelt hat. Die Kunst dabei bestand darin, dies zu erreichen und trotzdem eine fast völlige Abwärtskompatibilität der PHP-Sprache sicherzustellen. All dies hat Zend Technologies in Zusammenarbeit mit dem PHP-Core-Team erzielt.

Da die Änderungen an der Zend Engine von PHP 5 zur Version PHP 7 grundlegende Modifikationen und Verbesserungen beinhalten, wurde in PHP 7 die neue Zend Engine auf die Version 3 getauft. Sie löst die Zend Engine 2, die sich in PHP 5 befand, ab.

Die Umarbeitung der Zend Engine wirkt sich unter anderem auf den Zugriff auf verschachtelte Variablen aus, wie zum Beispiel:

```
$$foo['bar']['baz']
```

oder:

```
foo::$bar['baz']
```

Eine PHP-Anweisung wie die folgende:

```
global $$foo->bar;
```

ist in PHP 5.x durchaus zulässig. Für PHP 7 müssen Sie diese aber wie folgt umschreiben:

```
global ${$foo->bar);
```

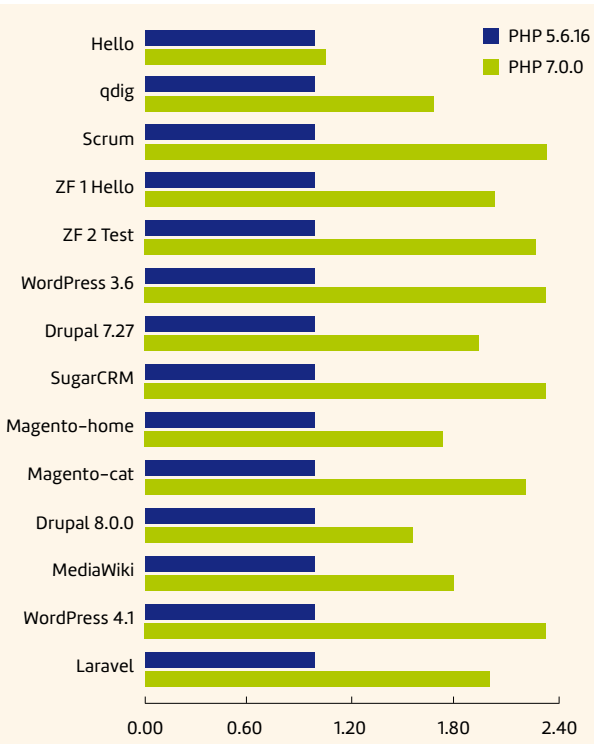
Natürlich kann man nun eine recht kontroverse Diskussion darüber führen, ob derartige Anweisungen überhaupt ihren Platz in PHP haben sollten, doch sei dieser Aspekt einmal dahingestellt, denn die Zahl der relevanten Verbesserungen ist diesmal ungewohnt hoch.

Typisierung in Hack und Rückgabetypen in PHP 7

Facebook hat Hack mit der Absicht entwickelt, eine Typprüfung in Echtzeit (statt erst zur Laufzeit) zu ermöglichen. Die meisten Programmiersprachen fallen in eine von zwei Kategorien: statisch typisierte und dynamisch typisierte Sprachen.

Statisch typisierte Sprachen (wie zum Beispiel Java) führen die Typprüfung bereits während der Kompilierung durch. Dynamisch typisierte Sprachen (wie etwa Python und PHP) nehmen sich diese Aufgabe erst zur Laufzeit vor und erkaufen sich ihre höhere Flexibilität daher durch eine wesentlich höhere Anzahl von Laufzeitfehlern zur Laufzeit. ▶

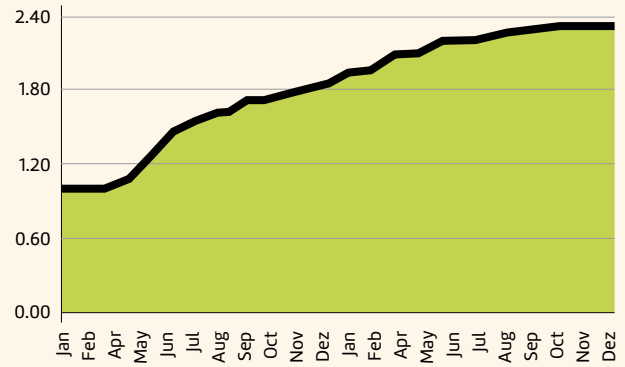
PHP 7.0.0 – Performance Improvement



Mehrheitlich: Nahezu alle PHP-Anwendungen ziehen in Benchmarks Nutzen aus der Umstellung auf PHP 7 (Bild 8)

web & mobile developer 4/2016 Quelle: Wordpress-4.1 PHP-7.0.0: 2.32

PHP 7 – WordPress 3.6.0 – Performance Evaluation



Entschlossen: Die Performance von WordPress ließ sich im Lauf der Entwicklung von PHP 7 nahezu verdoppeln (Bild 9)

web & mobile developer 4/2016

Hack Lang ermöglicht die Typüberprüfung (falls gewünscht) im Stil einer statisch typisierten Sprache, fühlt sich jedoch an wie eine dynamisch typisierte Sprache mit einer automatischen Erkennung von Fehlern der Typzuweisung in Echtzeit. Viele der zentralen Features von Hack Lang drehen sich dementsprechend auch rund um die Typisierung.

Ein Feature namens *Type Annotations* in Hack Lang erlaubt zum Beispiel die explizite Typisierung von Methoden- und Funktionsparametern, Klassenvariablen und Rückgabewerten, etwa in folgender Weise:

```
<?hh
class eineKlasse {
  const int eineKonstante = 0;
  private string $x = '';
  public function increment(int $x): int {
    $y = $x + 1;
    return $y;
  }
}
```

Explizite Typisierung ermöglicht die Nutzung typspezifischer Autocomplete-Features in IDEs und eine frühzeitige Fehlererkennung.

An den Entwicklern von PHP sind Facebooks Innovationen nicht spurlos vorübergegangen. PHP 7 bringt demnach eine Vielzahl von Verbesserungen in Bezug auf die Handhabung von Datentypen mit, umfasst jedoch derzeit – unter anderem in Ermangelung eines JIT-Compilers – noch keine Typprüfung in Echtzeit innerhalb der IDE.

Die Typbestimmung für Objekte und Felder in PHP gab es seit der Version 5 und 5.1. PHP 5 unterstützte ja auch bereits skalare Typdeklarationen. In PHP 7 kommen vier neue Typdeklarationen hinzu: *int, float, string* und *bool*. Zwei Modi wurden hierzu vorgesehen: zwingend und strikt. Die strikte Typprüfung muss allerdings erst mittels folgender Direktive aktiviert werden:

```
<?php declare(strict_types=1);
```

Diese Direktive wirkt sich ausschließlich auf die jeweilige Skriptdatei aus, dafür beeinflusst sie sowohl skalare Typdeklarationen als auch die Auswertung der Rückgabetypen einer Funktion.

Erstmals in der Version 7 von PHP können Sie Datentypen für die Rückgabewerte von PHP-Funktionen deklarieren. Ähnlich zu den Argumenttyp-Deklarationen spezifizieren Rückgabetyp-Deklarationen den Typ des Wertes, der von der Funktion zurückgeliefert wird. Es sind dieselben Typen sowohl für Rückgabetyp- als auch für Argumenttyp-Deklarationen verfügbar, zum Beispiel:

```
<?php
function arraysSum(array ...$arrays): array {
  return array_map(function(array $array): int {
    return array_sum($array);
  }, $arrays);
}
print_r(arraysSum([1,2,3], [4,5,6], [7,8,9]));
```

Das obige Beispiel liefert als Ausgabe:

```
Array
(
  [0] => 6
  [1] => 15
  [2] => 24
)
```

Listing 1: Typenvergleichsregeln in PHP

```
<?php
// Integer
echo 1 <=> 1; // 0
echo 1 <=> 2; // -1
echo 2 <=> 1; // 1

// Fließkommazahlen
echo 1.5 <=> 1.5; // 0
echo 1.5 <=> 2.5; // -1
echo 2.5 <=> 1.5; // 1

// Textzeichenketten
echo "a" <=> "a"; // 0
echo "a" <=> "b"; // -1
echo "b" <=> "a"; // 1
?>
```

Hack bietet im Übrigen einen eigenen Container-Typ für PHP-Arrays und zusätzliche Fähigkeiten der Typüberprüfung für PHP-Arrays (die zum Teil in spezialisierten Klassen bereitgestellt werden). Die offizielle Empfehlung der Hack/HHVM-Gemeinde lautet allerdings, anstelle von PHP-Arrays lieber die Hack-spezifischen Collections (Collection Classes) zu nutzen, denn die Flexibilität von PHP-Arrays überfordert derzeit die Fähigkeiten der HHVM, die Typprüfung zuverlässig umzusetzen.

Hack ignoriert zum Beispiel Fehler der Indizierung von PHP-Arrays, meldet stattdessen nicht existierende Fehler in Bezug auf Array-Schlüssel und weist unvorhersehbares Verhalten in Bezug auf die Konvertierung

int-ähnlicher Datentypen in *int*. Diese Fehler lassen sich zum Teil darauf zurückführen, dass Hack anders als PHP auf der Einhaltung des Datentyps des Array-Schlüssels beharrt. Zu den interessantesten Unterschieden zwischen Hack und PHP zählen neben der Typprüfung die Handhabung von *NULL* und die Auswahl an unterstützten Operatoren.

Typenvergleiche in PHP 7 mit dem Raumschiff-Operator

PHP 7 möchte unnötigen Quellcode-Ballast abwerfen und schneller auf den Punkt kommen, ohne dabei an Lesbarkeit zu verlieren. Mit dem neu in PHP 7 eingeführten Raumschiffoperator lassen sich Typenvergleiche in PHP 7 kurz und knackig auf den Punkt bringen, ohne dabei auf die vergleichsweise epische Länge eines voll ausformulierten If-Schleifenkonstrukts ausweichen zu müssen.

In PHP 7 gelten hierbei die bisher üblichen Typenvergleichsregeln, die es auch in PHP 5.x gab. Ein praktisches Beispiel illustriert Listing 1.

Hack-Entwickler können zwar den Raumschiff-Operator in ihrem Code nutzen, sollten sich aber bewusst sein, dass die Typprüfung bei diesem Operator nicht stattfindet, da er lediglich von HHVM und nicht von Hack selbst unterstützt wird.

Bezüglich der PHP-7-Kompatibilität musste Facebook mit Hack und HHVM eine wahre Aufholjagd aufnehmen. Doch bei der Implementierung von Operatoren liegt HHVM weit vorne. HHVM hat den Lambda-Operator (==>), Null-Safe-Operator (?->) und den XHP-Attribut-Access-Operator (->:) eingebaut, die es in PHP 5.6.x und PHP 7.x nicht gibt.

Der Lambda-Operator

Der Lambda-Operator zählt zu den Besonderheiten von Hack, die in PHP 7 nicht existieren. Mittels eines Konstrukts namens Lambda lässt sich in Hack eine Closure (also ein Funktionsabschluss) erzielen. Lambdas in Hack bringen es besser auf den Punkt; der Code wirkt weniger langatmig als in PHP 7. Lambdas werden typischerweise in Kombination mit Funktionen wie *array_filter()* oder *array_map()* verwendet. Ein Lambda wird durch den Lambda-Operator ==> eingeführt. Links von ==> befindet sich die Liste der Argumente und zur Rechten entweder ein Ausdruck oder eine Liste von Anweisungen, die in {}-Klammern eingeschlossen ist. Lambdas lassen sich verketten und haben eine niedrige Rangfolge (sie kommen daher ohne Klammern aus).

Die traditionelle PHP-Syntax unter Verwendung von *function ($name)* und *return* illustriert Listing 2. Der Funktionsabschluss lässt sich mit dem Lambda-Operator in Hack wie in Listing 3 gezeigt implementieren.

Listing 2: Funktionsabschluss in PHP-Syntax

```
<?hh
namespace Hack\in\PHP\Syntax;

function addLastnameTraditional(): array<string> {
  $people = array(
  "Joachim",
  "Hans",
  "Paul"
  );
  return array_map(function ($name) {
  return $name . "Mayer";
  }, $people);
}

function run(): void {
  var_dump(addLastnameTraditional());
}
run();
```

Listing 3: Funktionsabschluss in Hack-Syntax

```
<?hh

namespace Hack\in\Hack\Syntax\mit\dem\Lambda\
Operator;

function addLastname(): Vector<string> {
  $people = Vector {
    "Joachim",
    "Hans",
    "Paul"
  };
  return $people->map($name ==> $name . "Mayer");
}

function run(): void {
  var_dump(addLastname());
}
run();
```

Der null-safe-Operator

Der *null-safe*-Operator in Hack (?->) funktioniert wie der gewöhnliche Objektoperator in PHP (->), außer dass er mühsame *NULL*-Überprüfungen vor dem Aufruf einer Methode oder einer Eigenschaft in Bezug auf ein Objekt überflüssig macht. Sollte zum Beispiel *$obj* in der folgenden Anweisung

```
$obj?->foo($x, $y);
```

NULL sein, wird der gesamte Ausdruck als *NULL* evaluiert, statt einen fatalen Fehler zu verursachen. Einen fatalen Fehler zur Laufzeit erhalten Sie dennoch, wenn Sie mit diesem Operator, der nicht *NULL* ist, eine nicht existierende Methode auf ein Objekt loslassen.

Der *null-safe*-Operator darf nur für Leseoperationen benutzt werden; Schreiboperationen mit diesem Operator sind strikt untersagt. Falls Sie den Operator mit einer Funktion verwenden möchten, müssen Sie sicherstellen, dass sich diese als *nullable* typüberprüfen lässt. ▶

Für den Zugriff auf die Attribute eines XHP-Objekts bietet Hack einen eigenen Operator (->:). Anstatt:

```
$variable = $xhp_object->getAttribute('attributeName');
```

kommt in Hack der XHP-Operator für den Zugriff auf Attribute wie folgt zum Einsatz:

```
$variable = $xhp_object->:attributeName;
```

Dadurch stellen Sie sicher, dass die Typprüfung wie gewünscht stattfindet.

Im Übrigen sei PHP-Entwicklern der Rat ans Herz gelegt, ihren XHP-Code im globalen Namensraum einzunisten; Facebook ist das Problem bekannt, nur gibt es derzeit leider keine bessere Lösung.

Null-Koaleszenz-Operator in PHP 7 und Hack

Sowohl PHP 7 als auch Facebooks Hack Lang unterstützen den Null-Koaleszenz-Operator (dargestellt mit zwei Fragezeichen ??), der eine syntaktische Finesse darstellt. Perl-Entwicklern ist der Operator unter dem Namen Logical Defined-Or bekannt.

Der Null-Koaleszenz-Operator in PHP 7 kommt anstelle von *isset()* mit einem ternären Operator zum Einsatz, wie das nachfolgende Beispiel zeigt:

```
$benutzername = isset($_GET['user']) ? $_GET['user'] :
'niemand';
```

Dieselbe Zuweisung lässt sich nun wie folgt implementieren:

```
$benutzername = $_GET['user'] ?? 'niemand';
```

Der Null-Koaleszenz-Operator im obigen Beispiel sorgt dafür, dass der Wert von *$_GET['user']* erfasst wird (falls gesetzt und ungleich *NULL*); andernfalls liefert er den Wert *'niemand'* zurück.

Die Koaleszenz lässt sich im Übrigen verketten. Der folgende Einzeiler erfasst in der Variablen den ersten definierten Wert von *$_GET['user']*, *$_POST['user']* oder *'niemand'*, jeweils von links nach rechts:

```
$benutzername = $_GET['user'] ?? $_POST['user']
?? 'niemand';
```

Listing 4: Eintrittspunkt im Modus partial

```
<?hh

require_once 'lib/autoload.php';

function main() {
  setup_autoload();
  do_initialization();
  $controller = find_controller();
  $controller->execute();
}

main();
```

Listing 5: Anonyme Klassen in PHP 7

```
<?php
interface Logger {
  public function log(string $msg);
}

class Application {
  private $logger;

  public function getLogger(): Logger {
    return $this->logger;
 }

  public function setLogger(Logger $logger) {
    $this->logger = $logger;
  }
}

$app = new Application;
$app->setLogger(new class implements Logger {
  public function log(string $msg) {
   echo $msg;
  }
});

var_dump($app->getLogger());
?>
```

Eine Reihe von PHP-Features wurde in Hack nicht implementiert, doch lassen sich diese in HHVM dennoch nutzen. Das Fehlen jeglicher Unterstützung für diese Features betrifft nur ihren Hack-Code *(<?hh)*, nicht jedoch PHP in HHVM.

Die Interoperabilität Ihres Hack-Codes *(<?hh)* mit Ihrem PHP-Code (*<?php*) bleibt von diesen Einschränkungen unberührt, denn HHVM selbst unterstützt PHP ja ohne jegliche Einschränkungen. So können Hack- und PHP-Skripts miteinander reden.

Hack Lang bietet beispielsweise keinerlei Unterstützung für die Ausdrücke *isset()*, *empty()* und *unset()*. Die Aufgabe von *isset()* in Bezug auf Variablen übernimmt in Hack der Typchecker. Um zu prüfen, ob ein Element in einem Array existiert, gilt es, *array_key_exists()* aufzurufen. Anstelle von *unset()* setzen Sie die betreffende Variable auf *NULL*.

Referenzen in PHP

Eines der wichtigsten Features, das bei Hack außen vor blieb, sind Referenzen, ein Konzept von fundamentaler Bedeutung. Referenzen beeinflussen die Handhabung von Variablen, die

Listing 6: Klassen erweitern

```
<?php

class SomeClass {}
interface SomeInterface {}
trait SomeTrait {}

var_dump(new class(10) extends SomeClass implements
SomeInterface {
  private $num;

  public function __construct($num)
  {
    $this->num = $num;
  }

  use SomeTrait;
});

// Die hieraus resultierende Ausgabe würde
// folgendermaßen aussehen:

object(class@anonymous)#1 (1) {
  ["Command line code0x104c5b612":"class@
  anonymous":private]=>
  int(10)
}
```

Mechanismen der Funktionsaufrufe und -Ausgabe, Speichermanagement und vieles andere. Die Unterstützung für Referenzen würde eine gesunde statische Analyse enorm erschweren. Die Übergabe einer Variablen an eine Funktion via Referenz läuft in der Praxis darauf hinaus, dass der Typchecker unmöglich feststellen kann, was mit dieser Variablen genau jetzt passieren darf.

Ineffiziente Typprüfung

Die Tragweite der Typableitung (Typinferenz) beschränkt sich nämlich auf den lokalen Umfang der betreffenden Funktion. Daraus folgt aber, dass sich eine Typprüfung nur mit Ach und Krach (sprich: extrem ineffizient) implementieren ließe, schon ganz davon zu schweigen, dass ein treffsicheres, korrektes Resultat ohnehin nicht gewährleistet werden könnte.

Erschwerend kommt hinzu, dass jegliche Zugriffe auf eine Variable, bei der es sich um eine Referenz handelt, eine zusätzliche Pointer-Dereferenzierung (im Vergleich zum Zugriff auf eine gewöhnliche Variable) und somit einen zusätzlichen Speicherzugriff erfordern. Aus diesen Gründen werden Referenzen in Hack einfach ignoriert.

Die Anweisung *global* ist in Hack eben aus diesem Grunde verboten: weil ihre Implementierung intern auf Referenzen aufsetzt.

Im Modus *partial* in Hack haben Sie die Möglichkeit, das Feld *$GLOBALS* ohne Referenzen auszulesen und zu beschreiben. Viele Entwickler verlassen sich hierauf als einen Workaround für das Fehlen einer globalen Anweisung.

Im Modus *strict* meldet sich Hack mit einem Fehler zurück: Die Variable *$GLOBALS* sei unzulässigerweise undefiniert. Als eine syntaktische Verschönerung von *$x =& $GLOBALS ['x']* führt Hack die Anweisung *global $x* ein.

Die Ausführung einer jeden Webanwendung beginnt mit Top-Level-Code. Daraus ergeben sich in Hack einige Komplikationen, denn die Ausführung von Top-Level-Code im Modus *strict* von Hack ist generell verboten; im Modus *partial* ist sie zwar erlaubt, wird jedoch nicht typüberprüft.

Ersteres hängt damit zusammen, dass jeglicher Top-Level-Code einen globalen Gültigkeitsbereich besitzt, sodass der Zugriff auf eine scheinbar lokale Variable in der Tat einen Zugriff auf eine globale Variable darstellt. Ist der betreffende Top-Level-Code nicht auf den globalen Kontext angewiesen, können Sie ihn innerhalb einer Funktion einschließen; andernfalls müssen Sie den PHP-Code grundlegender umschreiben, um ihn als Hack auszugeben.

Darüber hinaus benötigt jede Hack-Anwendung als Eintrittspunkt mindestens eine Datei, die im Modus *partial* ausgeführt wird und als ein Gateway zur eigentlichen Programmlogik dient (Listing 4), damit dieser Teil Ihrer Webanwendung im *strict*-Modus aus der Typprüfung Nutzen ziehen kann.

Array-Konstanten zur Laufzeit definieren

Wollte man in PHP zur Laufzeit eine benannte Konstante nutzen, so war man bisher in PHP 5.x auf Skalar- und *NULL*-Werte eingeschränkt. In PHP 7 können Sie erstmals auch *array*-Werte definieren, zum Beispiel:

```
bool define ( string $name , mixed $value [, bool $case_
insensitive = false ] )
```

In PHP 5 musste die Konstante *value* einen skalaren Wert *(integer, float, string, boolean* oder *NULL)* annehmen. In PHP 7 sind erstmals auch Arrays zugelassen. Ungewohnterweise lassen sich Array-Konstanten in PHP 7 nun auch zur Laufzeit definieren. Ein Praxisbeispiel veranschaulicht dies:

```
<?php
define('SPRACHEN', [
  'C',
  'PHP',
   'Java'
]);

echo SPRACHEN[1]; // gibt "PHP" aus
?>
```

Es ist außerdem möglich, Konstanten vom Typ Ressource zu definieren, allerdings wird diese Vorgehensweise nicht empfohlen, da sie ein unvorhersehbares Verhalten Ihrer Anwendung zur Folge haben kann.

Anonyme Klassen für dedizierte Wegwerfobjekte haben sich die PHP-Entwickler schon lange gewünscht. Beabsichtigt man, ein Objekt ohnehin wegzuwerfen, kann man sich ▶

die voll ausgeformte Klassendefinition auch getrost ersparen und damit wertvolle Zeit bei der Entwicklung gewinnen (Listing 5).

Sie können Argumente über ihren Konstruktor empfangen, andere Klassen erweitern, Interfaces implementieren und Traits nutzen, so wie sich dies auch mit einer normalen Klasse erzielen ließe (Listing 6).

Ist eine anonyme Klasse in PHP 7 innerhalb einer anderen Klasse eingeschlossen, erhält sie keinerlei Zugriff auf private oder gar geschützte Methoden oder Eigenschaften der externen Klasse. Damit die anonyme Klasse diese privaten oder geschützten Methoden oder Eigenschaften der externen Klasse nutzen kann, muss sie die äußere Klasse erweitern.

Hierzu übergeben Sie die privaten oder geschützten Methoden oder Eigenschaften der externen Klasse an den Konstruktor der anonymen Klasse. Die praktische Umsetzung illustriert Listing 7.

Die PHP-7-Generation verbessert die Sicherheit

In PHP 7 fand endlich die längst überfällige Trennung von sicherheitsbedenklichen Altlasten statt. Features, die in PHP 5.x als deprecated vermerkt wurden, sind nun entfallen. So wird unter anderem die Zuweisung von *new* durch Referenzieren entfernt; es ist nur noch eine ganz gewöhnliche Zuweisung möglich. Anstelle von:

```
$obj =& new Klassenname;
```

gilt also:

```
$obj = new Klassenname;
```

Aufrufe nichtstatischer Methoden relativ zum Gültigkeitsbereich aus einem inkompatiblen *$this*-Kontext heraus (seit PHP 5.5 als veraltet markiert) wurden abgeschafft (zuvor lösten sie einen *E_STRICT*-Fehler aus).

Die Unterstützung für magische Anführungszeichen wurde bereits in PHP 5.4 zum alten Eisen gelegt. Aus Gründen der Rückwärtskompatibilität konnte man aber diese Altlasten-Funktionalität mittels *set_magic_quotes_runtime* und *magic_quotes_runtime* bisher noch erzwingen. In PHP 7 ist damit nun endlich Schluss, denn diese beiden Funktionen wurden – ersatzlos, versteht sich – gestrichen und lösen jetzt nur noch einen fatalen Fehler aus.

Anstelle von *mcrypt_ecb, mcrypt_cbc, mcrypt_cfb* und *mcrypt_ofb* stehen Webentwicklern jetzt *mcrypt_encrypt* und *mcrypt_decrypt* zur Verfügung. Statt *mcrypt_generic_end* kommt jetzt *mcrypt_generic_deinit* zum Einsatz. Die Funktionen *datefmt_set_timezone_id* und *IntlDateFormatter::setTimeZoneID* werden durch *datefmt_set_timezone* beziehungsweise *IntlDateFormatter::setTimeZone* ersetzt.

In PHP 7 wird außerdem die SQL-Erweiterung *mysql_** (seit PHP 5.5 als veraltet markiert) nicht mehr geboten. Das war auch höchste Zeit, denn diese mittlerweile ziemlich angestaubte Erweiterung basiert auf dem Funktionsumfang von MySQL 3.23 aus dem Jahre 2001 und bietet bei Weitem nicht den Funktionsumfang aktuellerer Versionen. Die Umstellung auf *mysqli_** oder PDO ist eine Aufgabe von höchster Dringlichkeit. Zudem bedeutet PHP 7 nun endlich auch das Ende der Unterstützung von PHP4-Konstruktoren.

Zu den sicherheitsrelevanten Verbesserungen in PHP 7 zählen unter anderem ein Generator von Zufallszahlen *random_int()* und ein Generator von zufälligen Textzeichenketten, *random_bytes()*, die beide als kryptografisch vertrauenswürdig gelten.

Listing 7: Private oder geschützte Methoden

```
<?php

class Outer
{
  private $prop = 1;
  protected $prop2 = 2;

  protected function func1()
  {
    return 3;
  }

  public function func2()
  {
    return new class($this->prop) extends Outer {
      private $prop3;

      public function __construct($prop)
      {
        $this->prop3 = $prop;
      }

      public function func3()
      {
        return $this->prop2 + $this->prop3 +
        $this->func1();
      }
    };
  }
}

echo (new Outer)->func2()->func3();
```

Endgültiger Abschied von Altlasten

In PHP 5 blieben PHP4-Konstruktoren noch zum Teil funktionsfähig und lediglich als deprecated markiert, konnten jedoch schon mal ein unvorhersehbares Verhalten aufweisen. PHP 7 schafft sie nun endlich ab.

Mit einem *ini*-Schalter zum Aktivieren der Unterstützung für PHP4-Konstruktoren hätte man den Bruch der Rückwärtskompatibilität sicherlich entschärft, doch man hätte sich – so ist die Überzeugung der PHP-Entwicklergemeinde – sicherlich mehr Probleme eingehandelt, als man hätte damit lösen können. Angesichts der Tatsache, dass sich Namensraum-

klassen *(namespaced classes*, unterstützt ab PHP 5.3) mit dem Einsatz von PHP4-Konstruktoren ausschließen, war diese Änderung längst überfällig.

Ein klassisches Beispiel dafür, was passiert, wenn die benötigte Kompatibilität nicht gewährleistet ist, war die Umstellung von PHP 4 auf PHP 5.x. Obwohl PHP 5.x bis ins Jahr 2004 zurückgeht, haben sich viele Webseitenbetreiber wie auch Entwickler an PHP 4.x geklammert. Der Grund hierfür: Zahlreiche prominente Open-Source-Projekte waren für PHP 4.x entwickelt worden, und so schien es logisch, sie auch mit PHP 4.x auszuführen. Da aber so viele Server gerade aus diesem Grunde noch PHP 4.x nutzten, haben sich die betroffenen Entwickler mit der Aktualisierung ihrer Code-Basis viel zu viel Zeit gelassen. Als Resultat daraus geriet die PHP-Entwicklung ins Stocken, was die Entstehung von HHVM und Hack erst ermöglicht hatte.

Dass die Gemeinde der PHP-Entwickler die Abschaffung dieser Konstruktoren in PHP 7 befürwortet hat, hat die Entwickler von PHP-Bibliotheken wie PEAR zum Handeln bewogen. So bringt PEAR in der Version 1.10.1 endlich Kompatibilität zu PHP 7, wenn auch auf Kosten der Rückwärtskompatibilität bis einschließlich Version 5.3.

Die Entwickler des PHP-Cores haben sich daher gezielt dafür eingesetzt, den Schwerpunkt auf die gelungene Umsetzung von Neuerungen zu fokussieren, statt sich allzu sehr mit den Altlasten zu befassen. Die Bedrohung durch eine feindliche Übernahme durch das von Facebook unterstützte HHVM- und Hack-Ökosystem war einfach zu groß. Im Rückblick hat sich gezeigt, dass es sich als klug erwiesen hat, den frontalen Angriff von Facebook mit einer hochperformanten und äußerst gelungenen PHP-7-Version zu kontern.

In PHP 7 dürfen alternative öffnende und schließende Tags nicht mehr genutzt werden. Diese Tags führten zu kuriosem Code. Vorerst bleiben uns diese in HHVM erhalten.

Einige PHP-Bibliotheken (darunter die PEAR-Bibliothek) müssen in PHP 7 in einer neuen Version eingesetzt werden.

Viele Open-Source-Projekte haben die Verbesserungen der Version 5.x von PHP verpasst. Als es dann aber hieß, die Weiterentwicklung und die Unterstützung von PHP 4.x seien eingestellt worden, konnte die Umstellung auf PHP 5.x plötzlich gar nicht mehr schnell genug gehen, wie zum Beispiel im Fall der PEAR-Bibliothek.

Jetzt ist die PHP-7-Kompatibilität die normalste Sache der Welt, und die Entwicklergemeinde weint der fehlenden Rückwärtskompatibilität zu PHP 4 keine Träne mehr nach. Doch nicht alle Erweiterungen liegen bereits in einer PHP-7-kompatiblen Version vor.

Drum prüfe (zuerst), wer sich an PHP 7 binden möchte, ob die benötigten Bibliotheken bereits angepasst wurden.

SELinux-Probleme beim Umstieg auf PHP 7

Das unerwartete Auftreten vieler Probleme mit PHP-7-Applikationen hängt mit der Einführung von Security-Enhanced Linux (SELinux) zusammen. Wer auf PHP 7 setzt, nutzt im Fall von Linux üblicherweise auch eine halbwegs aktuelle Version des Betriebssystems, und so ist plötzlich mit SELinux ein neues Sicherheitskonzept aktiviert.

Wird der PHP-Applikation nicht der richtige Sicherheitskontext zugewiesen, läuft möglicherweise erst einmal gar nichts. Da die Fehlermeldungen praktisch nichtssagend sind und auch die Logdateien nicht immer eindeutig auf eine Lösung hindeuten, sind viele Administratoren von Webanwendungen schlicht ratlos.

Zu den wichtigsten Gründen für das fehlerhafte Verhalten zählen vor allem zwei Faktoren:

- die Anwendung fehlerhafter SELinux-Sicherheitslabels auf Dateien und Verzeichnisse,
- die Anwendung fehlerhafter SELinux-Regeln auf Unix-Ressourcen wie zum Beispiel TCP-Sockets.

Möchte man Licht ins Dunkel bringen, so muss man die Statusausgabe für das nicht erwartungsgemäß funktionierende PHP 7 unter die Lupe nehmen und die Log-Dateien genau inspizieren. Eine mögliche PHP-7-Fehlkonfiguration kann sich in einer ganzen Reihe verschiedener Fehler reflektieren (Listing 8). ▶

Listing 8: Fehlerprotokoll

```
# systemctl status -l php70-php-fpm.service
php70-php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php70-php-fpm.service; enabled; vendor preset: disabled)
   Active: failed (Result: exit-code) since Fri 2015-12-11 12:03:16 UTC; 5min ago
   Process: 13468 ExecStart=/opt/remi/php70/root/usr/sbin/php-fpm --nodaemonize (code=exited, status=78)
   Main PID: 13468 (code=exited, status=78)
Dec 08 12:03:16 ip-16-0-0-40 systemd[1]: Starting The PHP FastCGI Process Manager...
Dec 08 12:03:16 ip-16-0-0-40 php-fpm[13468]: [11-Dec-2015 12:03:16] ERROR: unable to bind listening socket for
address '127.0.0.1:9002': Permission denied (13)
Dec 08 12:03:16 ip-16-0-0-40 php-fpm[13468]: [11-Dec-2015 12:03:16] ERROR: FPM initialization failed
Dec 08 12:03:16 ip-16-0-0-40 systemd[1]: php70-php-fpm.service: main process exited, code=exited, status=78/n/a
Dec 08 12:03:16 ip-16-0-0-40 systemd[1]: Failed to start The PHP FastCGI Process Manager.
Dec 08 12:03:16 ip-16-0-0-40 systemd[1]: Unit php70-php-fpm.service entered failed state.
Dec 08 12:03:16 ip-16-0-0-40 systemd[1]: php70-php-fpm.service failed.
```

In diesem Beispiel ist das Betriebssystem nicht in der Lage, das TCP-Listening-Socket zu binden:

```
ERROR: unable to bind listening socket for address
'127.0.0.1:9002': Permission denied (13)
```

Um zu ermitteln, ob SELinux überhaupt aktiv ist und damit das unerwartete Fehlverhalten von *php-fpm* ausgelöst haben könnte, hilft der Befehl *getenforce* weiter. Als Ausgabe erhalten Sie in der Kommandozeile die Bestätigung *enforcing*. Ist die strikte Einhaltung von SELinux-Regeln aktiviert *(enforcing)*, können Sie diese mit *setenforce 0* vorübergehend abschalten und den PHP-Dienst neu starten.

Scheint der Fehler dadurch behoben zu sein, können Sie davon ausgehen, dass Sie die Ursache im Großen und Ganzen auch schon gefunden haben. Im ersten Schritt sei Ihnen dann geraten, die SELinux-Label im Dateisystem zu korrigieren. Versetzen Sie SELinux wieder mit *setenforce 1* in den operativen Modus *enforcing*. Navigieren Sie zum Ablageort der Konfigurationsdateien von PHP im Verzeichnis Ihrer PHP-Installation:

```
cd /etc/opt/remi/php70/php-fpm.d
```

Listen Sie die Unix-Zugriffsrechte samt aktueller SELinux-Label wie folgt auf:

```
# ls -laZ
drwxr-xr-x. root system_u:object_r:etc_t:s0 .
drwxr-xr-x. root root system_u:object_r:etc_t:s0 ..
-rw-r--r--. root root system_u:object_r:etc_t:s0
www.conf
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0
www.webundmobile.de.conf
```

Die Korrektur der SELinux-Label gelingt nun folgendermaßen:

```
chcon -R system_u:object_r:etc_t:s0 www.webundmobile.
de.conf
```

Danach starten Sie den SAPI-Dienst neu. Doch damit hat sich die Liste der erforderlichen Anpassungen noch lange nicht erschöpft, denn auch Systemressourcen wie TCP-Sockets unterliegen der Zugriffsverwaltung durch SELinux.

Ein SELinux-Modul für php-fpm

Die SELinux-Richtlinien sind in vielen Fällen schlicht falsch eingestellt, und deshalb kann sich die betreffende Webapplikation nicht wie erwartet verhalten. Möchte man aus diesem Teufelskreis ausbrechen, so gilt es, die Änderungen zu ermitteln, die der jeweilige Dienst oder die Webapplikation zum korrekten Ablaufen benötigt. Dazu muss man SELinux temporär mit *setenforce 0* in den *permissive*-Modus schalten.

In diesem Modus zeichnet SELinux all diejenigen Aktivitäten auf, die es sonst im restriktiven Modus geblockt hätte. Zur Suche nach den relevanten Änderungen nutzen Sie *grep* mit einer Pipe-Umlenkung, um das gewünschte Ergebnis in eine Datei umzulenken:

```
grep php-fpm /var/log/audit/audit.log | audit2allow -m
phpfpm > phpfpmlocal.tmp
```

Die resultierende Datei – im Beispiel *phpfpmlocal.tmp* – sollten Sie jetzt unter die Lupe nehmen, um zu ermitteln, ob das automatisch erzeugte Resultat sinnvoll ist. Im Beispiel sieht die *phpfpmlocal.tmp*-Datei wie in Listing 9 aus.

Danach lassen Sie *audit2allow* erneut laufen, um das gewünschte Modul zu erstellen:

```
grep php-fpm /var/log/audit/audit.log | audit2allow -m
phpfpmlocal
```

Zur Aktivierung der SELinux-Richtlinien nutzen Sie den Befehl *semodule -i phpfpmlocal.pp*. Aktivieren Sie den strikten Modus von SELinux mit *setenforce 1*. Zur Sicherheit können

Listing 9: phpfpmlocal.tmp

```
# cat phpfpmlocal.tmp
module phpfpm 1.0;
require {
  type tor_port_t;
  type unreserved_port_t;
  type hugetlbfs_t;
  type httpd_t;
  type httpd_sys_content_t;
  class process execmem;
  class tcp_socket name_bind;
  class dir write;
  class file { write append };
}

#============= httpd_t ==============
#!!!! This avc can be allowed using the boolean
#'httpd_unified'
allow httpd_t httpd_sys_content_t:dir write;

#!!!! This avc can be allowed using the boolean
#'httpd_unified'
allow httpd_t httpd_sys_content_t:file append;
allow httpd_t hugetlbfs_t:file write;

#!!!! This avc can be allowed using the boolean
#'httpd_execmem'
allow httpd_t self:process execmem;
allow httpd_t tor_port_t:tcp_socket name_bind;

#!!!! This avc can be allowed using the boolean
#'nis_enabled'
allow httpd_t unreserved_port_t:tcp_socket
name_bind;
```

Sie mit *getenforce* nachprüfen, ob SELinux wirklich aktiv ist. Der PHP-Dienst *php70-php-fpm* verfügt nun über korrekte SELinux-Sicherheitsregeln zum Starten und sollte sich jetzt nicht mehr weigern:

```
systemctl start php70-php-fpm.service
```

Um sich einen Überblick über den Status von PHP zu verschaffen, genügt:

```
systemctl status -l php70-php-fpm.service
```

Leider ist es noch nicht alles. Beide Dienste, sowohl *php-fpm* als auch NGINX, müssen miteinander über ein TCP-Socket (bei weniger belasteten Websites alternativ über ein Unix-Socket) kommunizieren. Hierzu benötigen beide Dienste ungehinderten Zugriff auf das gewünschte Socket. Die soeben durchgeführte Prozedur müssen Sie also noch für NGINX wiederholen.

Ein SELinux-Modul für NGINX

NGINX benötigt Zugriff auf dasselbe TCP-Socket, auf dem PHP-FPM lauscht. Hierzu setzen Sie am besten das *audit2allow*-Utility ein, sodass Sie die hierzu passenden Nachrichten in den Log-Dateien wie folgt unter die Lupe nehmen:

```
grep nginx /var/log/audit/audit.log | audit2allow
```

Die resultierende Ausgabe könnte zum Beispiel folgendermaßen aussehen:

```
#============= httpd_t ==============
#!!!! This avc can be allowed using one of the these
#booleans:
# nis_enabled, httpd_can_network_connect
allow httpd_t unreserved_port_t:tcp_socket name_connect;
```

Im nächsten Schritt bemühen Sie wieder *grep* und richten die Ausgabe über eine Pipe-Umlenkung an *audit2allow*. Dies könnte wie folgt aussehen:

```
grep nginx /var/log/audit/audit.log | audit2allow -m
nginx
```

Die resultierende Ausgabe könnte zum Beispiel diese Form annehmen:

```
module nginx 1.0;
require {
  type httpd_t;
  type unreserved_port_t;
  class tcp_socket name_connect;
}

#============= httpd_t ==============
#!!!! This avc can be allowed using one of the these
# booleans:
```

▶

```
# nis_enabled, httpd_can_network_connect
allow httpd_t unreserved_port_t:tcp_socket name_connect;
```

Zu guter Letzt lesen Sie das Policy-Modul mit *semodule -i nginx.pp* auch ein und aktivieren Sie mit *setenforce 1* den strikten SELinux-Modus wieder. Damit NGINX die neue Konfiguration zur Kenntnis nimmt, ist ein Neustart des NGINX-Webservers angebracht:

```
systemctl restart nginx.service
```

Abschließend prüfen Sie mit # *getenforce* noch nach, ob SELinux die Einhaltung der Richtlinien auch tatsächlich erzwingt. Daraufhin erhalten Sie eine Bestätigung.

Möchten Sie sicherstellen, dass alle gewünschten und nur die gewünschten SELinux-Richtlinien plangemäß Anwendung finden, können Sie mit dem Befehl *semodule –l* die geladenen SEL-Module auflisten lassen.

Zu guter Letzt müssen Sie jetzt noch dafür Sorge tragen, dass der PHP-Interpreter auf Ihre PHP-Applikation zugreifen kann und nicht durch SELinux gehindert wird.

SELinux-Richtlinien für den Zugriff auf PHP-Applikationen

Korrekte Zugriffsrechte sind für die Lauffähigkeit einer Webapplikation von entscheidender Bedeutung und eine Voraussetzung für die Gewährleistung der Sicherheit. Auf einem System mit SELinux muss auch die Zugriffskontrolle durch SELinux korrekt eingestellt werden. Dies erfolgt durch die Zuweisung der sogenannten SELinux-Labels zu allen Objekten im Dateisystem.

SELinux kann den PHP-Interpreter nämlich auch dann am Zugriff auf Webverzeichnisse und PHP-Skripts hindern, wenn die Unix-Rechte korrekt gesetzt sind. Weder *chown* noch *chmod* können hierbei Abhilfe schaffen.

Neuerdings wird SELinux nicht mehr als bloßer Stolperstein angesehen, der der IT-Abteilung Hindernisse in den Weg stellt, sondern als eine notwendige und überaus nützliche Absicherung gegen Hackerangriffe.

Eigentlich kommt SELinux von Hause aus mit einem Grundwissen, welche Security-Kontexte es anzuwenden beziehungsweise bei Bedarf wieder herzustellen gilt. Leider sieht es bei der Dokumentation eher mager aus. Wo die SELinux-Dokumentation die Anwender im Stich lässt, müssen Sie selbst Detektivarbeit leisten, indem Sie die bestehenden Vorgaben auslesen und eigene Richtlinien daraus stricken.

Zum Auflisten der aktuellen Zuweisungen von SELinux-Sicherheitslabels im aktuellen Verzeichnis nutzen Sie den Befehl *ls –laZ*. Die Ausgabe könnte zum Beispiel wie folgt aussehen:

```
drwxr-x---. benutzer1 nginx unconfined_u:object_r:httpd_
sys_content_t:s0
www.webundmobile.de
```

Um zu ermitteln, welche Einstellungen Ihre Webanwendung benötigt, können Sie beispielsweise die vorhandenen SELinux-Richtlinien für WordPress untersuchen. Hierzu kommt mit *grep* ein Evergreen unter den Linux-Kommandos zum Einsatz:

```
grep wordpress /etc/selinux/targeted/contexts/files/
file_contexts
```

Die Ausgabe fördert die voreingestellten SELinux-Kontextlabels zutage:

```
/usr/share/wordpress/.*\.php --
system_u:object_r:httpd_sys_script_exec_t:s0

/usr/share/wordpress/wp-includes/.*\.php --
system_u:object_r:httpd_sys_script_exec_t:s0

/usr/share/wordpress-mu/wp-content(/.*)?
system_u:object_r:httpd_sys_rw_content_t:s0

/usr/share/wordpress/wp-content/uploads(/.*)?
system_u:object_r:httpd_sys_rw_content_t:s0
```

Listing 10: Installation von HHVM auf Ubuntu 14.04

```
# Installation der add-apt-Repository
sudo apt-get install software-properties-common

sudo apt-key adv --recv-keys --keyserver
hkp://keyserver.ubuntu.com:80 0x5a16e7281be7a449
sudo add-apt-repository "deb
http://dl.hhvm.com/ubuntu $(lsb_release -sc) main"
sudo apt-get update
sudo apt-get install hhvm
```

Listing 11: Installation von HHVM auf Debian 8 Jessie

```
sudo apt-key adv --recv-keys --keyserver
hkp://keyserver.ubuntu.com:80 0x5a16e7281be7a449
echo deb http://dl.hhvm.com/debian jessie main |
sudo tee /etc/apt/sources.list.d/hhvm.list
sudo apt-get update
sudo apt-get install hhvm
```

Listing 12: Installation von HHVM auf Debian 7 Wheezy

```
sudo apt-key adv --recv-keys --keyserver
hkp://keyserver.ubuntu.com:80 0x5a16e7281be7a449
echo deb http://dl.hhvm.com/debian wheezy main |
sudo tee /etc/apt/sources.list.d/hhvm.list
sudo apt-get update
sudo apt-get install hhvm
```

```
/usr/share/wordpress/wp-content/upgrade(/.*)?
system_u:object_r:httpd_sys_rw_content_t:s0

/usr/share/wordpress-mu/wp-config\.php --
system_u:object_r:httpd_sys_script_exec_t:s0
```

Nutzen Sie diese Informationen als eine Art Richtschnur zum Bestimmen der SELinux-Kontextlabel für Ihre eigene PHP-Anwendung.

Voreingestellte SELinux-Kontextlabel

Die voreingestellten SELinux-Kontextlabels sehen eine vergleichsweise restriktive Einstellung vor, die gerade noch das Ausführen von PHP-Skripts ermöglicht, jedoch nicht mehr das Updaten. Falls Sie dennoch diese Einstellungen einspielen möchten, können Sie die Anpassung aller Verzeichnisse in einem Arbeitsschritt mit dem folgenden Befehl durchführen:

```
find www.website.tld/ -type d -exec chcon
system_u:object_r:httpd_sys_content_t:s0 {} \;
```

Zum Korrigieren von SELinux-Zuweisungen für Ihre PHP-Skriptdateien nutzen Sie diesen Befehl:

```
find www.website.tld/ -type f -regex ".*\.php" -exec
chcon system_u:object_r:httpd_sys_script_exec_t:s0 {} \;
```

Ihre PHP-Applikation wird mit dieser SELinux-Richtlinie laufen, sich jedoch weigern, Themes, Plug-ins und andere Erweiterungen zu installieren oder auch Updates einzuspielen. Falls Sie diese Funktionalität in Ihrer Applikation bereitstellen möchten, müssen Sie den betreffenden Objekten eine Richtlinie zuweisen, welche dem jeweiligen Prozess das Überschreiben dieser Objekte gestattet, zum Beispiel mittels:

```
chcon -R -v system_u:object_r:httpd_sys_rw_content_t:s0
www.webundmobile.de/
```

Zum Anpassen der SELinux-Richtlinien Ihrer Webanwendung auf das übergeordnete Dokumentverzeichnis der Website nutzen Sie den folgenden Befehl (nicht-rekursiv, da Sie SELinux-Labels der untergeordneten Objekte bereits korrigiert haben):

```
chcon -v system_u:object_r:httpd_sys_content_t:s0
www.webundmobile.de
```

Damit Sie die SELinux-Sicherheitseinstellungen für den Verzeichnisbaum Ihrer Webanwendung permanent sichern, nutzen Sie den *semanage*-Befehl, zum Beispiel:

```
semanage fcontext -a -t httpd_sys_rw_content_t "/var/
www/www.webundmobile.de(/.*)?"
```

Dadurch stellen Sie sicher, dass der Befehl *restorecon* die benötigten Anpassungen nicht wieder zurücknimmt.

HHVM auf die Probefahrt

Wer HHVM auf die Probefahrt nehmen möchte, benötigt anfangs lediglich HHVM und etwas Hack- oder PHP-Code als Startpunkt. Wer sich in HHVM einarbeiten möchte und sich bereits in PHP auskennt, kann sich die HHVM-spezifischen Anweisungen für später aufheben.

HHVM ist im Hinblick auf die zulässigen Systemanforderungen vergleichsweise flexibel. In Frage kommen praktisch alle verschiedenen Varianten von Debian und Ubuntu, für die es offizielle HHVM-Packages gibt:

- Ubuntu 15.10 (Wily Werewolf),
- Ubuntu 15.04 (Vivid),
- Ubuntu 14.04 (Trusty),
- Debian 8 (Jessie),
- Debian 7 (Wheezy).

Es sei Ihnen der Einsatz vorkompilierter, stabiler HHVM-Binaries empfohlen, denn diese werden auch in Zukunft unterstützt. Zur Verfügung stehen zurzeit unter anderem Packages für Ubuntu (14.04, 15.04 und 15.10) und Debian (7 und 8). Falls Sie Ihre Entwicklungsumgebung langfristig einfrieren möchten, sind Sie mit den sogenannten LTS-Releases (Long-Term Support) am besten bedient.

Jede dritte Hauptversion (Major-Point Release) von HHVM avanciert zur LTS-Version (also zurzeit 3.9, 3.12 und demnächst 3.15) und wird für die nachfolgenden 48 Wochen unterstützt.

Die Installation von HHVM auf Ubuntu illustriert Listing 10. Auf Debian 8 Jessie und Wheezy müssen Sie die Vorgehensweise etwas abwandeln (siehe Listing 11 für Jessie und Listing 12 für Wheezy).

Um ein LTS-Release zu beziehen, öffnen Sie die Datei */etc/apt/sources.list* in einem Texteditor Ihrer Wahl. Darin finden Sie die *deb*-Zeile:

```
deb http://dl.hhvm.com/ubuntu trusty main
```

Passen Sie diese an, indem Sie in ihr mit der *-lts*-Anweisung die gewünschte LTS-Version vorgeben, konkret also zum Beispiel:

```
deb http://dl.hhvm.com/ubuntu trusty-lts main
```

So erhalten Sie alle langzeitstabilen LTS-Updates für Ubuntus 14.04 (Trusty). Sollten Sie die Auswahl noch weiter verfeinern und lediglich alle LTS-Updates aus der 3.9-Serie beziehen möchten, genügt es, den vorigen Befehl wie folgt anzupassen:

```
deb http://dl.hhvm.com/ubuntu trusty-lts-3.9 main
```

Anders als es das Facebook-Team erwartet hatte, hat PHP nicht nur überlebt, sondern mit PHP 7 einen durchschlagenden Erfolg erzielt. Für HHVM rückte das ursprünglich avisierte Ziel, PHP abzulösen, in weite Ferne (Bild 10). Stattdessen musste HHVM klein beigeben und PHP-7-Kompatibilität bieten. ▶

Als Resultat dieses Kurswechsels bei Facebook bietet HHVM nun seit der Version 3.11.0 die erste stabile, PHP-7-kompatible Fassung (wenn erst noch unvollständig).

Der Standpunkt der PHP-Entwickler ist eindeutig: Die Altlasten gehören höchstens in den Legacy-Zweig von PHP, nämlich 5.6.16. In PHP 7 werden die Altlasten nun zu den Akten gelegt.

Die Quadratur des Kreises

Obwohl die PHP-Kernentwickler für die Einführung nicht abwärtskompatibler Änderungen in PHP 7 eine Menge vernünftiger Argumente ins Feld gebracht haben, hat das HHVM-Team offenbar eine eigene Vorstellung davon, wie PHP aussehen sollte.

Die Standardkompatibilitätsstufe in HHVM bleibt auf absehbare Zeit PHP 5.6; die Unterstützung für PHP 7 soll erst schrittweise vervollständigt werden. Die Entscheidung war im Übrigen nicht zuletzt auch ein Zugeständnis an Facebooks eigene Entwickler, die ihre persönlichen Vorlieben mit einer gewissen Entschlossenheit durchsetzen. PHP 7 hat Facebook einen kräftigen Strich durch die Rechnung gemacht.

Der größte Vorteil von HHVM, nämlich der Performancevorsprung gegenüber PHP 5.x durch eine effizientere Speichernutzung, löste sich mit der Einführung von PHP 7 quasi im Nichts auf. Angesichts der massiven Mengen von interoperablem PHP- und Hack-Code, den es für das soziale Netzwerk intern zu pflegen gilt, ist Facebook aber darauf nach wie vor angewiesen.

In der Praxis soll der Mix aus PHP 5 und PHP 7 in HHVM folgendermaßen funktionieren:

- PHP-5-Features, die laut dem HHVM-Team keine Kompatibilitätsprobleme mit sich bringen, sind immer verfügbar.
- PHP-Features, die in Version 7 abgeschafft wurden (wie zum Beispiel alternative öffnende Tags), werden auch in absehbarer Zukunft weiterhin funktionieren.
- Bei PHP-7-Features, für die es keinerlei Fallback in PHP 5 gibt, bietet Ihnen HHVM die Möglichkeit, mit Hilfe von INI-Optionen das Verhalten der Engine zu bestimmen.

Die INI-Option *hhvm.php7.all = 1* in HHVM ermöglicht es Ihnen, das PHP-7-konforme Verhalten für nicht rückwärtskompatible Features zu aktivieren. Die verschiedenen Kompatibilitätsstufen sind unter der Adresse http://j.mp/HHVMphp7 dokumentiert.

Die PHP-7-Kompatibilität von HHVM ist recht neu, und tendenziell hängt HHVM immer etwas hinterher. Wenn PHP 7.1, 7.2 et cetera erscheint, braucht das Facebook-Team immer eine gewisse Zeit, um die Neuerungen und Korrekturen in HHVM einfließen zu lassen.

Segmentation-Faults und Crashes in HHVM

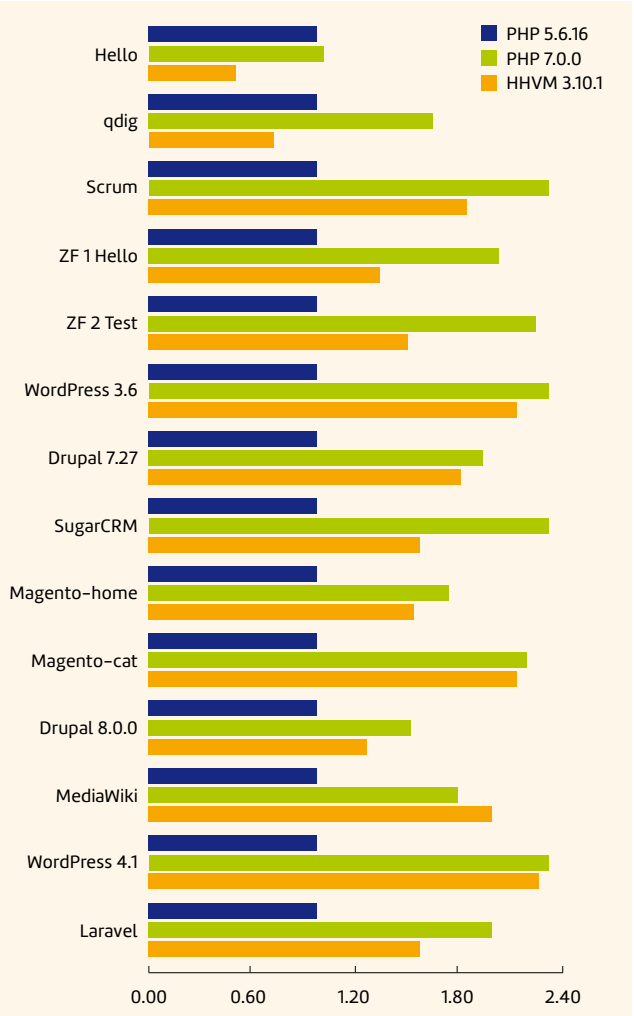
HHVM mag sich zwar bei Facebook bewährt haben, dennoch macht die Umgebung einen unfertigen Eindruck. HHVM führt zum Teil völlig kontraproduktive Innovationen ein, welche die betroffenen Entwickler an Facebooks Technologie binden. Auch sind Segmentation Faults, eine Ursache von Crashes, nicht gänzlich unbekannt. Manchmal liegen die Fehler im PHP- oder Hack-Code und manchmal in der HHVM-Umgebung.

Tritt der Fehler nur bei aktiviertem PHP-7-Modus auf und lässt er sich im PHP-5.6-Modus nicht reproduzieren, so handelt es sich dabei um einen Bug, den Sie den HHVM-Entwicklern berichten sollten. Leider kann die HHVM-Laufzeitumgebung derzeit weder einen Absturz eigenständig melden noch den Crash-Core automatisch weiterleiten.

Gelingt es, den Crash auch im Debug-Build zu reproduzieren, so ist es sehr wahrscheinlich, dass der Fehler bisher noch nicht behoben werden konnte und sich das Einreichen des Absturzberichtes für Sie lohnen könnte. Dank der so erzeugten Backtrace-Informationen dürfte das HHVM-Team viel schneller in der Lage sein, den auftretenden HHVM-Fehler zu diagnostizieren und hoffentlich auch rasch zu beheben.

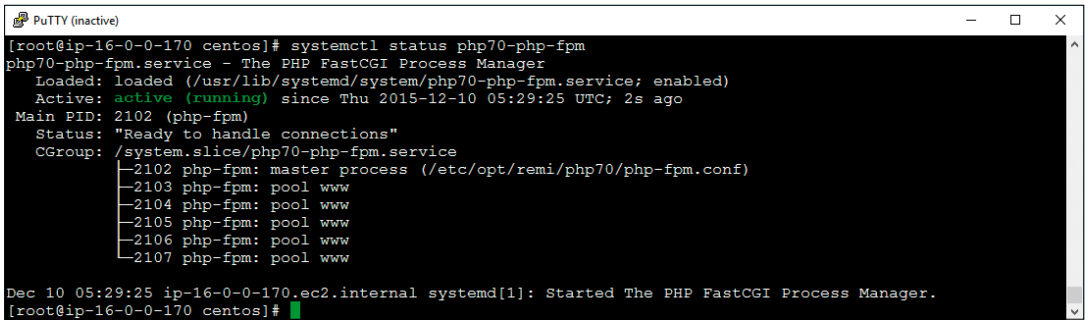
Zurzeit gibt es knapp über eintausend offiziell anerkannter Bugs der HHVM-Laufzeitumgebung. Die HHVM-Gemeinde arbeitet fieberhaft daran, diese Fehler schnellstmöglich zu beheben.

PHP 5.6.16, PHP 7.0.0 und HHVM 3.10.1



Benchmark: Die Performance von PHP 7 und HHVM 3.10/3.11 im Vergleich (Bild 10)

Quelle: Wordpress-4.1 PHP-7.0.0: 2.32



Geschafft: Kaum war PHP 7 da, schon lief es rund und blitzschnell (Bild 11)

Profile Guided Optimization (PGO)

Rasmus Lerdorf hat seine Performance-Boosts (durch den Umstieg von PHP 5.6.x auf PHP 7) auf durchschnittlich bloß 100 Prozent beziffert, während mancher Anbieter von PHP-7-Hosting-Diensten teilweise einen vielfach höheren Performance-Vorteil für sich in Anspruch nimmt.

Die PHP-Performance lässt sich, soweit es heute bekannt ist, mit folgenden Methoden stark anheben:

- Profile Guided Optimization (PGO) für PHP-basierte Web-Apps,
- Optimierungen an der in PHP 7 eingebauten Zend Engine und den dazu passenden PHP-Erweiterungen.

Umfragen zufolge nutzen nur die wenigsten PHP-Entwickler ein optimiertes und hierzu manuell kompiliertes PHP, welches auf eine bestimmte Webapplikation und ein Betriebssystem fein abgestimmt wurde. Wenn es aber auf die höchste Performance ankommt, wäre dies genau die richtige Vorgehensweise.

Damit Sie aus der Profile Guided Optimization (PGO) Nutzen ziehen, müssen Sie PHP selbst kompilieren.

Laden Sie dazu den Code von PHP herunter – aus Sicherheitsgründen direkt von der offiziellen PHP-Website: http://php.net/downloads.php. Extrahieren Sie den tar-Ball mittels *tar -xvzf php-7.x.x.tar.gz*. Normalerweise würden Sie jetzt mittels *cd* in das Verzeichnis der PHP-Quellen wechseln und die Befehle *configure*, *make* und *make install* in der Kommandozeile einsetzen, um PHP 7 zu kompilieren. Doch um den PGO-Turbobooster für PHP einzustellen müssen Sie die obige Prozedur erst noch leicht abgewandelt ausführen:

```
configure
make prof-gen
make install
```

Führen Sie die zu optimierende Zielanwendung aus, damit PHP ein Profil der Performance-Optimierung erstellen kann. Kompilieren Sie PHP dann erneut, diesmal aber mittels:

```
configure
make prof-use
make install
```

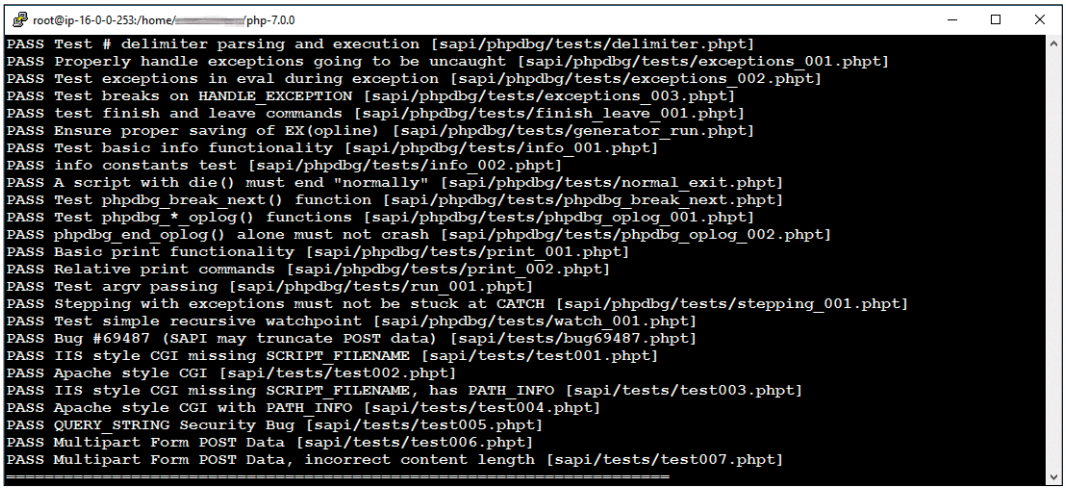
In dieser Befehlssequenz wurde *make prof-gen* durch *make prof-use* ersetzt. Als Resultat entsteht eine optimierte Edition von PHP, welche eine bestimmte Webapplikation beschleunigt ausführen kann.

Umstieg und Ausblick

Rasmus Lerdorf zufolge sollte sich halbwegs moderner Code, der innerhalb des letzten Jahrzehnts entstanden ist, problemlos an PHP 7 anpassen lassen.

Rückblickend sei Lerdorf selbst erstaunt, dass die PHP-Sprache, die er im Jahre 1995 erfunden hatte, sich zu so einem großen Erfolg entwickeln konnte. Er lässt sich mit den Worten zitieren, dass er »absolut keine Ahnung gehabt ha- ▶

Getestet: Vor dem Einsatz als Produktionsserver steht intensives Testen auf Herz und Nieren auf dem Programm (Bild 12)



Links zum Thema

- Die quelloffene Version von Hack Lang
  http://hacklang.org
- HHVM-Code
  https://github.com/facebook/hhvm
- PEAR in einer Version für PHP 7
  http://pear.php.net/package/PEAR
- Dokumentation zur PHP-Kompatibilität von HHVM
  https://docs.hhvm.com/hhvm/configuration/INI-settings#php-7-settings
- CMake
  https://cmake.org
- PHP-Projektseite
  http://de2.php.net
- PHP-7-Downloadseite
  www.php.net/downloads.php

be, dass andere Entwickler PHP überhaupt benutzen würden«. Er habe PHP primär für sich als Werkzeug entwickelt, um dynamische Websites schneller entwickeln zu können.

Rasmus Lerdorf schreibt sich den Erfolg von PHP fairerweise aber nicht nur selbst zu, sondern betont, dass er auch sehr viel seinem engsten Entwicklerkreis, also zurzeit vor allem Dmitry Stogov (von Zend Technologies), Xinchen Hui (von Laijia Technologies) und Nikita Popov, zu verdanken habe.

Trotz der beachtlichen Verbesserungen, die PHP 7 mit sich bringt, möchte sich die PHP-Gemeinde nicht auf den Lorbeeren ausruhen. Im Geiste sieht das Entwicklerteam offenbar das Hack-Team von Facebook im Rückspiegel und möchte sich daher keine Ruhe gönnen.

Jetzt steht dem PHP-Core-Team der Weg frei, um den nächsten großen Schritt anzupacken. In einer der kommenden Versionen, aller Voraussicht nach in PHP 7.1 oder 7.2, plant die PHP-Gemeinde, einen JIT-Compiler (Just in Time) nach dem Vorbild von Facebooks HHVM zu integrieren.

Diesen JIT-Compiler plant Lerdorf in PHP 7.1 oder spätestens in PHP 7.2 umzusetzen. Er räumt aber ein, dass dies eine Zeitspanne von mindestens einem Jahr oder vielleicht sogar noch etwas mehr erfordern könnte. Der Weg von PHP 5.x zu PHP 7 hat immerhin zwei volle Jahre in Anspruch genommen (Bild 11, Bild 12).

Fazit

Die Liste bahnbrechender Neuerungen in PHP 7 führt der neue PHP-Core aus dem PHPNG-Zweig von Zend Technologies an.

Das PHP-Projekt ist berühmt-berüchtigt dafür, dass es – nur um der Rückwärtskompatibilität willen – einige Sünden der Vergangenheit beibehalten hat. Mit PHP 7 soll es jetzt endlich ernst werden, denn jene Features, die in PHP 5.x als deprecated markiert wurden, werden jetzt endlich tatsächlich zum alten Eisen gelegt. Während SELinux viele PHP-Entwickler kalt erwischt hat, weil es unbemerkt mit einem Linux-Server-Update kam, verlief der Abschied von den berüchtigten PHP4-Konstruktoren nach Plan und die PHP-Gemeinde war hier bestens vorbereitet.

Einige Einbußen bei der Rückwärtskompatibilität sind bei so vielen Verbesserungen schlicht unvermeidlich, auch wenn das Gegenteil wünschenswert wäre.

Trotz all der Mühe, die sich das PHP-Core-Team mit den neuen Features gegeben hat, wie zum Beispiel den skalaren Typehints, den Rückgabetypdeklarationen, dem Raumschiffoperator, dem Null-Koaleszenz-Operator, den anonymen Klassen et cetera, werden für viele PHP-Entwickler allein die neuen Features sicherlich nicht den Ausschlag geben. Zur massenweisen Umstellung können vielmehr die verdoppelte Ausführungsgeschwindigkeit und der deutlich gesunkene Speicherbedarf beitragen, denn diese beiden Aspekte dürften für die breite PHP-Gemeinde den Umstieg auf PHP 7 auch finanziell sehr reizvoll machen.

Beim Umstieg auf PHP 7 lässt sich im Durchschnitt eine Beschleunigung auf das Doppelte erzielen. Mit der Profile Guided Optimization (PGO) können Sie eine PHP-basierte Applikation, wie zum Beispiel WordPress oder Drupal, noch weiter optimieren und das letzte Quäntchen Performance förmlich herausquetschen.

Die PHP-Core-Entwickler haben sich bemüht, die Migration von PHP 5.6.x zu PHP 7.x möglichst einfach zu gestalten. Dennoch gibt es mit SELinux einen Stolperstein, mit dem es bei der Einrichtung Ihrer Anwendungen zu rechnen gilt. SELinux sollte beim Umstieg auf PHP 7 unbedingt aktiviert und korrekt eingestellt sein. Sie sind gut beraten, Ihren PHP-Applikationen entsprechende Dokumentation beizufügen.

Die vielen überaus innovativen Features von Hack Lang können leider nicht über die Tatsache hinwegtrösten, dass es sich bei Hack/HHVM um einen unfreundlichen und vorerst gescheiterten Übernahmeversuch von PHP durch Facebook handelt. Die HHVM-Umgebung ist zudem auch noch eine Baustelle. Für PHP-Entwickler, die nicht gerade für das soziale Netzwerk arbeiten, gibt es dank der Verfügbarkeit von PHP 7 kaum noch einen triftigen Grund, PHP 5.6 in HHVM zu nutzen.

Da sich die beiden Sprachen in absehbarer Zukunft ein erbittertes Duell liefern werden, dürfen Webentwickler auf eine innovationsreiche Zukunft gespannt sein. ■

Filipe Pereira Martins und **Anna Kobylinska** sind international anerkannte IT-Berater mit dem Schwerpunkt auf Cloud-Lösungen. Sie stehen den Lesern der **web & mobile developer** gern per Twitter via **@D1gitalPro** und **@D1gitalInfo** zur Verfügung.

Developer Newsletter

Top-Informationen für Web- und Mobile-Entwickler.
Klicken. Lesen. Mitreden.

Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile

SQLITE-DATENBANK

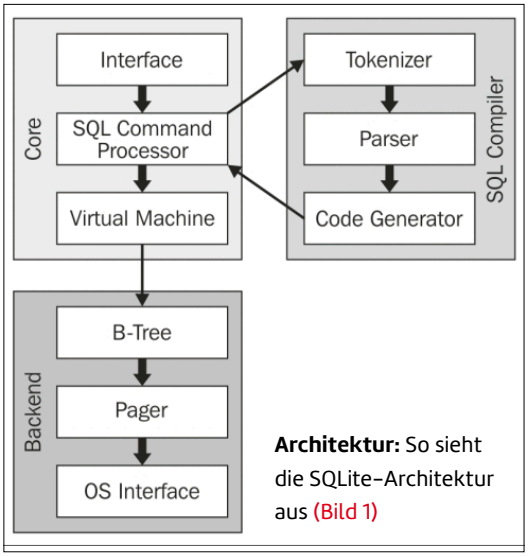
Smarte Alternative

Die SQLite-Datenbank wird in vielen mobilen Betriebssystemen verwendet.

Auch in Android, PHP oder Python wird SQLite als Standarddatenbank mit ausgeliefert. Wir stellen die verschiedenen Möglichkeiten und Einschränkungen vor. Mit sql.js existiert inzwischen sogar eine SQLite-Portierung in JavaScript.

Eine Datenbank für alles

Das Entwurfsziel von SQLite ist, den größten Teil des SQL-92-Standards zu unterstützen und eine selbstbeschreibende, transaktionale In-Process-Datenbank ohne Konfigurations- und Installationsaufwand zu entwerfen. Deswegen gibt es als Datentypen nur *NULL, INTEGER, REAL, TEXT (UTF-8, UTF-16BE, UTF-16LE)* und *BLOB*. Bei *ALTER TABLE* ist die Syntax etwas eingeschränkt, so wie es auch keine Möglichkeiten zu Zeitberechnungen gibt (Bild 1).



Architektur: So sieht die SQLite-Architektur aus (Bild 1)

Dadurch, dass der SQLite-C-Quellcode unter einer freien Open-Source-Lizenz verfügbar ist, kann SQLite unter allen Plattformen genutzt werden, auf denen ein Compiler verfügbar ist. Für die meisten Plattformen wie Windows, Linux, MacOS, Android oder .NET ist SQLite als 32-Bit-DLL verfügbar. Auf derselben Seite werden einfache Werkzeuge wie eine Kommandozeilen-Schnittstelle (CLI) oder ein Speicherplatz-Analysierer angeboten. Es können auch kommerzielle Erweiterungen (SEE, CEROD, ZIPVFS) zum Verschlüsseln und Komprimieren der Datenbank verwendet werden.

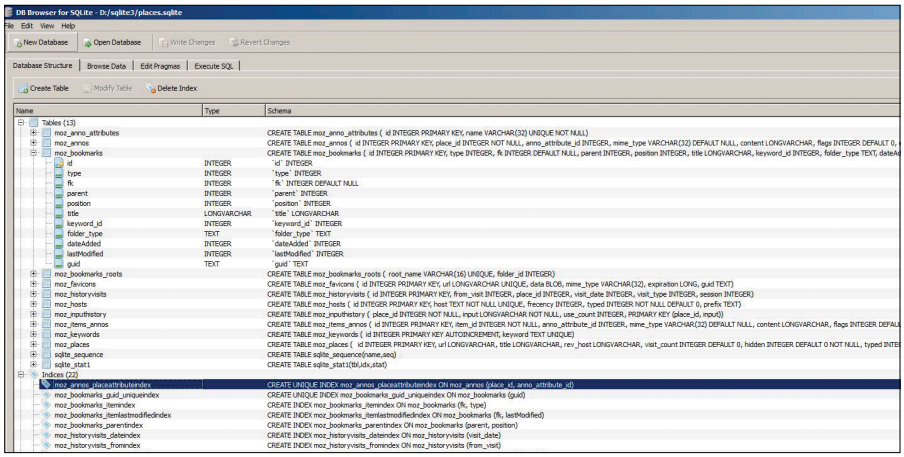
Eine SQLite-Datenbank besteht aus einer binären Datei mit der Endung *.sqlite* und bis zu neun weiteren temporären Dateien, die für die interne Verarbeitung oder die Transaktionsprotokollierung zuständig sind.

Dadurch, dass das SQLite-Dateiformat portabel ist, kann die Datenbank zwischen unterschiedlichen Betriebssystemen und Prozessorarchitekturen ausgetauscht werden. Es ist möglich, eine SQLite-Datenbank nur im Hauptspeicher zu halten, ohne diese auf Platte zu sichern.

Die einfachste Möglichkeit, mit einer SQLite-Datenbank zu arbeiten, ist mit dem SQLite-Browser (Bild 2) oder mit SQLiteSpy, die beide als separate Anwendungen vorliegen, oder mit dem SQLite-Manager (Bild 3), der als Firefox-Add-on installiert wird.

Über eine grafische Oberfläche kann man jeweils Abfragen stellen und eine Datenbank mit Tabellen erstellen. Die UIs bieten aber auch die Möglichkeit, die Datenbankstruktur zu analysieren, zu komprimieren, zu exportieren oder CSV-Daten in eine Tabelle zu importieren.

Es gibt jedoch auch für viele Programmiersprachen entsprechende Zugriffsbibliotheken oder die Möglichkeit, über Standardprotokolle wie ODBC, JDBC oder ADO.NET auf die Datenbank zuzugreifen.



Browser: Arbeiten mit dem SQLite-Browser (Bild 2)

Einschränkungen

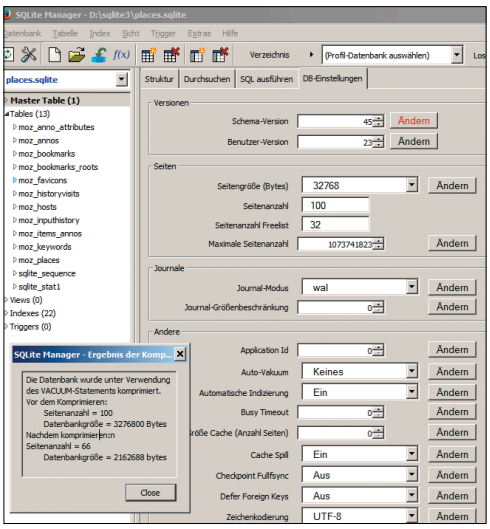
Standardmäßig gibt es sowohl bei der Datenbank, beim Blob-Datentyp oder bei SQL-Befehlen Grenzen. Diese Werte können jedoch mit der Funktion *sqlite3_limit()* angepasst werden, sodass man Datenbanken bis zu einer Größe von 140 Terabyte erzeugen kann, wenn man die maximal mögliche Seitengröße bei der Erstellung verwendet. Meistens fährt man jedoch mit den Standardwerten gut.

Mit SQLite Version 3 gingen einige Änderungen einher, etwa ein neues, mit früheren Versionen inkompatibles Dateiformat, URF-16-Codierung und

die Verwendung von 64-Bit-Werten für ROWIDs. Aktuell ist die Version 3.10.2. Um eine Datenbank von einer 2.8er-Version in eine neue zu konvertieren, führt man den Befehl *sqlite OLD.DB .dump | sqlite3 NEW.DB* aus.

Eine Datenbank weiß alles

Dadurch, dass SQLite intern von vielen bekannten Programmen verwendet wird, die Datenbank selbst jedoch keinen Sicherheitsmechanismus unterstützt, ist diese ein beliebtes Ziel für neugierige Zeitgenossen. Manchmal kann diese Eigenschaft jedoch auch hilfreich sein, wenn man zum Beispiel in Firefox, Skype oder Chrome Einträge sichern oder kaputte Datenbanken wiederherstellen möchte. Neben Anleitungen gibt es im Internet auch Werkzeuge, wie das kommerzielle Forensic Toolkit for SQLite, um die Daten aus verschiedenen SQLite-Anwendungsdatenbanken schnell auszuwerten.



Manager: So präsentiert sich die Oberfläche des SQLite-Managers (Bild 3)

Bei Firefox finden sich die Datenbanken im Verzeichnis *C:\Users\<UserName>\AppData\Roaming\Mozilla\Firefox\Profiles (Windows)* oder */home/$USER/.mozilla/firefox/$PROFILE.default* (Linux). Bei den Datenbanken sind Cookies, Search, Formhistory, Downloads, Places oder Sign-ons am interessantesten. Über *SELECT * FROM moz_bookmarks* können Sie sich Ihre gespeicherten Lesezeichen oder über *SELECT * FROM moz_places order by last_visit_date DESC* die letzten von Ihnen besuchten Seiten anzeigen lassen.

Datenbank optimieren

Zum schnellen Arbeiten mit SQLite bietet sich das Kommandozeilenwerkzeug an. Sie verbinden sich mit der Datenbank durch *sqlite3.exe places.sqlite*. Die dann möglichen Befehle können Sie sich mit *.help* anzeigen lassen.

Mit *.tables* können Sie alle Tabellen in der Datenbank und mit *.databases* die Pfade zur aktuellen und den damit verbunden Datenbanken anzeigen lassen. Mit *.dump* wird die ganze Datenbank mit Inhalten ausgegeben, was wir oben schon für die Migration von einer früheren Version verwendet haben. Bei *.schema* wird nur das Datenbank-DDL-Schema ausgegeben, mit *.fullschema* zusätzlich der Inhalt mit *INSERT*-Befehlen. Es lassen sich mit *.indices* nur die Befehle zum Anlegen von Indizes ausgeben. Die Beschreibung für eine Tabelle kann mit *PRAGMA table_info(moz_bookmarks);* ausgegeben werden. Wenn man vorher die Statistiken dafür mit *.stats on* aktiviert hat, bekommt man noch weitere Informationen, etwa über den Speicherplatzverbrauch. Mit *PRAGMA auto_vacuum = 2;* kann die automatische Komprimierung eingeschaltet werden. Um belegten Speicher wieder zur Verfügung zu stellen, kann man mit *VACUUM;* die Datenbank komprimieren. Um zu verhindern, dass gelöschte Werte ausgelesen werden, kann mit *PRAGMA secure_delete = 1;* deren Platz mit Nullen überschrieben werden.

Mit *PRAGMA integrity_check;* oder *PRAGMA quick_check;* kann man Inkonsistenzen in der Datenbank beseitigen lassen. Schlussendlich kann man auch eine Sicherung mit *.backup yymmdd.sqlite* anfertigen und mit *.quit* das Kommandozeilenwerkzeug verlassen.

Mit der SQLite-Version 3.5 wurde eine Volltextsuche (FTS3) eingeführt, die über virtuelle Tabellen umgesetzt wurde. Seit SQLite Version 3.7.4 wurde diese um eine mächtigere Variante (FTS4) erweitert:

```
CREATE VIRTUAL TABLE pages USING fts4(title, body,
compress=zip, uncompress=unzip);
INSERT INTO pages(docid, title, body) VALUES(53,
'Home Page', 'SQLite is a software...')
INSERT INTO pages(title, body) VALUES('Download',
'All SQLite source code...');
SELECT count(*) FROM pages WHERE title MATCH 'Download';
SELECT * FROM pages;
```

Die Abfrage kann dann ganz regulär mit SQL erfolgen.

Fazit

SQLite ist eine schlanke, aber recht weit verbreitete Datenbank. Durch ihre einfache Architektur ist die SQLite-Datenbank sowohl für große Datenmengen auf Servern als auch auf kleinen mobilen Geräten einsetzbar. ■

Links zum Thema

- SQLite-Website
  www.sqlite.org
- SQLite-Browser
  http://sqlitebrowser.org
- SQLite4-Infos
  https://sqlite.org/src4/doc/trunk/www/design.wiki

Frank Pientka
ist Senior Architekt bei der Materna GmbH in Dortmund. Er beschäftigt sich schon seit einigen Jahrzehnten mit Datenmengen, mobilen und Web-Anwendungen.

PROGRAMMIEREN MIT R

Goldgräber

Quelloffenes Statistikpaket mit exzellenten Möglichkeiten zur Datenauswertung.

Was früher die Goldminen waren, sind heute die großen Datenberge, die täglich wachsen. Ganz gleich ob Sie nun mit Big Data oder Small Data agieren, mit den richtigen Verknüpfungen der Daten finden Sie das darin verborgene Gold. Um im Goldjargon zu bleiben: Sie brauchen nur das richtige Schürfwerkzeug. Bei kleinen Datenmengen ist man schnell geneigt, zur Tabellenkalkulation, wie zum Beispiel MS Excel oder Calc in OpenOffice, zu greifen.

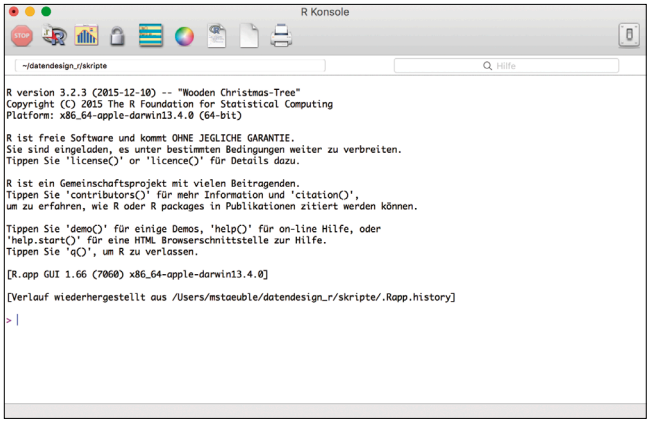
Nur irgendwann wird die Auswertung zu komplex und lässt sich auf Dauer nicht mehr warten. Da man bei komplexen Relationen auch bei Tabellenkalkulationen nicht um die Programmierung herumkommt, so kann man auch direkt zu einer ausgereiften Programmiersprache greifen – am besten zu R.

Klar, Sie könnten die Daten auch mit C# oder Java auswerten. Mit dem quelloffenen Statistikpaket R geht das aber alles viel intuitiver und mit viel weniger Codezeilen. Die Ergebnisse lassen sich dabei nicht nur in Textform darstellen, sondern auch in sehr beeindruckenden Grafiken. Damit sind Sie gewappnet für die nächste visuelle Präsentation Ihrer Zahlenreihen. Sollten Sie bereits Erfahrung in einer Programmiersprache mitbringen, so wird Ihnen der Einstieg in R leichtfallen.

Testumgebung

Die Beispiele wurden auf Mac OS X 10.11.3 mit R Project 3.2.3 und RStudio 0.99.491 umgesetzt. Das R Project steht in kompilierter Form für die Betriebssysteme Linux, Mac OS X und Windows zur Verfügung, in diesem Fall wird Version 3.2.3 für Mac OS X verwendet. Die Installation erfolgt in einem grafisch geführten Dialog. Nach wenigen Minuten kann der R-Interpreter verwendet werden. Nach dem Start präsentiert sich die R Konsole in einem grafischen Fenster (Bild 1). Alternativ zur Nutzung im grafischen Fenster kann der R-Interpreter auch direkt auf der Eingabekonsole genutzt werden.

Die grafische Oberfläche von R ist sehr minimalistisch und kann nicht mit der von kommerzieller Statistiksoftware mithalten. Da R Open Source ist, kann es auch erweitert und in andere Projekte eingebunden werden. Und einige Entwickler haben sich der Oberfläche von R angenommen und eigene Oberflächen oder gar komplette Entwicklungsumgebungen erstellt. Zwei davon sollen kurz vorgestellt werden: Rcmdr und das RStudio.



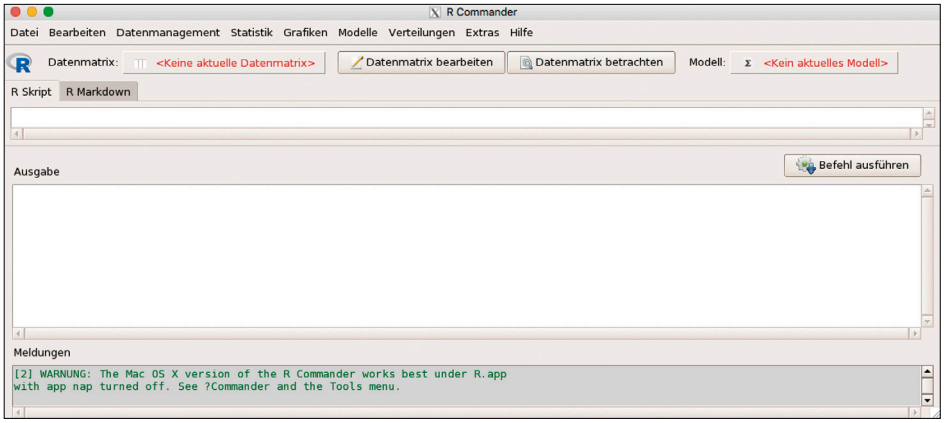
Nach dem Start von R Project präsentiert sich die Konsole (Bild 1)

Der R Commander (Rcmdr) steht als R-Paket zur Verfügung und kann direkt über einen Befehl in der R Konsole installiert werden:

```
install.packages("Rcmdr")
```

Nachdem alle notwendigen Dateien geladen sind, kann der R Commander über den Befehl *library(Rcmdr)* gestartet werden. Beim ersten Start werden die noch notwendigen Pakete geladen und installiert (Bild 2).

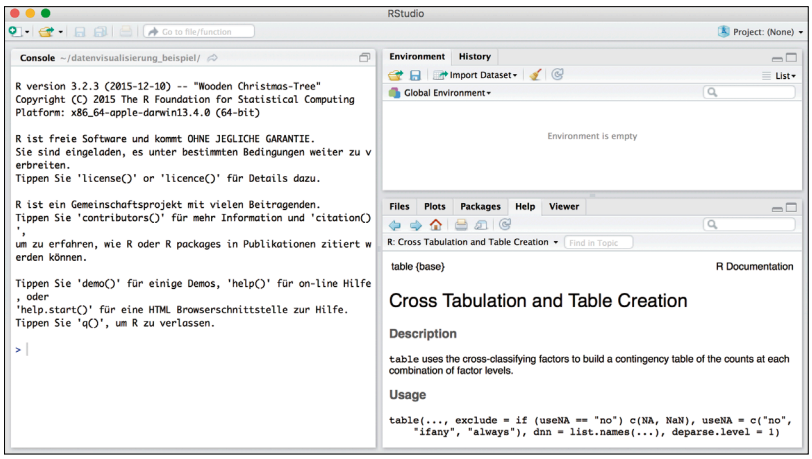
Die Alternative RStudio geht schon in Richtung ausgewachsene Entwicklungsumgebung. Es existiert eine Desktop- und Serverversion. Die Desktopversion trägt den Namen RStudio Desktop und die Serverversion analog dazu den Na-



R Commander: So päsentiert sich der R Commander nach dem Start (Bild 2)

men RStudio Server. Von beiden Varianten steht eine kostenfreie Open-Source-Version unter der APGLv3-Lizenz und eine kommerzielle Version zur Verfügung. Für den Einstieg in die R-Programmierung reicht die kostenlose Version von RStudio Desktop völlig aus.

RStudio Desktop steht für Linux, Mac OS X und Windows zur Verfügung. Für die Installation der IDE wird das bereits im Vorangegangenen installierte R Project benötigt. Für die Beispiele wird RStudio Desktop 0.99.491 für Mac OS X verwendet. Die Installation gestaltet sich ebenso einfach wie die Installation von R Project. Nach dem Start präsentiert sich zwar immer noch die grafische R Konsole, nun aber gepaart mit anderen Fenstern in einer einheitlichen Oberfläche (Bild 3).



RStudio besitzt mehrere Fenster für die Entwicklungsarbeit (Bild 3)

Grundlagen der R-Programmierung

Nachdem eine R-Umgebung installiert ist, kann es endlich an die Programmierung gehen. Bevor es zu Datentypen und Datenbereitstellung geht, werden ein paar erste Befehle in der Konsole abgesetzt. Befehle (Ausdrücke) in der Konsole werden nach Betätigen der Eingabetaste ([Return]) ausgeführt. Die Rückgabe des Ausdrucks wird dabei direkt ausgegeben:

```
> 4 + 5
[1] 9
> 11 * 3
[1] 33
```

In der Ausgabe steht vor dem Ergebnis von 4 +5 eine [1]. Dies liegt daran, dass jeder eingegebene Wert in R als Vektor interpretiert wird, und die [1] bedeutet, dass der Index des ersten in der Zeile angezeigten Wertes 1 ist. Bei einem Ergebnisvektor mit mehreren Zeilen steht an jedem Zeilenanfang der entsprechende Index. In folgendem Beispiel wird mit dem Befehl 1:52 ein Vektor mit den Zahlen von 1 bis 52 erstellt:

```
> 1:52
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[17] 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
[33] 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
[49] 49 50 51 52
```

R kann sehr gut mit Vektoren umgehen, ein neuer Vektor wird über die Funktion *c()* erstellt. Addition, Subtraktion und Multiplikation werden dabei paarweise ausgeführt. Die beiden Vektoren müssen dabei nicht aus der gleichen Anzahl an Elementen bestehen, aber ein Vielfaches voneinander sein:

```
> # Addition von zwei gleichlangen Vektoren
> c(2,4,5,8) + c(1,2,3,4)
[1]  3  6  8 12
> # Addition von nicht gleichlangen Vektoren
> c(2,4,5,8) + c(1,2)
[1]  3  6  1 10
```

Auch R-Skripts gehören kommentiert. Das Kommentarzeichen ist das #-Zeichen. Natürlich existieren auch Funktionen in R. Im obigen Fall wurde die Funktion *c()* zur Erzeugung eines Vektors genutzt. In den drei bisherigen Beispielen wurde das Ergebnis der ausgeführten Operation direkt ausgegeben. Um mehr als Basisoperationen ausführen, werden Variablen als Zwischenspeicher benötigt. Der Datentyp einer Variablen ergibt sich dabei aus der Zuweisung. Nachfolgendes Listing zeigt die Zuweisung von Werten an Variablen:

```
> # Zuweisung des Ausdrucks an die linke Variable
> a <- 3
> b <- 4
> c <- c(a,b)
> c
[1] 3 4
> # Zuweisung des Ausdrucks an die rechte Variable
> 5+9 -> a
> a
[1] 14
```

Der Zuweisungsoperator kann für beide Seiten verwendet werden. Mittels <- wird der Wert an die Variable links vom Ausdruck zugewiesen. Über -> erfolgt die Zuweisung an die Variable rechts von dem Ausdruck.

Befehlsfolgen in Skripts auslagern

Natürlich können Befehle auch in eine Datei ausgelagert werden und in die R Konsole geladen werden. Über den Befehl *source()* kann die gewünschte Datei geladen werden. Die enthaltenen Ausdrücke werden direkt ausgeführt. Nachfolgend ist ein einfaches R-Skript abgebildet:

```
#######################
## Datendefinition.r
#######################

a <- 3
b <- 4
c <- c(a,b)
```

Diese Datei kann über das Kommando *source("~/r-samples/datendefinition_01.r")* geladen werden. Anschließend kann direkt auf die darin definierten Objekte zugegriffen werden.

Datentypen

Bisher haben Sie den Vektor als Basisdatentyp kennengelernt. Ein Vektor kann nur Werte des gleichen Datentyps aufnehmen. Auf die einzelnen Werte kann dabei über einen Index zugriffen werden:

```
> # Vierstelligen Vektor erzeugen und der Variablen a
> # zuweisen
> a <- c(2,4,6,8)
> a
[1] 2 4 6 8
> # Auf den zweiten Wert von a zugreifen
> a[2]
[1] 4
```

Um einen Vektor zu erzeugen, existieren unterschiedliche Möglichkeiten. Die bereits eingeführte Funktion *c()* wurde dafür im vorigen Beispiel verwendet. Die Liste der übergebenen Werte wird dabei jeweils in einen einheitlichen Datentyp umgewandelt.

Um zu prüfen, in welchen Datentyp die einzelnen Elemente umgewandelt wurden, existiert die Funktion *typeof()*. Die Klasse eines Elements kann über die Funktion *class()* geprüft werden:

```
> typeof(1)
[1] "double"
> typeof('c')
[1] "character"
> typeof("test")
[1] "character"
> a <- c(1, 2.2, 'c', "test")
> a
[1] "1"    "2.2"  "c"    "test"
> typeof(a[1])
[1] "character"
> class(a[1])
[1] "character"
```

Die Länge eines Vektors lässt sich über die Funktion *length()* bestimmen. Um den Vektor zu verlängern, kann der Vektor über die Funktion *length()* auch vergrößert werden. Die sich dadurch neu ergebenden Speicherplätze im Vektor werden mit *NA* (not available) belegt:

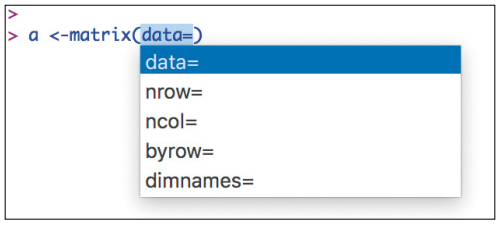
Listing 1: Erstellung von Listen

```
> testliste <- list(1, 2, "test")
> testliste
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] "test"
> adresse <- list(vorname="hans", name="meier",
strasse="allee", hausnummer=2)
> adresse
$vorname
[1] "hans"

$name
[1] "meier"

$strasse
[1] "allee"

$hausnummer
[1] 2
> adresse$vorname
[1] "hans"
> typeof(adresse$vorname)
[1] "character"
> typeof(adresse$hausnummer)
[1] "double"
```

Listing 2: Erstellung von Matrizen

```
# Matrix mit zwölf Elementen und drei Zeilen
> my_matrix <- matrix(1:12, nrow=3)
> my_matrix
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
# Matrix mit zwölf Elementen, drei Zeilen und
# Benennung der Zeilen
> my_matrix <- matrix(1:12, nrow=3,
dimnames=list(c("Zeile 1", "Zeile 2", "Zeile 3")))
> my_matrix
        [,1] [,2] [,3] [,4]
Zeile 1    1    4    7   10
Zeile 2    2    5    8   11
Zeile 3    3    6    9   12
# Matrix mit zwölf Elementen, drei Zeilen und
# Benennung der Zeilen und Spalten
> my_matrix <- matrix(1:12, nrow=3,
dimnames=list(c("Zeile 1", "Zeile 2", "Zeile 3"),
c("Spalte 1", "Spalte 2", "Spalte 3", "Spalte 4")))
> my_matrix
        Spalte 1 Spalte 2 Spalte 3 Spalte 4
Zeile 1        1        4        7       10
Zeile 2        2        5        8       11
Zeile 3        3        6        9       12
```

```
> a <- c(1,2,3)
> a
[1] 1 2 3
> length(a)
[1] 3
> #Vektor verlängern
> length(a) <- 6
# Nicht gesetzte Elemente werden
# mit NA belegt
> a
[1]  1  2  3 NA NA NA
```



Codevervollständigung in der R Konsole (Bild 4)

```
[[2]]
[1] 4

[[3]]
[1] 5

[[4]]
[1] 6
```

Die *c()*-Funktion kann nicht nur einfache Datentypen aufnehmen, sondern auch Datencontainer wie zum Beispiel eine Liste.

Eine Liste wird über die Funktion *list()* erzeugt. Beim Erzeugen des Vektors ist wichtig, dass über den Parameter *recursive* festgelegt wird, dass die einzelnen Elemente der Liste dem Vektor hinzugefügt werden sollen. Bei *TRUE* findet dies statt, ansonsten (also beim Wert *FALSE)* entsteht keine Liste, sondern ein Vektor:

```
> # Vektor wird rekursiv aus einer Liste zusammengesetzt
> a <- c(2,4,list(5,6), recursive=TRUE)
> a
[1] 2 4 5 6
> # Keine kursive Vektorerstellung.
> # Ergebnis ist deswegen eine Liste.
> a <- c(2,4,list(5,6))
> a
[[1]]
[1] 2
```

Listing 3: Erstellung von Arrays

```
> # Dreidimensionales Array mit 30 Elementen
> my_array <- array(data=1:30, dim=c(2,5,3))
> my_array
, , 1

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20

, , 3

     [,1] [,2] [,3] [,4] [,5]
[1,]   21   23   25   27   29
[2,]   22   24   26   28   30
```

Neben den bisher gezeigten Möglichkeiten kann ein Vektor auch über den *:*-Operator und die Funktion *seq()* erzeugt werden. Mittels *:* wird ein Intervall angegeben und daraus eine Folge von Werten erzeugt. Bei der Funktion *seq()* kann die Zahlenfolge noch mit Sprüngen versehen werden:

```
> # Vektor mit fünf Elementen, beginnend mit 2
> a <- c(2:6)
> length(a)
[1] 5
> a
[1] 2 3 4 5 6
# Vektor über die Funktion seq() erzeugen
# Erster Wert (from) ist 2
# Maximaler Wert (to) ist 50
# Erhöhung (by) des Wertes um 2
> a <- seq(from = 2, to = 50, by = 2)
> a
[1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34
36 38 40 42 44 46 48 50
```

Neben den Vektoren existieren auch Container zur Aufnahme von Werten unterschiedlichsten Datentyps. Ein Beispiel für solch einen Container ist die Liste. Wie beim Vektor kann auf die einzelnen Elemente über einen Index zugegriffen werden. Zusätzlich können die einzelnen Listenelemente benannt werden. Mittels *$*-Operator kann der Name zum Zugriff genutzt werden. Die Unterschiede zum Vektor sind, dass eine Liste Elemente mit unterschiedlichen Datentypen aufnehmen kann und dass keine automatische Konvertierung stattfindet (Listing 1).

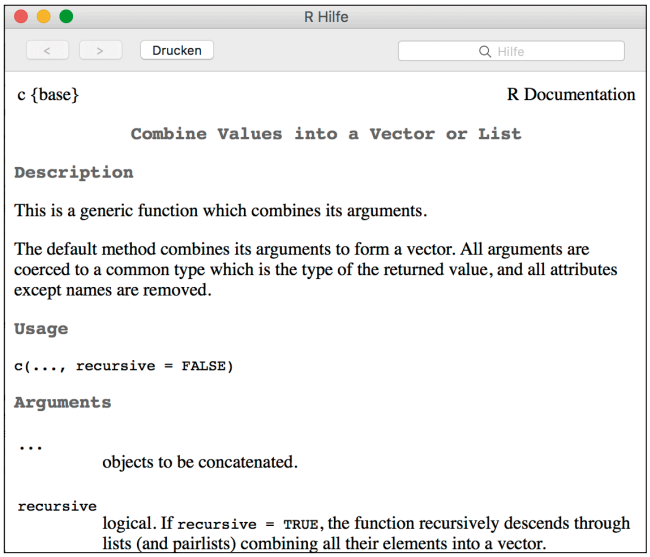
Matrizen werden unterstützt

Auch Matrizen werden direkt von R unterstützt. Dabei handelt sich um einen Vektor mit zwei Dimensionen. Bei mehr als zwei Dimensionen spricht man von einem Array.

Für die Erzeugung einer Matrix (Listing 2) steht die gleichnamige Funktion *matrix()* zur Verfügung und für ein Array die Funktion *array()*. Die Dimension des Arrays wird über den Parameter *dim* angegeben (Listing 3).

Damit man sich nicht die vielen Parameter bei komplexeren Funktionen merken muss, bietet die R Konsole eine Codevervollständigung mittels [Tab] an (Bild 4). Auch schon ohne [Tab] werden in der Fußzeile der R Konsole die Parameter der eingetippten Funktion angezeigt (Bild 5). Auch das RStudio unterstützt beim Programmieren mit einer Codevervollständigung und einem Kontextmenü (Bild 6). ▶

Neben der Codevervollständigung ist auch die integrierte Hilfe eine große Unterstützung bei der Nutzung von Funktionen. Über *help()* und Angabe der Funktion als Parameter wird die jeweilige Hilfeseite in einem neuen Fenster geöffnet (Bild 7). So ruft etwa *help(c)* die Hilfe der Funktion *c()* auf.



Die Hilfeseite wird in einem eigenen Fenster dargestellt (Bild 7)

Typen für Datenauswertung

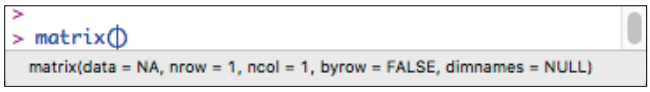
Bei der statistischen Auswertung geht es häufig um die Einordnung von Daten in Kategorien. Um die Werte eines Vektors in solche Kategorien einzuordnen, steht die Funktion *factor()* zur Verfügung (Listing 4). Die einzelnen Kategorien können über die Funktion *levels()* angezeigt werden. Die Kategorien können auch geordnet werden. Dies erfolgt über den Parameter *levels* in Verbindung mit *ordered*.

Bei der statistischen Bearbeitung von Daten kommt man um Tabellen nicht herum. Für die Speicherung von Tabellen eignet sich am besten der Datenrahmen *(data.frame)* in R. Ein Datenrahmen wird über die Funktion *data.frame()* erzeugt:

```
> # Definition einer Tabelle mit zwei Spalten
> data.frame(spalte1=c(1,2,3), spalte2=c(4,4,4))
  spalte1 spalte2
1       1       4
2       2       4
3       3       4
```

Es ist sicher nicht überraschend für Sie, dass Sie auch in R eigene Funktionen definieren können. Die Definition beginnen Sie mit dem Schlüsselwort *function*. Bei einzeiligen Funktionen müssen Sie den Funktionsrumpf nicht umrahmen. Bei mehreren Zeilen müssen Sie mit geschweiften Klammern arbeiten. Die Parameter einer Funktion können dabei auch mit einem Standardwert belegt werden:

```
> # Funktionsdefinition
> my_func1 <- function(a,b) a + b
> my_func1(2,4)
[1] 6
> my_func1 <- function(a,b) {
+ a + b
+ }
> my_func1(2,4)
[1] 6
> # Parameter b mit Wert 5 vorbelegen
> my_func1 <- function(a,b = 5) a + b
> my_func1(3)
[1] 8
```

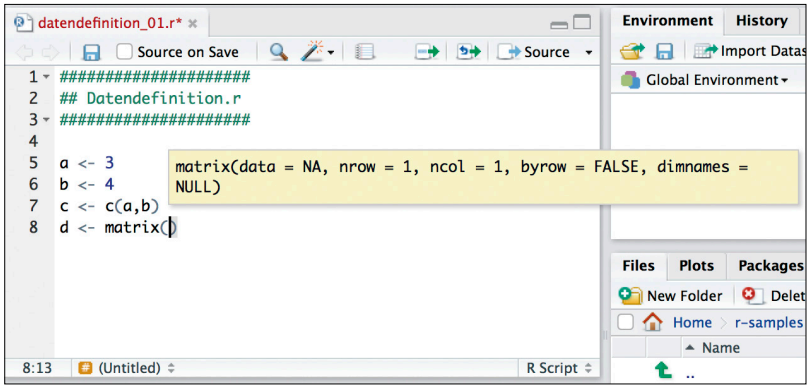


Funktionsparameter in der R Konsole werden auch in der Statuszeile angezeigt (Bild 5)

Wie in dem Beispiel zu sehen ist, muss nicht explizit ein Rückgabewert definiert werden. R liefert automatisch das Ergebnis des letzten Aufrufs zurück. Trotzdem kann man über den Aufruf *return()* explizit den Rückgabewert festlegen.

Funktionen können auch als Parameter an andere Funktionen übergeben werden. Zusätzlich unterstützt die Programmiersprache R das Prinzip von anonymen Funktionen, das heißt, Funktionen können ohne explizite Definition als Parameter übergeben werden:

```
> myfunc1 <- function (a,b,func) func(a,b)
> myfunc2 <- function (a,b) a * b
> myfunc1(2,4,myfunc2)
[1] 8
> myfunc1(2,4,function(a,b) {a+b})
[1] 6
```



RStudio: Auch RStudio beitet eine Codevervollständigung (Bild 6)

Das Prinzip von anonymen Funktionen können Sie auch zur Ergebnisüberprüfung nutzen. In den bisherigen Beispielen wurde ein Datenobjekt aufgebaut und durch Aufruf des Objekts dessen Wert ausgegeben.

Wenn Sie das Datenobjekt mit einer runden Klammer einrahmen, definieren Sie eine ano-

nyme Funktion und geben dadurch direkt den Wert des Datenobjekts aus:

```
> a <-3
> b <-4
> c <-c(a,b)
> # Ausgabe von c
> c
[1] 3 4
> # Anonyme Funktion
> (c <-c(a,b))
[1] 3 4
```

Größere Funktionen werden in R häufig in Pakete ausgelagert. Bereits bei der Installation des R Commander sind wir mit Paketen in Berührung gekommen. Um ein Paket nutzen zu können, muss es in einer lokalen Bibliothek (Verzeichnis) verfügbar sein und zusätzlich daraus geladen werden. Die aktuell geladenen Pakete und auch alle verfügbaren Pakete lassen sich anzeigen (Listing 5).

Mit der Funktion *library()* lässt sich ein Paket laden, dies haben wir bereits beim R Commander gesehen. Beispielsweise wird über *library(Rcmdr)* das lokal verfügbare Paket Rcmdr geladen.

Wenn Ihnen die lokal installierten Pakete nicht ausreichen, finden Sie im Internet eine unerschöpfliche Quelle an zusätzlichen Paketen. Die erste Anlaufstellte ist dabei CRAN (Comprehensive R Archive Network). Für Bioinformatiker sei Bioconductor empfohlen. Hier finden sich viele Pakete für die Analyse von Genomdaten. Pakete in einem frühen Entwicklungsstand findet man bei R-Forge.

Die Installation eines neuen Paketes kann über den in R enthaltenen Paketmanager (Bild 8) erfolgen, oder über den Befehl *install.packages()* auf der Konsole.

Sowohl bei der Installation über die Konsole als auch über die Oberfläche muss der Name des entsprechenden Pakets zuvor bekannt sein. Hier hilft nur eine Internetrecherche bei den oben genannten Quellen oder über Suchmaschinen. Nicht mehr benötigte Pakete werden über den Befehl *remove.packages()* entfernt.

Eigene R-Pakete erstellen

Ein Paket besteht aus einem Satz aus Dateien in einer bestimmten Verzeichnisstruktur. Über die Funktion *package.skeleton()* kann ein Verzeichnisgerüst erstellt werden:

```
> package.skeleton(name="meinPaket",
path="~/r-samples/")
Creating directories ...
Creating DESCRIPTION ...
Creating NAMESPACE ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
```

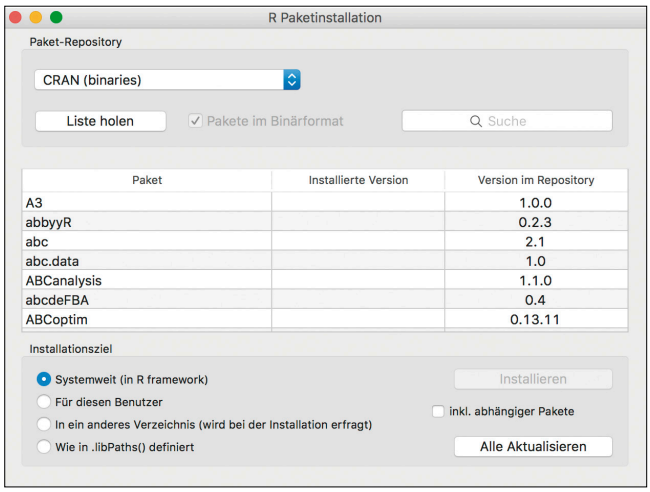
In der Umgebung vorhandene Objekte (zum Beispiel *my_array)* wurden mit ins Paket aufgenommen (Bild 9). Das Verzeichnis *man* ist für die Hilfetexte vorgesehen, *data* für Da- ▶

Listing 4: Erstellung von Arrays

```
> # Definition eines Datenvektors
> autos <- c("bmw", "audi", "audi", "mercedes",
"bmw", "bmw", "opel")
> autos
[1] "bmw"      "audi"     "audi"     "mercedes"
"bmw"      "bmw"      "opel"
> # Defintion als Faktor
> autos <- factor(c("bmw", "audi", "audi",
"mercedes", "bmw", "bmw", "opel"))
> autos
[1] bmw      audi     audi     mercedes bmw      bmw
opel
Levels: audi bmw mercedes opel
> # Ausgabe der Levels
>  levels(autos)
[1] "audi"     "bmw"      "mercedes" "opel"
> # Kategorien ordnen
> autos <- factor(c("bmw", "audi", "audi", "merce-
des", "bmw", "bmw", "opel"), levels=c("mercedes",
"bmw", "opel", "audi"), ordered=TRUE)
> autos
[1] bmw      audi     audi     mercedes bmw      bmw
opel
Levels: mercedes < bmw < opel < audi
```

Listing 5: Anzeige der lokalen Pakete

```
> # Anzeige der Pakete, die nach Start von R geladen
> # werden
> getOption("defaultPackages")
[1] "datasets"  "utils"     "grDevices" "graphics"
[5] "stats"     "methods"

> # Anzeige der geladenen Pakete
> (.packages())
[1] "stats"     "graphics"  "grDevices" "utils"
[5] "datasets"  "methods"   "base"

> # Anzeige aller verfügbarer Pakete
> # Ergebnis ist abgeschnitten
> (.packages(all.available = TRUE))
 [1] "abind"         "acepack"       "aplpack"
"arm"
 [5] "base"          "bitops"        "boot"
"car"
 [9] "caTools"       "class"         "cluster"
"coda"
[13] "codetools"     "colorspace"    "compiler"
```



Paketmanager: Die Programmiersprache R bietet dem Entwickler auch einen Paketmanager (Bild 8)

tensätze und R (wurde nicht erzeugt) für Quellcode. Die nachfolgend dargestellte Datei *DESCRIPTION* beschreibt das Paket:

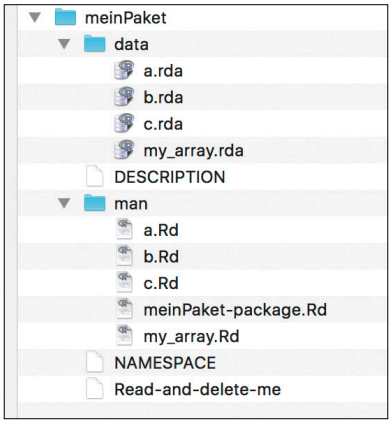
```
Package: meinPaket
Type: Package
Title: What the package does (short line)
Version: 1.0
Date: 2016-01-31
Author: Who wrote it
Maintainer: Who to complain to <yourfault@somewhere.net>
Description: More about what it does (maybe more than
one line)
License: What license is it under?
```

Auf der Kommandozeile (nicht in der R Konsole!) kann nun mittels *R CMD check meinPaket* geprüft werden, ob das Paket alle notwendigen Eigenschaften erfüllt. Im positiven Fall kann das Paket anschließend über *R CMD build meinPaket* erstellt werden.

Daten für R bereitstellen

Die Basis jedes R-Programms sind Daten. Wie bereits in den oberen Beispielen erfolgt, kann die Datenbereitstellung direkt im Programm durch Definition von Variablen erfolgen:

```
> name <- c("Meier", "Huber",
"Metzger")
> vorname <- c("Karl", "Hans",
"Anton")
> stadt <- c("Hamburg", "Bremen",
"München")
>
> adressen <- data.frame(name,
vorname, stadt)
> adressen
```



Erzeugte Dateien und Verzeichnisse durch *package.skeleton()* (Bild 9)

```
    name     vorname stadt
1   Meier    Karl    Hamburg
2   Huber    Hans    Bremen
3   Metzger  Anton   München
```

Objekte im Speicher lassen sich über die Funktion *save()* unter Angabe eines Dateinamens abspeichern. Geladen werden die Daten über *load()*. Die Daten werden dabei in einem plattformübergreifenden Binärformat abgespeichert. Auch mehrere Objekte lassen sich über die Methode *save()* abspeichern. Dabei gibt man die einzelnen Objektnamen, durch Kommas getrennt, in *save()* an. Neben einzelnen Objekten können auch alle definierten Objekte abgespeichert werden. Hierfür steht der Befehl *save.image()* zur Verfügung:

```
> # Speichern das Datenrahmens adressen
> save(adressen, file = "~/r-samples/adressen.rda")
> # Daten aus Datei laden
> load("/Users/mstaeuble/r-samples/adressen.rda")
> # Speichern von drei Objekten
> save(name, vorname, stadt ,
file = "~/r-samples/adressdaten.rda")
```

Neben dem binären R-Format unterstützt R viele weitere Formate. Objektdefinitionen können dabei in externe R-Skripts ausgelagert werden:

```
# adressen.r

name <- c("Meier", "Huber", "Metzger")
vorname <- c("Karl", "Hans", "Anton")
stadt <- c("Hamburg", "Bremen", "München")
```

Über den Befehl *source()* werden sie wieder eingelesen:

```
> source("~/r-samples/adressen.r")
> name
[1] "Meier"   "Huber"   "Metzger"
```

Daneben werden auch CSV-Dateien unterstützt. Diese können über die Funktion *read.table()* eingelesen werden. Die Funktion unterstützt etliche Parameter, darunter kann auch das Trennzeichen angegeben werden. Eine CSV-Datei kann auch mit Spaltenüberschriften versehen sein, die für den Datenzugriff genutzt werden können:

```
Name;Vorname;Stadt
Meier;Karl;Hamburg
Huber;Hans;Bremen
Metzger;Anton;München
```

Bei der CSV-Datei ist es wichtig, dass die letzte Zeile durch einen Zeilenvorschub abgeschlossen ist. Beim Einlesen der Datei ist wichtig, dass über den Parameter

header angegeben wird, ob eine Spaltenüberschrift gesetzt ist oder nicht:

```
> adressen <- read.
table("~/r-samples/adressen.csv",
header = TRUE, sep =";", flush=TRUE)
> adressen

   Name    Vorname Stadt
1  Meier   Karl    Hamburg
2  Huber   Hans    Bremen
3  Metzger Anton   München
```

R kann noch viele weitere Dateiformate lesen, unter anderem auch das Statistikformat SPSS. Auch Microsoft Excel wird über das Paket *xlsReadWrite* unterstützt. Der Zugriff auf relationale Datenbanken über ODBC, JDBC und native Datenbanktreiber ist ebenfalls möglich.

Diagramme

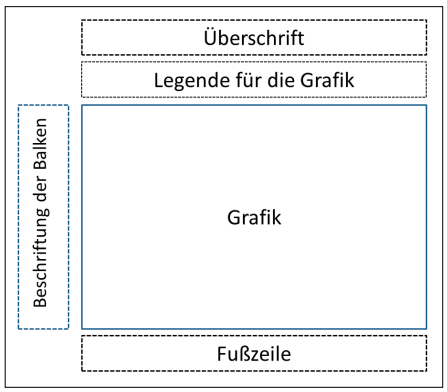
Eine besondere Stärke von R ist die grafische Darstellung von Daten mittels Diagrammen. Die Funktionen für das Zeichnen sind in Paketen gebündelt, Zwei wichtige Pakete dafür sind *graphics* und *lattice*.

Für die Einführung in die Nutzung von Diagrammen wird ein kleines Beispiel mit dem Paket *graphics* durchgeführt. In einem Balkendiagramm sollen Antworten einer Befragung dargestellt werden. Die fiktive Befragung steht unter dem Motto »Welches Medium ist Ihnen für die Informationsbeschaffung wichtig?«.

Die Befragung besteht aus drei Fragen. Auf jede Frage sind insgesamt drei unterschiedliche Antworten möglich: Stimme zu, stimme teilweise zu und stimme nicht zu. Zusätzlich ist es möglich, dass keine Antwort abgegeben wurde. Damit sind insgesamt vier Antwortmöglichkeiten für eine Frage vorhanden.

Die Fragen und die Antwortmöglichkeiten werden jeweils in einer Datendatei abgebildet. In der Datei *fragen.r* wird zusätzlich der Titel der Umfrage in der Variablen *umfrage_ueberschrift* abgelegt:

```
# Fragen der Umfrage (Dateiname: fragen.r)
# fragen.r
umfrage_ueberschrift= "Welches Medium ist Ihnen für die
Informationsbeschaffung wichtig?"
frage1 <- "Das TV-Programm ist
mir für die tägliche
Informationsbeschaffung wichtig"
frage2 <- "Das Internet ist mir
für die tägliche
Informationsbeschaffung wichtig"
frage3 <- "Die Tageszeitung ist
mir für die tägliche
Informationsbeschaffung wichtig"

fragen<-c(frage1, frage2, frage3)
```



Layout für das Beispieldiagramm (Bild 10)

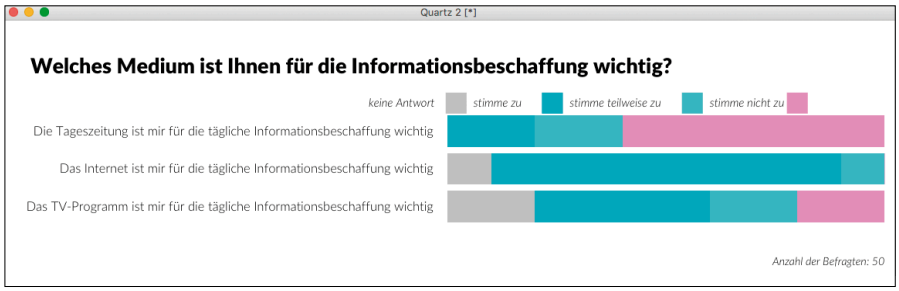
```
# Mögliche Antworten (Dateiname:
antwort_mgl.r)
# umfrage_antwort_auswahl.r
antwort_moeglichkeiten<-c("keine
Antwort","stimme zu", "stimme
teilweise zu", "stimme nicht zu")
```

Die fiktiven Antworten werden ebenfalls in einer separaten Datei mit dem Namen *antworten.r* abgelegt. Wie im nachfolgenden Listing zu erkennen, sind die drei Antworten als drei Ergebnisvektoren mit jeweils vier Elementen abgespeichert:

```
## 50 Antworten werden erwartet

antworten1 = c(10,20,10,10)
antworten2 = c(5,40,5,0)
antworten3 = c(0,10,10,30)

alle_antworten = cbind(antworten1,antworten2,antworten3)
```

Um die Antworten für die Erzeugung der Grafik verwenden zu können, werden die drei Antwortvektoren zu einem einzigen Objekt zusammengesetzt. Hierfür stehen zwei Funktionen zur Verfügung: *cbind* und *rbind*. Die Funktion *cbind* ist die Abkürzung für *column bind* und bedeutet, dass jeder Vektor eine Spalte im Ergebnis darstellt. Analog dazu steht *rbind* für *row bind,* und hier stellt jeder Vektor eine Zeile im Ergebnis dar:

```
> alle_antworten = cbind(antworten_1,antworten_2,
antworten_3)
> alle_antworten
     antworten_1 antworten_2 antworten_3
[1,]          10           5           0
[2,]          20          40          10
[3,]          10           5          10
[4,]          10           0          30
> alle_antworten = rbind(antworten_1,antworten_2,
antworten_3)
> alle_antworten
            [,1] [,2] [,3] [,4]
antworten_1   10   20   10   10
```



Das fertige Diagramm im Ausgabefenster (Bild 11)

```
antworten_2     5    40     5     0
antworten_3     0    10    10    30
```

Bevor Sie mit der Erstellung eines Diagramms beginnen, sollten Sie sich ein grobes Layout dafür überlegen.

Für das Beispiel ist dieses Layout in Bild 10 abgebildet. Das Diagramm besteht aus einer Überschrift, einer Legende, dem Grafikbereich inklusive Beschriftung der einzelnen Balken und einer Fußzeile.

Links zum Thema

- Bioconductor, Paketübersicht
  https://cran.r-project.org/packages
- CRAN (Comprehensive R Archive Network), Paketübersicht
  https://cran.r-project.org/web/packages
- Googles R-Styleguide
  https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml
- OpenOffice Calc
  https://www.openoffice.org/de/product/calc.html
- Omegahat
  www.omegahat.org
- Pretty R Syntax Highlighter
  www.inside-r.org/pretty-r
- R Project
  https://www.r-project.org
- R Project – Download
  https://cran.rstudio.com
- R-Forge
  https://r-forge.r-project.org
- Rite
  https://cran.r-project.org/web/packages/rite/README.html
- RStudio
  https://www.rstudio.com
- RStudio Desktop – Download
  https://www.rstudio.com/products/rstudio/download
- RStudio Desktop – Quellcode
  https://github.com/rstudio/rstudio
- Rcmdr
  https://cran.r-project.org/web/packages/Rcmdr/index.html
- Rkward
  https://rkward.kde.org
- Schriftart Lato
  www.latofonts.com/lato-free-fonts
- xlsReadWrite
  www.swissr.org

Die Ausgabe eines Diagramms kann sowohl am Bildschirm als auch in einer Grafikdatei oder einem PDF erfolgen. In diesem Beispiel wird das Diagramm direkt in einem Oberflächenfenster ausgegeben. Um das Diagramm aufbauen zu können, werden die bereits definierten Daten im Speicher benötigt. Hierfür werden die drei Datendateien über den Befehl *source()*, unter Angabe des Dateinamens, geladen:

```
# Fragen einlesen
source("~/r-samples/fragen.r",encoding="UTF-8")
# Antwortmöglichkeiten einlesen
source("~/r-samples/antwort_mgl.r",encoding="UTF-8")
# Antworten einlesen
source("~/r-samples/antworten.r",encoding="UTF-8")
```

Das Grafiksystem von R bietet zahlreiche Einstellmöglichkeiten. Über die Funktion *par()* können viele Parameter gesetzt werden:

```
# Grafikparameter
par(oma=c(0.0,2,4,0.75), # Äußerer Rand in Zoll
  (Unten, Links, Oben, Rechts)
  mai=c(1.6,6,0.5,0), # Rand für Diagramm in Zoll
  (Unten, Links, Oben, Rechts)
  family="Lato Light", # Schriftart die verwendet w
  erden soll
  las=1 # Ausrichtung der Achsenbeschriftungen
  (1: Horizontal)
)
```

Insgesamt gibt es vier unterschiedliche Antwortmöglichkeiten pro Frage, das heißt, ein Balken im Diagramm muss deswegen in insgesamt vier Segmente aufgeteilt werden.

Damit die Segmente voneinander unterschieden werden können, wird jedes Segment mit einer anderen Farbe gezeichnet, Die Definition einer Farbe erfolgt über die Funktion *rgb()*. Für bestimmte Farben steht auch eine Zeichenkonstante zur Verfügung:

```
# Farbe für die einzelnen Balkensegmente definieren
farbe1<-rgb(50,88,200,maxColorValue=255)
farbe2<-rgb(159,121,245,maxColorValue=255)
farbe3<-rgb(0,255,238,maxColorValue=255)
# Farben in einem Ergebnisvektor ablegen
farben<-c("lightgrey",farbe1,farbe2,farbe3)
```

Für das Zeichnen eines Balkendiagramms steht in *graphics* die Funktion *barplot* zur Verfügung. Die Funktion hat etliche Parameter. Neben den Daten (hier die Variable *alle_antworten)* wird auch die Achsenbeschriftung (hier die Variable *fragen)* und die Farben (hier die Variable *farben)* für die Balken übergeben:

```
x<-barplot( alle_antworten, # Daten
  names.arg=fragen,
  # Balken werden mit den Fragen beschriftet
  cex.names=1.1,
```

Listing 6: Diagramm erstellen

```
# Definition der Koordinaten
px<-c(1,12,28,40);
py<-rep(4,4);
tx<-c(-9,3,14,30);
ty<-rep(4,4)

# Farbmarkierung der Legende
points(px, # X-Koordinaten
  py, # Y-Koordinaten
  pch=15, # Symbol für die Datenpunkte
  cex=4, # Skalierung der Datenpunkte
  (cex = character expansion)
  col=farben, # Zu verwendente Farben
  xpd=T # Ausgabe/Clipping innerhalb des
  Diagrammbereichs
)

# Beschriftung in der Legende
text(tx,ty,antworten,adj=0,xpd=T,family="Lato
Light",font=3)

# Überschrift ausgeben
mtext(umfrage_ueberschrift, # Text
  3, # Ausrichtung, 3: Oben
  line=0, # Abstand vom Rand
  adj=0, # Ausrichtung, 0: Unten
  cex=1.8, # Skalierung für Schrift
  outer=T, # Äußern Rand nutzen
  family="Lato Black" # Schriftart
)

# Fusszeile
mtext("Anzahl der Befragten: 50",
  1, # Ausrichtung, 1:Unten
  line=2,adj=1,cex=0.95,font=3)

  # Skalierung für die Achsenbeschriftung
  horiz=T, # Balken werden horizontal gezeichnet
  border=NA, # Kein Rahmen für die einzelnen Balken
  xlim=c(0,50), # Intervall für die Abszisse (X-Achse)
  col=farben, # Farben für die Balken
  axes=F # Keine Achsen zeichnen
)
```

Als Nächstes wird nun die Legende über der Grafik gezeichnet. Dies erfolgt über die Funktion *points()*. Das Wichtigste an der Funktion ist die Festlegung der Koordinaten. Über den Parameter *pch* kann das Symbol für die Datenpunkte gesetzt werden. In diesem Fall ist es ein ausgefülltes Rechteck. Neben den ausgefüllten Rechtecken wird die jeweilige Antwortmöglichkeit positioniert. Die Ausgabe des Textes erfolgt über die Funktion *text()*. Für die Ausgabe der Überschrift wird die Funktion *mtext()* verwendet. Damit kann ein Text im Randbereich des Diagramms ausgegeben werden. Zuletzt wird ebenfalls mit *mtext()* die Fußzeile ausgegeben (Listing 6). Das fertige Diagramm wird in einem Fenster dargestellt (Bild 11).

Für eine schnelle Kontrolle ist die Ausgabe am Bildschirm gut. Für die Nutzung in einer Präsentation ist diese Ausgabe aber nicht geeignet. Hier wird entweder eine Grafikdatei benötigt oder ein PDF. Zuerst soll die erstellte Datei in eine PNG-Grafik ausgegeben werden. Sobald die Grafik nicht mehr am Bildschirm ausgegeben wird, ist es wichtig, dem System mitzuteilen, wann die Grafikausgabe beendet ist. Dies erfolgt über den Funktionsaufruf *dev.off()*; den Aufruf fügen Sie am besten am Ende des R-Skripts ein. Um nun eine PNG-Grafik zu erstellen, muss vor dem Zeichnen die Funktion *png()* mit entsprechenden Parametern aufgerufen werden:

```
png(filename="~/r-samples/balkendiagram.png",
  width=16, # Breite in Zoll
  height=12, # Höhe in Zoll
  units="in", # Einheit für Dimension
  res=300 # Auflösung
)
```

Die Erzeugung einer PDF-Datei mit der Grafik erfolgt ähnlich zur vorigen Erzeugung der PNG-Grafik. Über die Funktion *cairo_pdf()* wird die PDF-Datei erzeugt:

```
pdf_datei<-"~/r-samples/balkendiagramm.pdf"
cairo_pdf(pdf_datei,
  width=16, # Breite in Zoll
  height=12 # Höhe in Zoll
)
```

Mit dem Artikel konnte nur ein kleiner Ausschnitt aus der großen Funktionsvielfalt von R gezeigt werden. Nach etwas Arbeit mit R werden Sie feststellen, dass die Erstellung des Diagramms ein Kinderspiel ist, sobald die Daten im richtigen Format vorliegen. In der Praxis müssen die Daten in der Regel vorher noch transformiert werden. Zwar können auch in R Daten bearbeitet werden, bei großen Datenmengen sollten Sie aber auf andere Skriptsprachen wie Perl zurückgreifen und nach der Aufbereitung wieder zurück zu R gehen.

Fazit

Mit etwas Übung in R kommen Sie schnell zu beindruckenden Ergebnissen und werden Ihre Diagramme nicht mehr mit einer Tabellenkalkulation erstellen. ■

Dr. Markus Stäuble
ist Informatiker, Conference Chair der Developer Week, Fachautor und Programmleiter Make beim Franzis Verlag. Er beschäftigt er sich intensiv mit dem Thema Mobile.
@staeuble

WOOCOMMERCE MIT PHP VERWALTEN

API im REST-Stil

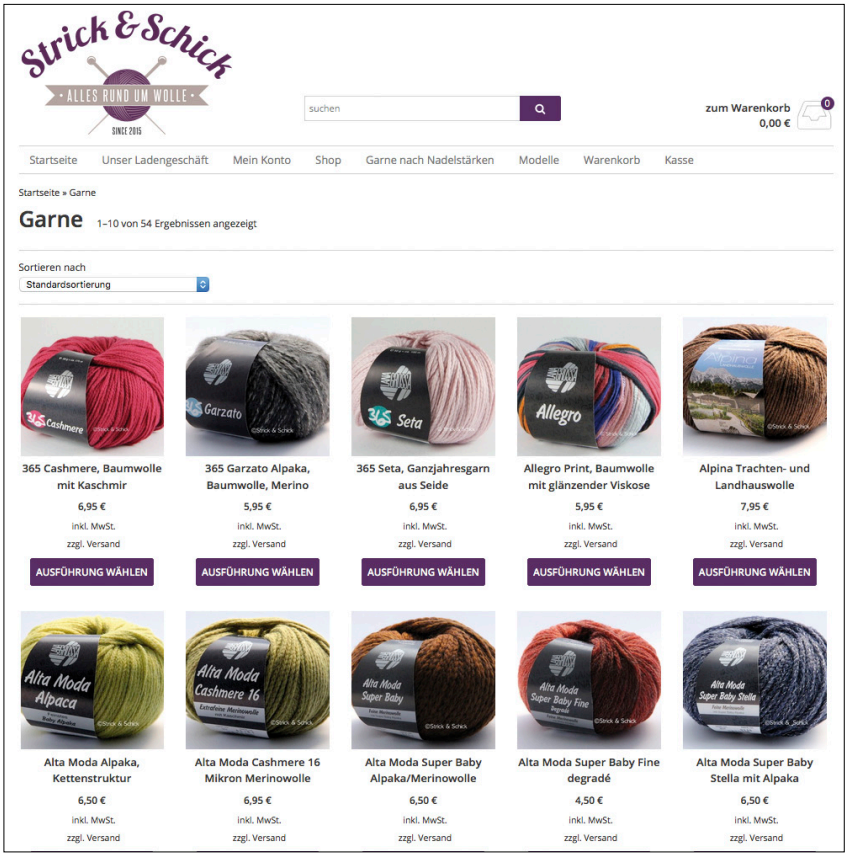
Unter den Shop-Aufsätzen zu WordPress ist Woocommerce der Hahn im Korb.

Woocommerce ist das erfolgreichste Shop-System mit WordPress als Unterbau. Auch wenn die Lösung für sich betrachtet sehr gelungen ist, kommt doch oft der Wunsch auf, eine Brücke zu anderen Anwendungen zu schlagen, etwa um die Lagerbestände mit anderen Warenwirtschaftslösungen abzugleichen (Bild 1).

Angesichts von MySQL als Datenbank-Basis von WordPress/Woocommerce ist die Versuchung sicher groß, direkt über SQL-Abfragen Informationen aus Woocommerce zu extrahieren oder einzubringen. Doch das ist kein empfehlenswerter Ansatz, weil sich der Aufbau der Daten von Woocommerce jederzeit ändern kann. Bestenfalls kommt es dann nach einem Update zu strauchelnden Skripts, schlimmstenfalls zu Fehlern in der Datenbasis.

Um auf sicherem Weg mit dem Shopsystem reden zu können, bietet Woocommerce ein API im REST-Stil an. Das API schützt dabei die Integrität des Systems.

Die Kommunikation zwischen Client und Server erfolgt per JSON-Strukturen. Wenn Sie die vorgestellte Bibliothek für den Zugriff verwenden, können Sie aber einfache PHP-Datenstrukturen nutzen und haben keinen Kontakt zu JSON.



Flexibel: Mit dem auf WordPress aufbauenden Woocommerce sind Shop-Lösungen aller Art möglich (Bild 1)

Die Schnittstelle bietet Zugriff auf alle wichtigen Elemente eines Woocommerce-Shops wie Produkte, Bestellungen, Kunden oder Gutscheine. Sie können Daten auslesen, verändern, anlegen und löschen. Damit sind Anwendungen wie die Verarbeitung von Bestellungen außerhalb von Woocommerce oder die Generierung von Kundenadressen für eine Mailingaktion einfach zu realisieren.

So legen Sie los

Die Installation erfolgt am einfachsten über Composer, mit dessen Hilfe Sie das Paket *woothemes/woocommerce-api* einbinden. Wenn Sie dann noch das Skript *vendor/autoloader.php* einbinden, können Sie die Bibliothek im Prinzip schon verwenden. Zur Absicherung des Zugangs verwendet das API einen Schlüssel. Den legen Sie in der Verwaltungsoberfläche von Woocommerce beim Menüpunkt *Einstellungen, API* und *Schlüssel/Anwendungen* an. Bei den *API - Einstellungen* sollten Sie auch gleich sicherstellen, dass die Checkbox bei *REST-API aktivieren* angehakt ist (Bild 2).

Beim Generieren der Schlüssel geben Sie als Beschreibung einfach einen Text ein, der Ihnen später dabei hilft, sich an den Zweck des Schlüssels zu erinnern. Als Benutzer wählen Sie das User-Konto, in dessen Namen der Client agieren soll. Wenn Sie später genau nachvollziehen können möchten, welche Änderungen durch das API erledigt wurden, dann legen Sie sich zuvor einen speziellen Benutzer dafür an. Das letzte Feld namens *Berechtigung* regelt dann, ob mit diesem Zugang Daten gelesen und geschrieben werden können.

Als Ergebnis des Anlegens erhalten Sie zwei Schlüssel namens Consumer Key und Consumer Secret.

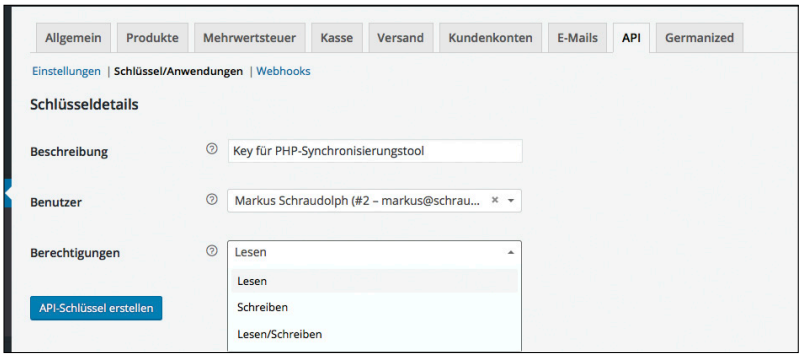
So werden die Schlüssel eingesetzt

Die beiden Keys können Sie auf verschiedene Weise dazu verwenden, Zugang zu erlangen. Neben OAuth 1.0 und der Basic Authentication bietet das API auch die Methode an, die beiden Schlüssel einfach als zusätzliche *GET*-Parameter in die Anfrage einzubauen. Damit bei dieser Variante nicht

durch Abhören der Kommunikation die Zugangsdaten ausgelesen werden können, ist sie nur bei der Verwendung einer HTTPS-Verbindung erlaubt. Über die Details der Authentifizierung müssen Sie sich beim Einsatz des Client-API keine Gedanken machen: Die Klasse fordert beim Instanzieren diese Daten von Ihnen und erledigt die Anmeldung selbstständig:

```
$client = new WC_API_Client(<url>, <key>,
<secret>);
```

Dabei ist der URL einfach die Adresse Ihres Shops mit abschließendem Schrägstrich.



Die notwendige Zugangsberechtigung zum REST-API von Woocommerce legen Sie im Shopsystem unter *Einstellungen, API* an (Bild 2)

Zusatzoptionen festlegen

Der Konstruktor der Client-Klasse verfügt noch über einen optionalen vierten Parameter. Darin übergeben Sie ein Array mit Optionen. Darunter sind recht nützliche Schalter:

- *ssl_verify:* Ist per Default aktiv und prüft bei jedem API-Call die Gültigkeit des SSL-Zertifikats.
- *debug:* Wenn Sie diesen Wert aktivieren, liefern die Klassenmethoden neben den Nutzdaten auch den kompletten Satz von HTTP-Request und HTTP-Antwort zurück. Bei Problemen ist das sehr hilfreich.
- *return_as_array:* Normalerweise erhalten Sie die zurückgegebenen Datenstrukturen als Objekte. Wenn für Sie PHP-Arrays einfacher zu handhaben sind, dann aktivieren Sie diese Option.
- *timeout:* Standardmäßig ist die Zeit, die die Bibliothek auf Antworten des Servers wartet, auf 30 Sekunden eingestellt.

Schauen wir uns einmal einen einfachen Zugriff an: Sie möchten die Daten eines bestimmtes Produkts im Shop laden. Zur Identifikation nutzen Sie die von Woocommerce vergebene ID. Dabei nehmen wir an, das Objekt *$client* sei bereits angelegt worden:

```
$answer = $client->products->get(1234);
```

Wenn die Produkt-ID existiert, erhalten Sie als Rückgabewert ein Objekt mit der einzigen Property *product*, das dann alle Merkmale des Produkts als hierarchisch gegliederte Untereigenschaften enthält (Bild 3).

Dieses Konzept zieht sich durch die gesamte Verwendung des Client-API durch: Zurückgegeben wird nie direkt das abgefragte Objekt, sondern immer eine Antwort, die den gewünschten Inhalt als Unterelement enthält, wie eben *product, order* oder *customer*. Um etwa den Titel des Produkts auszugeben, würden Sie schreiben *echo $answer->product ->title;*, und um Änderungen an einem Artikel vorzunehmen, verwenden Sie die Methode *$client->products->update (<produkt-id>,<daten>);*.

Dabei müssen Sie als zweiten Parameter nicht den kompletten Satz an Infos angeben, wie er beim Lesen zurück- ▶

Mehr als nur Artikel

In den Beispielen dieses Beitrags kommen fast nur die Daten vom Typ *product* zur Sprache.

Woocommerce bietet aber natürlich auch Zugriff auf allen anderen relevanten Informationen des Shop-Systems:

Die Bestellungen liefert Ihnen das Objekt *orders*. Für Informationen zu Rückgaben verwenden Sie *order_refunds*, und den Verlauf der Bearbeitung (in Woocommerce unter *Bestellung Anmerkungen* zu finden) bietet *order_notes*. Als ID für diese Unter-Objekte zu einer Bestellung verwenden Sie jeweils die Nummer der Bestellung selbst. Mit *coupons* fragen Sie die in Woocommerce vergebenen Gutscheine ab, legen neue an oder löschen sie. Das Objekt *customers* gibt Ihnen Zugriff auf Ihre Kundendaten. Damit könnten Sie zum Beispiel die Post- oder E-Mail-Adressen für eine Mailingaktion abrufen. Handelt Ihr Shop auch mit virtuellen Waren, bietet Ihnen *customers_downloads* Zugriff auf die Download-Informationen für die virtuellen Güter, die der Kunde erworben hat.

Informativ: Die Dokumentation des Woocommerce-API erklärt den Zweck der verschiedenen Datentypen und Felder und zeigt jeweils, ob sie vom Client geändert werden können

geliefert wird – ein Ausschnitt genügt. Ein Beispiel: Sie möchten den Bestand des Artikels mit der Produkt-ID 123 auf 10 Stück und gleichzeitig den Preis auf 9,99 setzen. Dazu würden Sie folgenden Aufruf vornehmen:

```
$d = ['stock_quantity' => 10, 'price' => 9.99];
$answer = $client->products->update(123,$d);
```

Als Antwort liefert Ihnen das API die Daten des Artikels nach dem Wirken Ihrer Änderungen zurück.

Es ist ratsam, diese Informationen auch auszuwerten, um zu überprüfen, dass die Änderungen tatsächlich Niederschlag gefunden haben. Denn es gibt Fälle, wo zum Beispiel andere Einstellungen dies verhindern. Haben Sie zum Beispiel das Häkchen bei *Lagerbestand verwalten* nicht gesetzt, dann wirkt sich das Setzen des Lagerbestands gar nicht aus. Machen Sie also eine Gegenprüfung, um Fehler dieser Art zu erkennen.

Spezielle API-Objekte

Mit *reports* bietet Ihnen Woocommerce Zugriff auf die beiden Berichtsarten Verkäufe und Topseller.

Dabei erhalten Sie allerdings keinen grafischen Bericht, sondern lediglich die Rohdaten zur Erstellung eines Berichts, etwa zur Weiterverwendung in ener Tabellenverarbeitung oder einem JavaScript-API für Charts. Jeder Datensatz beim Verkaufsreport ist beispielsweise die Tagessumme von Werten wie Bestellwert, Kunden und Menge.

Anders als etwa bei Produkten oder Bestellungen bringt ein leerer Aufruf alleine, also mit *reports->get_sales()* oder *reports ->get_top_sellers()*, nicht etwa zeitlich unlimitierte Reports über alles. Stattdessen müssen Sie zeitliche Filter angeben, um den gewünschten Zeitraum einzugrenzen. So erhalten Sie beispielsweise die Zahlen für den Dezember 2015 mit folgendem Aufruf:

```
$answer = $client->reports->get_sales(
  ['filter' =>
    ['date_min' => '2015-12-01',
    'date_max' => '2015-12-31']
  ]
);
```

Über das Objekt *index* erhalten Sie Zugriff auf allgemeine Informationen zum Shop. Dazu gehört beispielsweise der Name, die Version von Woocommerce oder Einstellungen wie Währung oder Zeitzone, mit denen das System arbeitet. Zusätzlich enthält das Index-Objekt alle Endpunkte des API zusammen mit den erlaubten HTTP-Methoden wie *GET, POST* oder *DELETE*. Der Aufruf erfolgt analog zu den normalen Shop-Objekten:

```
$answer = $client->index->get();
```

Das Objekt *webhooks* ermöglicht Ihnen die Verwaltung der gleichnamigen Einrichtung von Woocommerce. Die Webhooks dienen zur Verknüpfungen von Woocommerce-Aktionen mit externen Webaufrufen. Diese stellen quasi das Gegenstück zum API dar, weil sie dazu dienen, andere Systeme mit Woocommerce zu steuern.

Alternativen zur Abfrage per Product-ID

In Woocommerce ist das Feld *SKU* dazu vorgesehen, selbst eine eindeutige Identifikation zu vergeben. Das kann die EAN des Artikels sein oder eine andere von Ihnen verwendete Kennung – beispielsweise eine interne Bestellnummer. Woocommerce sichert die Eindeutigkeit der SKU. Wenn Sie versehentlich dieselbe Kennung einem anderen Artikel zuweisen, meldet es dies und verweigert das Speichern.

Um Produkte per SKU zu holen, gibt es eine spezielle Methode des Objekts *products*:

```
$client->products->get_by_sku('1234');
```

Neben der eindeutigen Identifizierung eines einzelnen Artikels über seine Produkt-ID oder SKU gibt es noch die Methode über Filter. Das können Attribute des Produkts sein, wie der Typ (einfach/variabel) oder die Kategorisierung.

Angenommen, Sie möchten zum Beispiel alle Produkte der Kategorie *Zubehör* abrufen. Dann verwenden Sie eine Abfrage wie diese:

```
$answer = $client->products->get(null, ['filter' =>
['category' => 'Zubehör']]);
```

Der Parameter für die ID des Artikels wird hier also auf *NULL* gesetzt und der Filter in einer Array-Struktur ausgedrückt. Übrigens erhalten Sie bei einem hierarchischen Aufbau Ihrer Kategorien damit auch Produkte, die nicht selbst als *Zubehör* getaggt sind, aber mit einer Unterkategorie davon. Objekte, die im Papierkorb liegen, liefert das API dagegen generell nicht zurück.

Weil die Antwort im Gegensatz zur eindeutigen Abfrage per ID aus mehr als einem Artikel bestehen kann, lautet der Name der zurückgelieferten Objekteigenschaft nicht *product*, sondern *products*. Diese ist als Array über mehrere Produkte ausgebildet (Bild 4).

Wollen Sie ohne jegliche Filterung arbeiten, also einfach alle Produkte geliefert bekommen, dann verwenden Sie die Methode *get()* ohne Parameter:

```
$answer = $client->products->get()
```

Allerdings erhalten Sie auch so nicht die komplette Produktpalette auf einen Rutsch.

So behandelt Woocommerce Mengen von Produkten

Denn um lange Antwortzeiten zu verhindern und die Serverlast zu minimieren, sendet das API nie mehr als zehn Elemente in einer Antwort mit.

```
object(stdClass)[16]
  public 'product' =>
    object(stdClass)[17]
      public 'title' => string 'Gallina,  Fransengarn mit Baumwolle' (length=35)
      public 'id' => int 7100
      public 'created_at' => string '2016-01-26T15:16:36Z' (length=20)
      public 'updated_at' => string '2016-01-26T15:34:48Z' (length=20)
      public 'type' => string 'variation' (length=9)
      public 'status' => string 'publish' (length=7)
      public 'downloadable' => boolean false
      public 'virtual' => boolean false
      public 'permalink' => string 'https://strick-und-schick.de/produkt/gallina-feines-fransenga
      public 'sku' => string '4033493190558' (length=13)
      public 'price' => string '5.95' (length=4)
      public 'regular_price' => string '5.95' (length=4)
      public 'sale_price' => null
      public 'price_html' => string '<span class="amount">5,95 &euro;</span>' (length=44)
      public 'taxable' => boolean true
      public 'tax_status' => string 'taxable' (length=7)
      public 'tax_class' => string '' (length=0)
      public 'managing_stock' => boolean true
      public 'stock_quantity' => int 20
      public 'in_stock' => boolean true
      public 'backorders_allowed' => boolean false
      public 'backordered' => boolean false
      public 'sold_individually' => boolean false
      public 'purchaseable' => boolean true
      public 'featured' => boolean false
      public 'visible' => boolean true
      public 'catalog_visibility' => string 'visible' (length=7)
      public 'on_sale' => boolean false
      public 'product_url' => string '' (length=0)
      public 'button_text' => string '' (length=0)
      public 'weight' => null
```

Der API-Client für Woocommerce liefert die Antworten auf Abfragen als PHP-Datenstrukturen zurück (Bild 3)

Damit Sie dennoch an die Daten von Element 11 und größer kommen, bietet das API einen Paging-Mechanismus an. Dazu setzen Sie als zusätzliches Element im Array einen Eintrag *page*.

Die zweite Seite für Ihre Abfrage aller Produkte erhalten Sie bei Bedarf also so:

```
$answer = $client->products->get(null, ['page'=>2]);
```

Sie können auch einen Filter mit dem Paging kombinieren. Das folgende Beispiel zeigt, wie Sie diese Variante realisieren können:

```
$answer = $client->products->get(null,
['page'=>2, 'filter' => ['category' =>
'Zubehör']]);
```

Als kleines Gerüst, um eine Schleife über alle Produkte zu realisieren, können Sie folgende Zeilen verwenden:

```
$filter=[];
$page=1;
while (count($products = $client->products
->get(null, ['filter' => $filter, 'page' =>
$page])->products) > 0) {
  foreach($products as $product){
    echo $product->title ."<br>";
  }
  $page++;
}
```

Das Skriptfragment holt in der *while*-Bedingung gleich die aktuelle Seite der Produkte ab und bricht erst ab, wenn die gelieferte Anzahl *0* ist, also das Ende der Liste erreicht wurde. Innerhalb der Schleife über alle Seiten finden Sie dann ▶

Die Versionen des Woocommerce-API

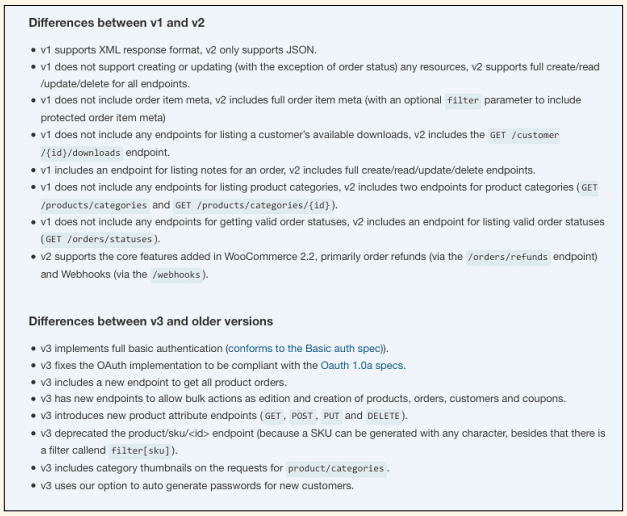
Das Shopsystem entwickelt sich weiter. Dem muss auch das REST-API Rechnung tragen, weshalb es in verschiedenen Ausführungen vorliegt.

Der angesprochene URL-Pfad entscheidet, welche Version angesprochen wird. So tritt mit einer Adresse, die etwa mit http://xy-server.de/v2 ... beginnt, die 2. Version in Aktion. Version 1 wurde mit dem Erscheinen von Woocommerce 2.5 ad acta gelegt. Das vorgestellte Client-API setzt auf dem Nachfolger v2 auf.

Der User *mikylucky* hat aber einen Fork der Bibliothek erzeugt, die mit Version 3 spricht (siehe Links). Neben der größeren Zukunftssicherheit hat dieser Fork den Vorteil, bestimmte Fähigkeiten der Version 3 zu unterstützen. So ist damit über einen Aufruf wie

```
$client->bulk->send( $products_array )
```

ein Massenmodus möglich, der mit v3 eingeführt wurde und die Bearbeitung von bis zu 100 Produkten mit einem einzigen Aufruf bietet. Zum Umstellen auf dieses neue Client-API müssen Sie Ihren eigenen Code nicht umstellen: In der Benutzung ist der Fork identisch zum Original, verfügt aber noch über zusätzliche Methoden. Unklar ist allerdings noch, inwieweit der Fork weitergepflegt wird. Bei seiner oft eingesetzten Vorlage kann man davon ausgehen, dass ein Umstieg auf v3 erfolgt, sobald bei Woocommerce ein Wegfall des v2-API in Aussicht steht. Wenn Sie also ständig eine große Menge an Produkten anlegen oder ändern, kann der Umstieg für Sie sinnvoll sein. Sie sollten aber gut testen, ob der Fork seine Arbeit korrekt erledigt, weil er einfach von weniger Programmierern verwendet wird als sein Original und daher eher noch Bugs enthalten könnte.



Differences between v1 and v2

- v1 supports XML response format, v2 only supports JSON.
- v1 does not support creating or updating (with the exception of order status) any resources, v2 supports full create/read/update/delete for all endpoints.
- v1 does not include order item meta, v2 includes full order item meta (with an optional `filter` parameter to include protected order item meta)
- v1 does not include any endpoints for listing a customer's available downloads, v2 includes the `GET /customer/{id}/downloads` endpoint.
- v1 includes an endpoint for listing notes for an order, v2 includes full create/read/update/delete endpoints.
- v1 does not include any endpoints for listing product categories, v2 includes two endpoints for product categories (`GET /products/categories` and `GET /products/categories/{id}`).
- v1 does not include any endpoints for getting valid order statuses, v2 includes an endpoint for listing valid order statuses (`GET /orders/statuses`).
- v2 supports the core features added in WooCommerce 2.2, primarily order refunds (via the `/orders/refunds` endpoint) and Webhooks (via the `/webhooks`).

Differences between v3 and older versions

- v3 implements full basic authentication (conforms to the Basic auth spec)).
- v3 fixes the OAuth implementation to be compliant with the Oauth 1.0a specs.
- v3 includes a new endpoint to get all product orders.
- v3 has new endpoints to allow bulk actions as edition and creation of products, orders, customers and coupons.
- v3 introduces new product attribute endpoints (`GET`, `POST`, `PUT` and `DELETE`).
- v3 deprecated the product/sku/<id> endpoint (because a SKU can be generated with any character, besides that there is a filter callend `filter[sku]`).
- v3 includes category thumbnails on the requests for `product/categories`.
- v3 uses our option to auto generate passwords for new customers.

Aktualisiert: Die Dokumentation erklärt die Unterschiede der verschiedenen Versionen des Woocommerce-API

```
object(stdClass)[16]
  public 'products' =>
    array (size=10)
      0 =>
        object(stdClass)[17]
          public 'title' => string 'Testprodukt' (length=11)
          public 'id' => int 7230
          public 'created_at' => string '2016-02-03T19:32:55Z' (length=20)
          public 'updated_at' => string '2016-02-03T19:32:55Z' (length=20)
          public 'type' => string 'variable' (length=8)
          public 'status' => string 'publish' (length=7)
          public 'downloadable' => boolean false
          public 'virtual' => boolean false
              ...
      1 =>
        object(stdClass)[20]
          public 'title' => string 'Opera, Effektgarn mit viel Kontrast' (length=35)
          public 'id' => int 7204
          public 'created_at' => string '2016-01-27T15:07:36Z' (length=20)
          public 'updated_at' => string '2016-01-27T15:07:36Z' (length=20)
          public 'type' => string 'variable' (length=8)
          public 'status' => string 'publish' (length=7)
          public 'downloadable' => boolean false
          public 'virtual' => boolean false
              ...
      2 =>
        object(stdClass)[77]
          public 'title' => string 'Cotone uni, mercerisiertes Basic-Garn' (length=37)
          public 'id' => int 7162
          public 'created_at' => string '2016-01-26T16:29:50Z' (length=20)
          public 'updated_at' => string '2016-01-26T16:29:50Z' (length=20)
          public 'type' => string 'variable' (length=8)
          public 'status' => string 'publish' (length=7)
          public 'downloadable' => boolean false
          public 'virtual' => boolean false
```

Durch die Nutzung eines Filters liefert Ihnen das API mehrere Objekte vom Typ *products* zurück (Bild 4)

das *foreach* über alle Produkte der aktuellen Seite. Beispielhaft gibt das Skript hier den Produkttitel aus. Möchten Sie die Auswahl filtern, dann tragen Sie Ihre Bedingung in die Variable *$filter* ein, ändern also die erste Zeile etwa so:

```
$filter = ['category'=>'Zubehör'];
```

Gibt es bei der Abfrage des Woocommerce-API ein Problem, wirft die Bibliothek eine Exception. Über die Methode *getCode()* der Exception können Sie die Art des Problems erfahren. Allerdings ist dieser Fehlercode lediglich der HTTP-Antwortcode des REST-Aufrufs und damit relativ unspezifisch.

So erhalten Sie beispielsweise einen 400er-Fehler, wenn das Schreiben von Produkt-Daten fehlschlägt, weil die SKU-Kennung bereits existiert. Derselbe Fehlercode kommt aber auch zurück, wenn Sie einen fehlerhaften REST-Aufruf ausführen. Einen 404 bekommen Sie, wenn ein abgerufenes Objekt nicht existiert oder ein benötigter Parameter fehlt.

Aber die genaue Identifizierung eines Fehlers ist dennoch möglich. Denn über die Methode *getMessage()* der Exception erhalten Sie eine verständliche Fehlermeldung, die zusätzlich eine eindeutige Fehlerkennung enthält, wie in folgendem Beispiel:

```
Error: Diese Art.-Nr. ist bereits einem anderen
Produkt zugeteilt [woocommerce_api_product_sku_
already_exists]
```

Wenn Sie also in der Lage sei möchten, auf das exakte Problem zu reagieren, dann können Sie die in eckigen Klammern gehaltenen Fehlerkennung verwenden und zum Beispiel für das Anlegen eines Produkts schreiben:

```
try {
  $erg = $client->products->create([...]);
} catch (WC_API_Client_Exception $e) {
  if(strpos($e->getMessage(),
'woocommerce_api_product_sku_already_exists')){
// Behandlung SKU-Doublette;
...
} else {
  // Allgemeinen WC-Fehler behandeln
}
```

Hat der Shop auch Produkte mit Varianten, wie verschiedene Größen oder Farben, dann sind diese zwar als eigene Produkte mit separaten IDs gespeichert, werden aber vom Datenmodell her dem Hauptprodukt untergeordnet. Wenn Sie dieses abfragen, finden Sie alle seine Varianten als Array in der Property *variations* wieder.

So gehen Sie mit variablen Produkten um

Wollen Sie ein vorhandenes Produkt um eine Variante erweitern, weil es zum Beispiel in einer neuen Farbe lieferbar ist, dann muss die neue Eigenschaft vorhanden sein. Sie müssen also Woocommerce sowohl mitteilen, dass dieses Produkt nun auch über die Eigenschaft *Farbe=rot* verfügt, als auch diese Farbe bei den Attributen der neuen Produktvariante setzen.

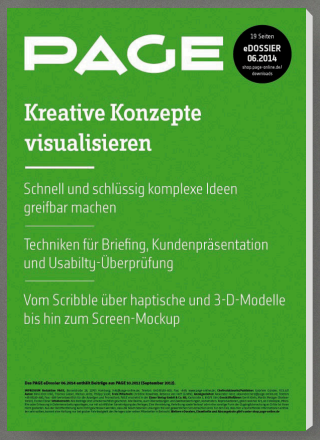
Dazu genügt es, wenn Sie die neu hinzugekommene Eigenschaft und die Produkt-Varianten in einem Rutsch mittels einer Operation vom Typ *products->update()* anlegen. Weil das zugehörige Beispiel eine recht umfangreiche Datenstruktur hat, die den Rahmen dieses Beitrags sprengen würde, nutzen Sie zur Anschauung am besten das Beispielskript von unserer Website. ■

Links zum Thema

- Dokumentation zum REST-API von Woocommerce
  http://woothemes.github.io/woocommerce-rest-api-docs
- Client-Bibliothek für das Woocommerce-API
  http://github.com/kloon/WooCommerce-REST-API-Client-Library
- Fork der Client-Lib, die Bulk-Operations unterstützt
  http://github.com/mikylucky/WooCommerce-REST-v3-API-Client-Library
- Blog der Woocommerce-Entwickler mit News zum API
  http://woocommerce.wordpress.com

Markus Schraudolph
ist Journalist und Programmierer. Er schreibt seit 16 Jahren Bücher und Artikel für Fachzeitschriften. Seine Schwerpunktgebiete als Programmierer sind die Webprogrammierung und Datenbanken.

Downloaden, aufschlauen!



PAGE eDossiers – Best-of-Kompilationen aus PAGE und WEAVE im Originallayout: PDFs einfach und jederzeit runterladen in unserem Online-Shop **shop.page-online.de/downloads**

PAGE
Das Magazin der Kreativbranche

Ein Magazin der Verlagsgruppe Ebner Ulm • www.ebnerverlag.de

GRAFIK FÜR ENTWICKLER

Screendesign im Wandel

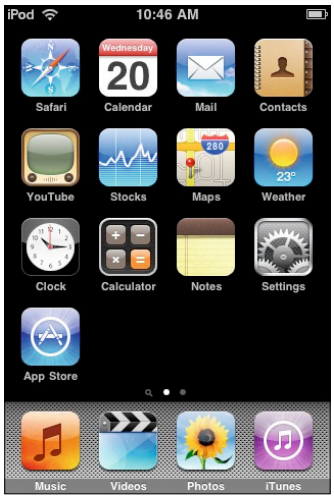
Digitale Oberflächen orientieren sich nicht nur an der Mode.

Die Gestaltung einer digitalen Oberfläche sollte optisch ansprechend sein und dabei funktionale Anforderungen und Gegebenheiten berücksichtigen. Je besser dieses Zusammenspiel funktioniert, desto höher ist die Benutzerfreundlichkeit der Anwendung.

In diesem Zusammenhang wird oft auch von Software-Ergonomie gesprochen: Ein Programm, eine Webseite oder auch eine App sollte vom Benutzer intuitiv zu bedienen sein – eben benutzerfreundlich. Daraus ergeben sich, je nach Art der Anwendung, verschiedene Standards für die technische Umsetzung. Hinzu kommen Gestaltungslinien, die sich an einer vorhandenen CI (Corporate Identity) orientieren oder neue Regeln zum Schaffen einer solchen aufstellen.

Wie dann aber die grafische Umsetzung erfolgt, hängt von weiteren Faktoren ab. Unter anderem zählt der Umfang der geplanten Anwendung, also die Anzahl und Größe der Elemente, die auf der Bildfläche untergebracht werden müssen. Handelt es sich um ein Spiel, sind grafische Elemente wichtiger als etwa bei einem Webauftritt einer Tageszeitung.

Hinzu kommt die angepeilte Zielgruppe, die vorgibt, ob es kürzere oder längere Texte gibt und wie stark Bilder und Grafiken vertreten sind. Zudem unterliegt die Gestaltung einer Bedienoberfläche den technischen Gegebenheiten und muss dabei die Art des Ausgabemediums berücksichtigen. Aber auch unterschiedliche Stilrichtungen aus anderen Bereichen, etwa aus der bildenden Kunst, beeinflussen die grafische Gestaltung der verschiedenen digitalen Anwendungen.



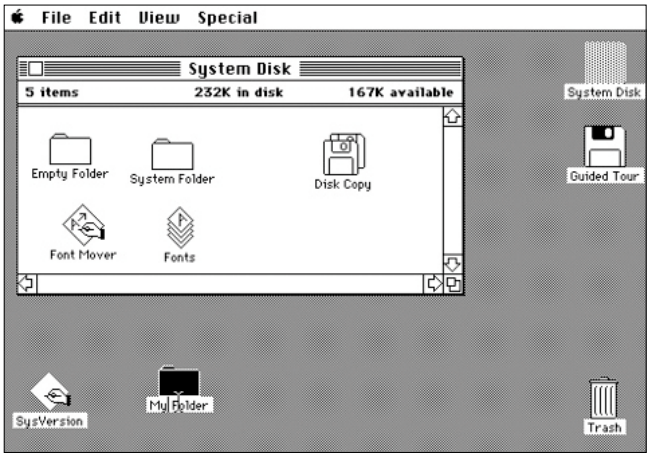
Quelle: Flickr/Chris Enns

Skeuomorphismus: Die früheren Icons der iPhone-Apps hat Apple in einem naturgetreu nachgebildeten Design angelegt (Bild 2)

Rückblick

Im Jahr 1973 zeigte der Xerox Alto als erster Computer eine grafische Bedienoberfläche, deren Fenster bereits mit einer Maus gesteuert werden konnten. Dem Modell Alto II spendierten die Entwickler einen Monitor mit einer Auflösung von immerhin 606 x 808 Pixeln, die allerdings nur Schwarz oder Weiß darstellen konnten.

Davon inspiriert, entwickelte Steve Jobs den Apple Lisa, dem ein Jahr später, 1984, der erste Apple Macintosh folgte – mit einem 9-Zoll-Monitor, der jedoch ebenfalls nur Schwarz-Weiß darstellen konnte (Bild 1). Erst der Monitor des Mac II (1987) erstrahlte in Farben.



Die Oberfläche des ersten Macintosh ließ nur wenig Gestaltungsspielraum (Bild 1)

Eine Traumwelt für Online-Spieler, die liebevoll gestaltete Details zeigt (Bild 3)

Zur gleichen Zeit entwickelte die Atari Corporation den Atari ST, dessen erstes Modell 1985 auf den Markt kam. Parallel dazu erfreute sich der Commodore Amiga, dessen Monitor wie auch das Display des Atari Farben darstellen konnte, immer mehr Beliebtheit. 1986 wartete Microsoft ebenfalls mit einer ersten Version von Windows auf.

Mit Mac OS X stellte Apple das neue Oberflächen-Design Aqua vor (Bild 4)

Skeuomorphismus

Dank größerer Monitore mit höherer Auflösung sowie einer optimierten Farbdarstellung von 8 Bit pro Kanal (RGB) konnten Screendesigner nahezu unbegrenzt neue Ideen und Konzepte zur Gestaltung digitaler Oberflächen verwirklichen.

Skeuomorphismus (Altgriechisch: Behälter, Werkzeug, Gestalt) war früher eher im Kunstgewerbe bekannt, bezeichnet jedoch heute einen grafischen Stil von Bedienoberflächen und -elementen: Die Objekte werden möglichst naturgetreu nachgebildet. Dabei versucht diese Stilrichtung, ältere Erscheinungsformen realer Gegenstände nachzubilden, um die Objekte vertraut erscheinen zu lassen. Zur Umsetzung dieses Stils muss sich der Grafiker intensiv mit 3D-Effekten auseinandersetzen. Neben der realistischen Darstellung von verschiedenen Texturen sind gute Kenntnisse über Lichtwirkung und Schattenwurf Voraussetzung.

Bis vor Kurzem noch gab es mannigfaltige Beispiele zu dieser grafischen Orientierung, etwa bei den verschiedenen iOS-Apps: So bettete ein ledernes Ringbuch die Kontakte, Notizzettel zeigten deutliche Abrissspuren, die Einstellungen konnte der User an den ineinandergreifenden Zahnrädern auf dem Icon erkennen, und YouTube-Filme starteten über einen uralten Fernseher (Bild 2).

Mittlerweile hat Apple von solchen voluminösen Grafiken Abstand genommen, wie auch so mancher Webbetreiber, der nun seinen Internetauftritt schlicht und modern hält.

Anzutreffen ist Skeuomorphismus jedoch immer noch – und hat in vielen Fällen, in denen er verwendet wird, sogar eine Daseinsberechtigung: So zeigen verschiedene Spiele – online im Web oder auf dem iPhone als App – sehr schöne und detailreich gestaltete Traumwelten (Bild 3).

Der iMac Bondi Blue stand Modell für das neue Aqua-Design (Bild 5)

Aqua-Design

Bis zur Version Mac OS 9 (1999) zeigte das User-Interface des Macintosh die Menüleiste und Rahmen der einzelnen Fenster in Schwarz-Weiß oder Grau. Zwar wurde mit OS 9 der Apfel links oben bunt, sonst änderte sich jedoch nur wenig am Stil der Bedienoberfläche. Erst die Public Beta von Mac OS X (September 2000) löste das bis dahin aktuelle Platinum-Design ab und zeigte als Neuerung eine Bedienoberfläche im Aqua-Design: Auf einer dezent texturierten Leiste wurden die Schaltflächen zum Verkleinern, Vergrößern oder zum Verbergen des Fensters als farbig transparente Knöpfe angelegt, die in ihrer Anmutung an flüssige Substanzen oder Gel erinnern. Auch die Scroll-Leiste zeigte dieses Design (Bild 4).

Dazu passend gab's die entsprechende Hardware: Bereits vor OS X löste sich Apple vom schnöden Hellbeige, in dem nahezu alle Computergehäuse zu dieser Zeit erstrahlten, und schuf den ersten iMac, Bondi Blue, dessen Aussehen die Vorgabe für das neue Interface-Design lieferte: farbig-transparent und mit einer feinen Streifenstruktur (Bild 5).

Der frische, neue Stil begeisterte und regte viele zum Nachahmen an. Besonders auf Webseiten waren in den Jahren ab 2001 häufig Schaltflächen im Aqua-Design zu betrachten, zudem gab es in jeder Ecke des virtuellen Netzes entsprechende Tutorials – ob schriftlich oder als YouTube-Video. Erst ab 2007 ebbte die Begeisterung für Aqua etwas ab (Bild 6).

Bartelme.at: Ein Tutorial zum Thema Aqua-Design von vielen, die es bis vor wenigen Jahren im Netz gab (Bild 6)

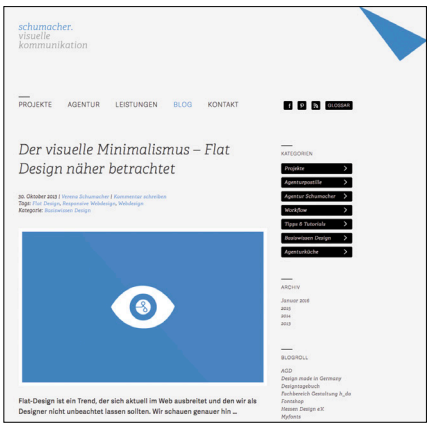
Mit Jaguar (Oktober 2003), der nachfolgenden Version des Betriebssystems von Apple, wurde der Aqua-Effekt bereits etwas abgemildert und Details entfärbt. Weitere kleinere Änderungen an der Aqua-Oberfläche folgten, die Gel-artige Erscheinung der Fenster-Schaltknöpfe blieb jedoch. Erst mit OS X Yosemite (August 2015) rangierte der Hersteller Aqua zugunsten eines neuen grafischen Interfaces aus und reagierte somit etwas spät auf den aktuellen Geschmack: Schließlich präsentierten bereits die meisten Hersteller ihr User-Interface in einem wesentlich schlichterem Design – dem Flat-Design.

Flat-Design

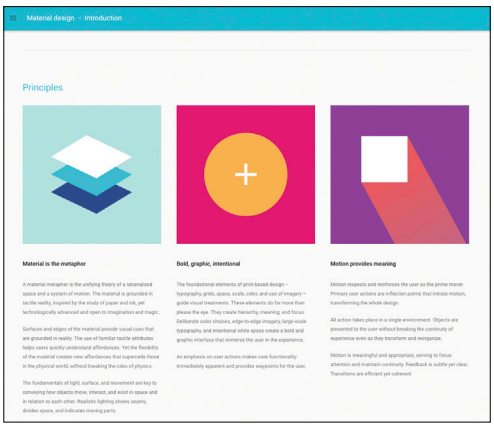
Kein Wunder, dass nach den eher verspielten und dabei aufwendigen Designrichtungen Skeuomorphismus und Aqua ein eher schlichtes Layout gefragt war. Dabei erwies sich Miccosoft schneller als Apple: Den Anfang machte im Jahr 2010 das Windows Phone 7; zwei Jahre später etablierte sich mit der Version 8 das Flat-Design end- ▶

gültig (Bild 7). Erst im Jahr 2013 mit iOS 7 unterzog Apple sein mobiles Betriebssystem einer radikalen Design-Änderung. Icons, Schalter und die Oberflächen einzelner Apps zeigten sich wesentlich schlichter, und verspielte Details verschwanden. Einzelne Elemente richten sich sichtbar an einem Raster aus.

Klarer Ansatz der Stilrichtung ist eine Reduzierung auf das Wesentliche: So werden dreidimensionale Elemente kurzerhand entfernt, auch aufwendige Strukturen, Farbverläufe und Verzierungen jeglicher Art müssen den meist streng geometrischen Formen und Flächen weichen.



Auf der Webseite der Designagentur Schumacher-visuell.de blogt die Inhaberin über das Flat-Design – dabei ist auch die Seite selbst im Flat-Design gestaltet (Bild 8)



Das von Google entwickelte Material Design unterliegt in seiner Umsetzung exakten Regeln, die Google auf seiner Webseite zugänglich gemacht hat (Bild 9)

Alle Objekte richten sich in der Regel kachelartig an einem Raster aus. Mit entsorgt werden alle Schlagschatten; ob sie nun rein optisches Beiwerk sind oder unter einer Schaltfläche als Indikator für den Aktions-Zustand dienen. Neben Bedienoberflächen profitieren auch Webseiten von diesem neuen Stil: Wenn die ausschmückende Elemente wegfallen, wird der Blick des Betrachters automatisch auf die wichtigen Inhalte gelenkt (Bild 8).

Quelle: Flickr, Mahesh Mohan

Windows Phone 8.1: Einfache Formen ohne 3D-Effekte und Schatten zeichnen das übersichtliche Flat-Design aus (Bild 7)

Material Design von Google

Flat-Design erlaubt lediglich das absolute Minimum an Gestaltungselementen. Darauf basierend entwickelte Google Mitte 2014 das Material Design: Auch hier richten sich die verschiedenen Flächen an einem rechteckig angelegten Raster aus und zeigen keine Texturen und auch keinerlei Dreidimensionalität.

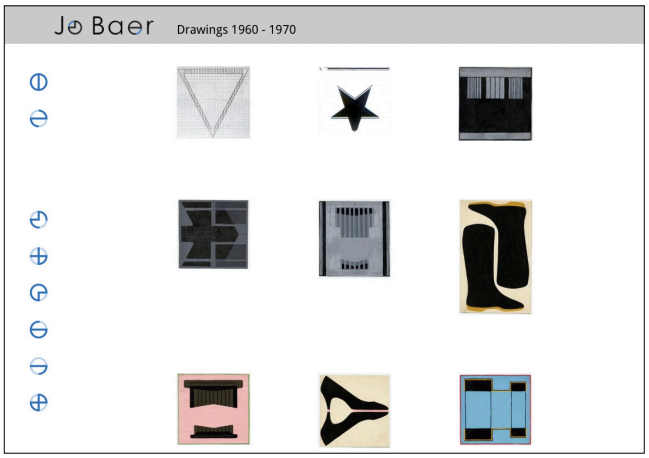
Anders als beim Flat-Design ist nun jedoch eine dezente Schattenwirkung und deren Animation erlaubt, ja regelrecht erwünscht, wenn sie funktional ist: So entsteht beispielsweise durch einen Schatten eine Tiefenwirkung, die dem Anwender zeigt, dass an dieser Stelle etwas zu drücken ist.

Material Design ist hauptsächlich bei Android-Apps zu finden. Für Entwickler hat Google unter der Adresse https://www.google.com/design/spec/material-design/introduction.html#introduction-goals die verschiedenen Design-Prinzipien zusammengefasst. Vermittelt werden zudem genaue Angaben zu den Eigenschaften und der Darstellung von Material, einer Räumlichkeit durch die dritte Dimension, sowie Licht und Schatten (Bild 9).

Minimaldesign und Minimalismus

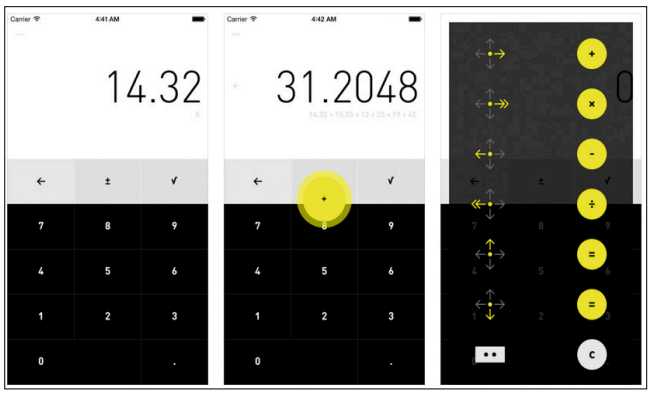
Minimaldesign kann als gestalterische Grundlage von Flat- oder Material Design verstanden werden und zeigt somit sehr ähnliche Gesetzmäßigkeiten. Der Grundgedanke des Minimalismus stammt aus dem asiatischen Raum. Geprägt davon behauptete sich in Amerika in den frühen 1960er Jahren die Minimal Art als neue Stilrichtung. Bilder, Objekte und Skulpturen zeigten sich reduziert auf einfache Strukturen, oft geometrischer Art. Form und Farbe wurden damals in den Werken auf das Wesentliche reduziert (Bild 10).

Auch im Internet zeigen sich mittlerweile verschiedene Auftritte im Minimaldesign von ihrer besten Seite. Wie in der Kunst kommt es hier auf Funktion und Inhalt an: kein überflüssiger Schmuck, egal ob Form oder Farbe, keine blinken-



Quelle: http://www.jobaer.net

Minimal Art: Sowohl die Webseite als auch die Kunstwerke der amerikanischen Malerin und Grafikerin Jo Baer folgen dem klaren minimalistischen Stil (Bild 10)

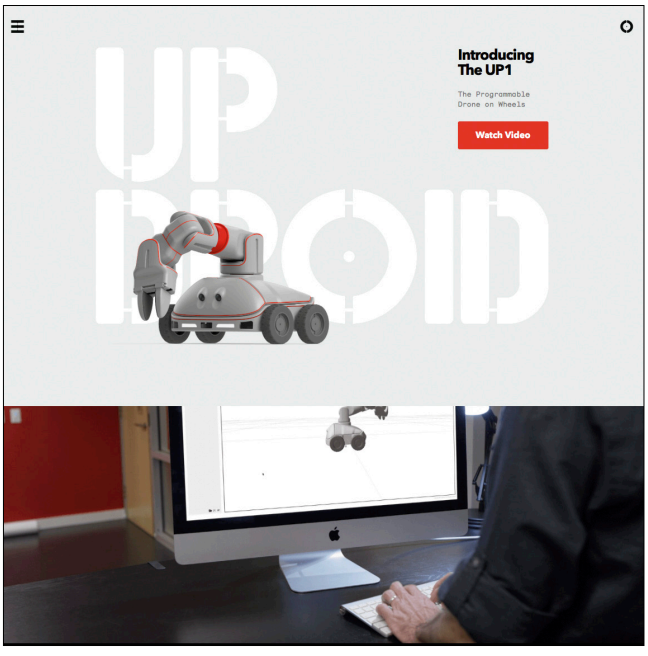


Bei der Rechner-App wirkt das minimale Design funktional und ansprechend (Bild 11)

den Animationen oder Schaltflächen und keine mit unnützen Inhalten vollgestopften Flächen. Dabei gilt es, die wenigen Details geschickt in Szene zu rücken, um so eine funktionale Ästhetik aufzubauen.

Bei vielen Beispielen dominieren Weiß, Schwarz und unterschiedliche Grauabstufungen, oft gepaart mit einer leuchtenden Farbe als Akzent: In Bild 11 zeigt beispielsweise die durch Gesten gesteuerte Rechner-App von Aeliox neben den unbunten Farben ein leuchtendes Gelb. Serifenlose Schriften unterstützen die klare Linie.

Bei der Flächenaufteilung kommt – wie auch beim Flat-Design – häufig ein Gestaltungsraster zum Einsatz, an dem sich die verschiedenen Texte, Bilder und Grafiken ausrichten. Die freien Flächen werden ebenfalls durch ein solches Raster definiert und begrenzt. Webseiten in diesem Stil werden häufig als Single Page angelegt (Bild 12). ▶



Minimaldesign: Die Webseite Updrioid. com zeigt einen klaren Aufbau, viel Leerraum und setzt den Fokus auf die typografische Gestaltung (Bild 12)

One-Page- oder Single-Page-Design

Das im vorangegangenen Absatz bereits erwähnte Single-Page-Design hat nichts mit einer Stilrichtung zu tun. Hier geht es vielmehr um einen besonderen Aufbau von Webseiten: Alle Informationen befinden sich auf einer einzigen Seite, und der Betrachter kann über Sprungmarken zu den entsprechenden Stellen navigieren. Dieses Konzept – auch Pageless Design genannt – löst sich deutlich von der althergebrachten Struktur aus dem Print-Bereich, bei der die Inhalte thematisch auf einzelnen Seiten untergebracht werden (Bild 13, Bild 14).

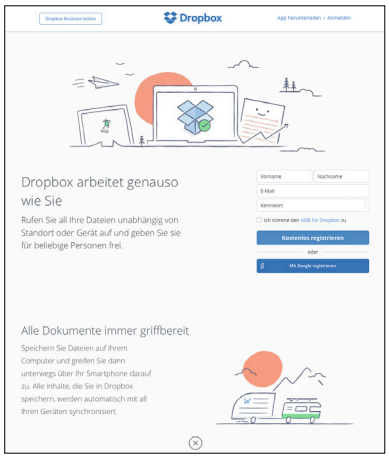
Single-Page: Weshootbottles.com wählt einen eher außergewöhnlichen Weg und ordnet die Seiteninhalte auf einer langen Seite nebeneinander an (Bild 14)

Nicht jeder Internetauftritt verträgt jedoch einen solchen Aufbau. Gibt es viele komplexe Inhalte nebeneinander, sollte auf ein Single-Page-Design verzichtet werden, da hier meist eine verschachtelte Navigation vonnöten ist. Vielmehr eignet sich diese Form für Präsentationen, die auch eine eher plakative Darstellung vertragen. So lassen sich beispielsweise kleinere Firmenauftritte oder Produkte oft bestens auf diese Art darstellen.

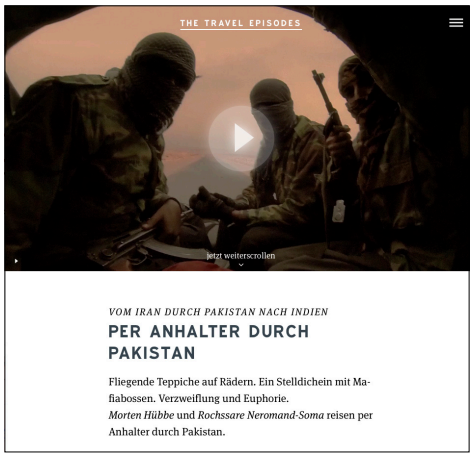
Damit diese Layout-Form auch benutzerfreundlich ist, sollten die Inhalte in eine schlüssige Reihenfolge gebracht werden und sich deutlich optisch voneinander absetzen. Zu viel Text macht die Site unübersichtlich, hier punkten aussagekräftige Bilder, die im Idealfall eine „Geschichte" erzählen.

Scrollytelling

Scrollytelling kann fast als Sonderform der One-Pager verstanden werden. Hier ist Scrollen von oben nach unten angesagt, manchmal auch von links nach rechts, um durch eine – in der Regel stark bebilderte – Geschichte zu navigieren. Meist handelt es sich dabei um eine Reportage, wie etwa bei Snow Fall, der ersten Veröffentlichung in diesem Stil, einem Bericht über ein Lawinenunglück, der auf der New York Times veröffentlicht wurde (www.nytimes.com/projects/2012/snow-fall/#/?part=tunnel-creek). Oft finden sich weitere multimediale Elemente in einer solchen digitalen Welt, etwa Videos und Sound, beispielsweise zu Beginn der Seite Travelepisodes.com, auf der eine Reise »Per Anhalter durch Pakistan« beschrieben wird. Gehalten im Responsive-Design, können solche Webseiten auch auf einem Smartphone oder Tablet-Computer betrachtet werden (Bild 15).



Dropbox liefert alle Informationen gut strukturiert und minimalistisch gehalten auf einer Seite (Bild 13)



Scrollytelling: Das Beispiel zeigt eine multimediale Reportage über eine Reise durch Pakistan auf einem Smartphone (Bild 15)

Fazit

Vom Skeuomorphismus über Aqua-Design bis hin zum Minimalismus – die Entwicklung der grafischen Gestaltung von Oberflächen verdeutlicht, dass sehr häufig gilt: Weniger ist mehr.

Zu viele Elemente lenken den Betrachter vom Wesentlichen ab. So zeigt ein gut strukturiertes und einfach gehaltenes Design, etwa das Flat-Design, klipp und klar die Funktionalität der Anwendung. Zudem treffen nicht alle Stilrichtungen die Geschmäcker der einzelnen User.

Dennoch haben auch aufwendig gestaltete Interfaces durchaus ihre Berechtigung, solange die Grafiken zum jeweiligen Thema passen und zielgruppenorientiert sind, etwa bei Spiele- oder Kinder-Apps. ■

Katharina Sckommodau
arbeitet als freiberufliche Autorin, Grafikerin und Dozentin, unter anderem für die Akademie der Bayerischen Presse und für Macromedia. Sie veröffentlicht regelmäßig Beiträge in renommierten Fachzeitschriften.

Impressum

Verlag
Neue Mediengesellschaft Ulm mbH
Bayerstraße 16a,
80335 München
Telefon: (089) 741 17-0,
Fax: (089) 741 17-101
(ist zugleich Anschrift aller Verantwortlichen)

Herausgeber
Dr. Günter Götz

Chefredakteur
Max Bold
– verantwortlich für den redaktionellen Teil –
E-Mail: redaktion@webundmobile.de

Schlussredaktion
Ernst Altmannshofer

Redaktionelle Mitarbeit
Philip Ackermann, Christian Bleske, Ed Charbeneau, Ekkehard Gentz, Tam Hanna, Johannes Hoppe, Olf Jännsch, Anna Kobylinska, Bernhard Lauer, Patrick Lobacher, Ferdinand Malcher, Filipe Martins, Florence Maurice, Konstantin Pfliegl, Frank Pientka, Michael Rohrlich, Jochen Schmidt, Markus Schraudolph, Katharina Sckommodau, Thomas Sillmann, Frank Simon, Markus Stäuble

Art Directorin
Maria-Luise Sailer

Grafik & Bildredaktion
Alfred Agatz, Dagmar Breitenbauch, Catharina Burmester, Hedi Hefele, Manuela Keller, Simone Köhnke, Cornelia Pflanzer, Petra Reichenspurner, Ilka Rüther, Christian Schumacher, Nicole Üblacker, Mathias Vietmeier

Anzeigenberatung
Jens Schmidtmann, Anzeigenleiter
Klaus Ahlering, Senior Sales Manager
Telefon: (089) 741 17-125
Fax: (089) 741 17-269
E-Mail Anzeigenberatung: sales@nmg.de

Anzeigendisposition
Dr. Jürgen Bossmann
Telefon: (089) 741 17-281
Fax: (089) 741 17-269
E-Mail: sales@nmg.de

Leitung Herstellung/Vertrieb
Thomas Heydn
Telefon: (089) 741 17-111
E-Mail: thomas.heydn@nmg.de

Leserservice
Hotline: (089) 741 17-205
Fax: (089) 741 17-101
E-Mail: leserservice@nmg.de

Kooperationen
Denis Motzko
Telefon: (089) 741 17-116
E-Mail: kooperationen@nmg.de

Druck
L.N. Schaffrath Druckmedien
Marktweg 42–50
47608 Geldern

DVD-Produktion
Stroemung GmbH

Vertrieb
Axel Springer Vertriebsservice GmbH
Objektvertriebsleitung Lothar Kosbü
Süderstraße 77
20097 Hamburg
Telefon: (040) 34724857

Bezugspreise
web & mobile developer ist das Profi-Magazin für Web- und Mobile-Entwickler und erscheint zwölfmal im Jahr. Der Bezugszeitraum für Abonnenten ist jeweils ein Jahr. Der Bezugspreis im Abonnement beträgt 76,20 Euro inklusive Versand und Mehrwertsteuer im Halbjahr, der Preis für ein Einzelheft 14,95 Euro. Der Jahresbezugspreis beträgt damit 152,40 Euro.

In Österreich sowie im übrigen Ausland kostet das Abonnement 83,70 Euro im Halbjahr. Der Jahresbezugspreis beträgt somit 167,40 Euro. In der Schweiz kostet das Abonnement 152,00 Franken im Halbjahr. Der Jahresbezugspreis in der Schweiz beträgt 304,00 Franken.

Das Abonnement verlängert sich automatisch um ein Jahr, wenn es nicht sechs Wochen vor Ablauf der Bezugszeit schriftlich beim Verlag gekündigt wird.

Studenten erhalten bei Vorlage eines Nachweises einen Rabatt von 50 Prozent.

ISSN: 2194-4105

CARDBOARD VON GOOGLE

Glass auf Sparflamme

Das Google Cardboard macht aus jedem Smartphone eine Virtual-Reality-Brille.

Googles erstem Datensichtgerät war ob des hohen Preises nur wenig kommerzieller Erfolg beschieden. Mit der aus Karton bestehenden Cardboard wendet das IT-Unternehmen nun das alte angelsächsische Prinzip des Think Small auf den VR-Markt an.

Die mittlerweile von einer Vielzahl verschiedener Anbieter verfügbare Konstruktion ist verblüffend einfach: Es handelt sich um eine faltbare Box aus Karton, die zwei Linsen enthält. Ein eingelegtes Smartphone wird mit einer speziellen Software ausgestattet, die zwei nebeneinanderliegende Kamerabilder rendert. Dank der in der Brille verbauten Linsen sieht jedes Auge eines der Bilder, wodurch eine grundlegende stereoskopische Darstellung entsteht.

Bevor wir uns mit der Nutzung der Brille beschäftigen, noch ein Hinweis aus der schmerzlichen Praxis. Cardboard wird gerne – insbesondere in der von Google hergestellten, braunen Version – von Mitarbeitern und Reinigungspersonal als Abfall betrachtet und entsorgt. Als Alternative dazu bietet sich die Nutzung von weißen oder individuell bedruckten Varianten an. Ein Beispiel dafür ist die in Bild 1 gezeigte Brille aus dem Hause Just-vr, die mit Logos und Designs bedruckt werden kann. Das Unternehmen hat den Designvorschlag von Google zudem um höherwertige Linsen, Halteband und anpassbaren Augenabstand ergänzt.

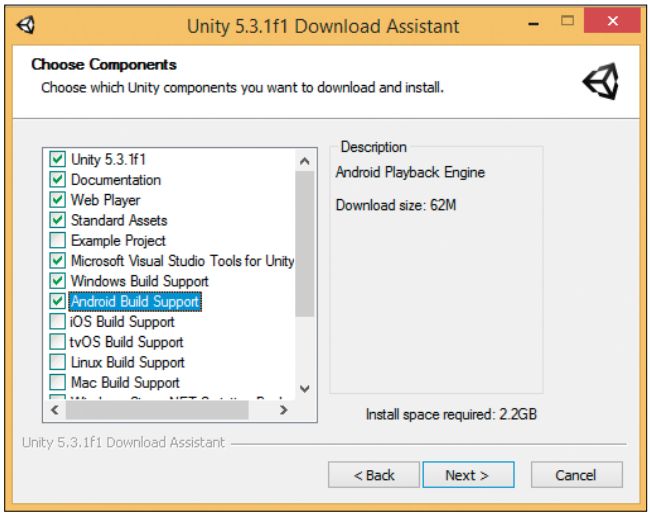
Dank Firmenlogo wandert diese Brille nur selten in den Müll (Bild 1)

Einfach mit Unity

Die Entwicklung des Spielemarktes hat dafür gesorgt, dass kleine und mittlere Unternehmen nur schwer eigene Engines erzeugen können: Ein Gutteil der im Store erhältlichen Spiele dürfte mittlerweile auf Unity basieren. Es ist schon aus diesem Blickwinkel nicht verwunderlich, dass Google diese Engine in seinem Cardboard-SDK unterstützt.

Wir wollen in diesem Artikel auf Unity 5 setzen. Google liefert zwar einige Unterstützungsskripts für Unity 4 aus, deren Verwendung aber nur für Personen mit signifikantem auf Unity 4 basierendem geistigen Eigentum interessant ist.

Öffnen Sie den URL https://unity3d.com/get-unity und laden Sie den Installer für die Personal Edition herunter. Achten Sie darauf, im Download-Assistenten die in Bild 2 hervorgehobene Option *Android Build Support* zu aktivieren. Standardmäßig ist diese Option deaktiviert.



Android wird nicht von Haus aus installiert (Bild 2)

Cardboard SDK for Unity

Laden Sie im nächsten Schritt das unter der Adresse https://developers.google.com/cardboard/unity/download bereitstehende Cardboard SDK for Unity herunter. Das Anklicken des in Bild 3 hervorgehobenen Links auf der Downloadseite liefert ein installationsbereites Paket.

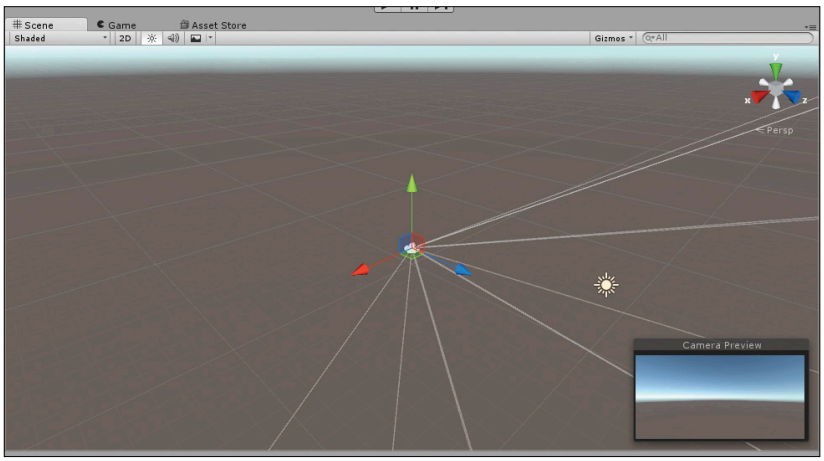
Erstellen Sie im nächsten Schritt ein neues 3D-Projekt. Wer Unity frisch installiert hat, muss sich zunächst mit seinem Unity-Konto anmelden und eine Lizenz aktivieren (Personal genügt). Laden Sie nach der Erstellung der im Beispielcode als *NMGGlasses1* bezeichneten Projektstruktur das Cardboard-SDK durch Anklicken von *Assets, Import Package* und *Custom Package*.

Wählen Sie die Datei *CardboardSDKForUnity.unitypackage* aus, um den Importassistenten zu starten. Die Unity-IDE erlaubt das teilweise Laden von Assetpaketen. Für unser Projekt sind alle Elemente erforderlich. Klicken Sie daher auf *All*, bevor Sie den Importprozess durch den *Import*-Button abschließen.

Kamera auf Speed

Das Laden des Asset-Pakets beeinflusst die von Unity erzeugte Szene nicht wesentlich. Es handelt sich nach wie vor um ein Projekt für Workstations. Dieses Problem lässt sich unter *File* und *Build Settings* beheben, wo sie die Plattform *Android* markieren und mittels Klick auf *Switch Platform* als aktuelles Entwicklungs- und Deploymentziel festlegen.

Aufgrund eines bekannten technischen Fehlers in Unity ist an dieser Stelle ein Neustart der IDE erforderlich (http://answers.unity3d.com/questions/1030897/cannot-find-component-cardboard-update-stereo-came.html). Markieren Sie die Main Camera in der Hierarchy-Ansicht und klicken Sie auf *Component, Cardboard* und *Update Stereo Cameras.* Die IDE ergänzt die Kamera daraufhin um ein *StereoController*-Skript, das sich wie in Bild 4 gezeigt auf die Darstellung der Szene ebenso wie auf die Inhalte des Inspectors auswirkt.

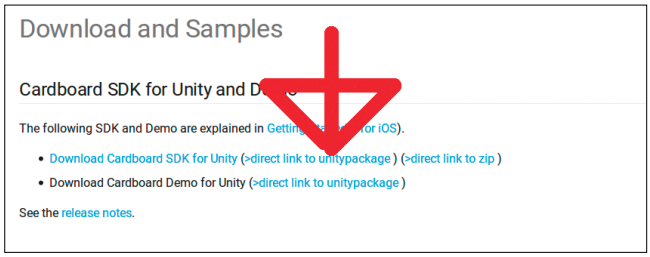


Die vier Linien zeigen an, dass es sich hier um eine Stereo Camera handelt (Bild 4)

Der Button *Update Stereo Cameras* muss nach jedem Ändern der Kameraparameter angeklickt werden. Erfolgt dies nicht, so übernimmt das StereoController-Skript die Einstellungsänderungen nicht. Klicken Sie im nächsten Schritt auf *Play,* um die Szene zu rendern (Bild 5).

Jetzt kommt Bewegung hinein

Als nächste Aufgabe wollen wir ein Objekt realisieren, das um den Kopf des Benutzers herumschwebt. Da auf Kugeln basierende Beispiele bald langweilig werden, importieren wir stattdessen das unter https://www.assetstore.unity3d.com/en/#!/content/16508 bereitstehende Paket mit diversen geometrischen Primitiva.



Dieser Link ist der richtige (Bild 3)

Klicken Sie im Asset-Store auf den Download-Button, um das Paket auf die Arbeitsstation herunterzuladen. Importieren Sie danach alle enthaltenen Assets und öffnen Sie im Asset Explorer den neu erstellten Unterordner *ProceduralToolkit* und *Primitives.*

Durch *GameObject* und *Create Empty* lässt sich ein neues leeres *GameObject* erstellen: Es handelt sich dabei um einen Container, der mit diversen Verhalten kombinierbar ist. Ziehen Sie per Drag and Drop sowohl das C#-Skript *Icosahederon* als auch den *Primitives*-Shader auf das *GameObject,* um es mit Verhalten und Rendering-Anweisungen auszustatten.

Nutzen Sie den Editor zur Anpassung des Aufenthaltsorts des Vielflächers. Da das von Daniil Basmanov gepflegte Werkzeug keine Unterstützung für den Editor mitbringt, erscheint das Icosahederon nur dann am Bildschirm, wenn es im Hierarchy-Fenster markiert wird. Bild 6 zeigt den Grundaufbau, mit dem wir erste Schritte in die Welt der virtuellen Realität wagen wollen.

Wenn Sie Unity mit einer Zweitastenmaus bedienen, zoomen Sie mittels Drücken von [Alt]. Halten Sie danach die rechte Maustaste gedrückt und schieben Sie den Cursor nach vorne oder nach hinten, um den angezeigten Bildausschnitt anzupassen.

Vielflächer auf Wanderschaft

An dieser Stelle ist es Zeit für etwas Mathematik: Das Element soll sich mit beständiger Geschwindigkeit um die Kamera herumbewegen. Dazu müssen wir im ersten Schritt ein C#-Skript erstellen und es in Visual Studio oder MonoDevelop öffnen.

Unser Skript deklariert eine Gruppe von Membervariablen, die das Verhalten des Objekts beeinflussen. *MyCamPos* wird im Rahmen der Initialisierung des Programms mit dem Aufenthaltsort der Hauptkamera versehen, um auf diese Weise eventuelle Änderungen des Aufenthaltsorts des Objekts auszuschließen:

```
public class MovementScript : MonoBehaviour {
  Vector3 myCamPos;
  float myRadius = 8;
  float myAngleInRad = Mathf.PI/2;
  float myAngleStep=0.01f;
  void Start () {
    Camera aCamera = Camera.main;
    myCamPos = aCamera.transform.position;
  }
...
}
```

Angemerkt sei, dass die Kamera ihre Position bei Kopfbewegungen normalerweise nicht ändert: Die Anpassung be- ▶

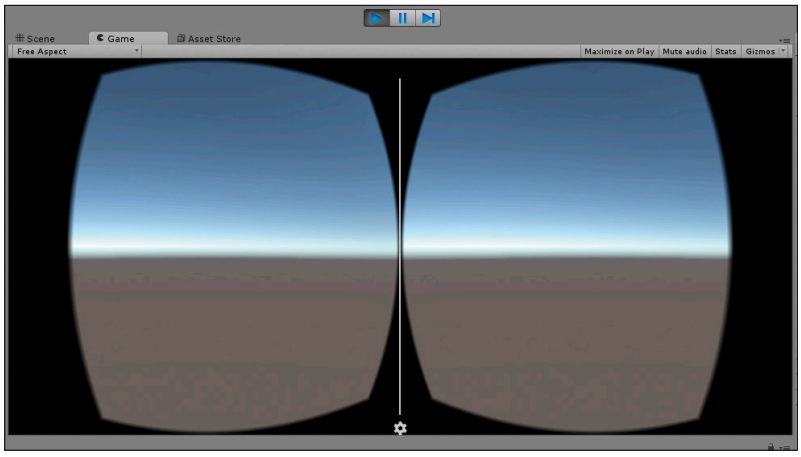
schränkt sich auf Rotationen. Es wäre theoretisch auch möglich, die Position zur Laufzeit zu ermitteln.

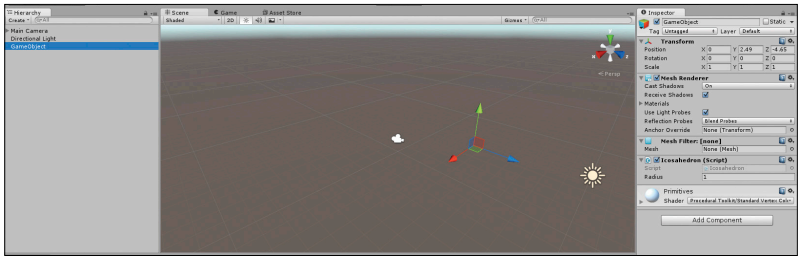
Damit können wir uns der Realisierung der eigentlichen Bewegung zuwenden: Unity ruft die *Update()*-Funktion vor jedem Frame auf, um der Spiel-Engine so eine Möglichkeit zur Anpassung des internen Zustands der diversen Objekte zu gewähren:

```
public class MovementScript : MonoBehaviour
{
...
  void Update () {
  myAngleInRad += myAngleStep;
  Vector3 helperVect = myCamPos;
  helperVect.x += Mathf.Cos(myAngleInRad) * myRadius;
  helperVect.z += Mathf.Sin(myAngleInRad) * myRadius;
  this.transform.position = helperVect;
  }
}
```

Update() beginnt mit dem Addieren eines konstanten Werts zu *myAngleInRad*. Diese Vorgehensweise nutzt die zyklische Art der Sinus- und Cosinusfunktionen aus. Das bei 2pi erkennbare Ergebnis von *sin()* oder *cos()* entspricht dem Wert von 4pi, 6pi oder 8pi.

Die Ermittlung der aktuellen Koordinaten beginnt mit der Beschaffung des ursprünglichen Aufenthaltsorts der Kamera. Er wird im nächsten Schritt um die Ergebnisse der Multiplikation des Skalierungswerts und des Vollausschlagsradius ergänzt und wandert daraufhin eins zu eins in die *position*-Eigenschaft des Icosahederons.





Dank StereoController erscheint die Szene wie auf einem Smartphone (Bild 5)
Stellen Sie sich anstelle des Quaders ein Icosahederon vor (Bild 6)

Kenner trigonometrischer Funktionen wundern sich über die verwendeten Achsen. Im Fall von Google Cardboard zeigt die y-Achse nach oben. Das Festlegen eines Startwerts von Pi/2 sorgt dafür, dass die Kugel anfangs in der Mitte des Sichtfelds ist und dieses nach und nach verlässt.

Klicken Sie danach abermals auf *Play*, um eine Vorschau der Szene zu rendern. Googles Cardboard-Plug-in ergänzt den Unity-Player um einige Tastaturkombinationen, die die Interaktion zwischen Kopf und Szene modellieren.

Drücken Sie die [Alt]-Taste und bewegen Sie danach die Maus, um Drehungen des Kopfes zu simulieren. Neigungen werden durch Drücken von [Strg] modelliert, während das Auslösen des im folgenden Abschnitts besprochenen Buttons durch einfache Mausklicks erfolgt.

Mit der Umgebung interagieren

Virtuelle Realität ist immer dann besonders interessant, wenn der Nutzer mit der Umgebung interagieren kann. Das Anzeigen von attraktiven dreidimensionalen Szenen ist nur ein Teil der Miete. Google stattet hauseigene Brillen mit einem Magnetschalter aus. Es handelt sich dabei um einen Magneten, der mit einer auf der Außenseite der Brille angebrachten Mutter bewegt werden kann.

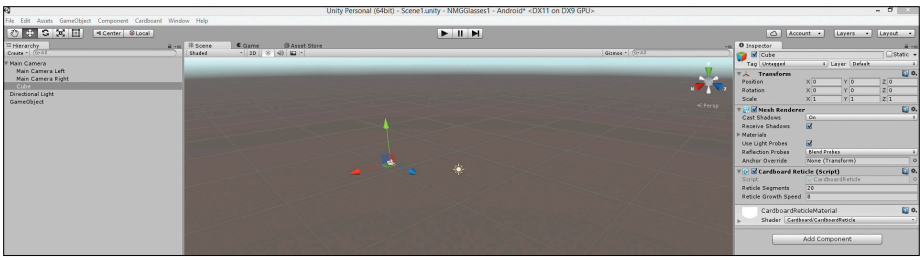
Im Idealfall erkennt ein am Telefon laufender Daemon Veränderungen am Magnetfeld, die in Knopfdruckereignisse umgewandelt werden. Diese Vorgehensweise ist in mehrerlei Hinsicht suboptimal: Daniel Kogel beschreibt unter www.bloculus.de/eine-eigene-cardboard-app-in-60-minuten-mit-unity5-free-teil-2-eigene-inhalte-und-character-controller, dass das Verfahren prinzipbedingt auf das Erkennen von Klicks beschränkt ist: Hält der User die Taste gedrückt, so ist dies ob des Fehlens der charakteristischen Feldänderung nicht feststellbar.

Die Breite des Android-Marktes schlug Google ein weiteres Schnippchen. In den Hardwarerichtlinien findet sich keine Anweisung für die Platzierung des Gyroskops. Jeder Hersteller kann den Sensor in seinem Telefon dort platzieren, wo er es für passend befindet. Daraus folgt, dass viele Telefone die Magnetfeldänderung nicht erkennen können. Wer sein App auf dem Knopf aufbaut, handelt sich in vielen Fällen Ärger ein.

Entwickler reagieren auf diese Situation auf zweierlei Arten: Bei teuren beziehungsweise produktiv verwendeten Applikationen bietet sich die Nutzung eines per Bluetooth LE ansprechbaren Knopfes beziehungsweise Controllers an. Spiele und preiswerte Apps arbeiten hingegen mit einem in der Mitte des Bildschirms eingeblendeten Cursor, der bei längerem Kontakt mit einem Element die dazugehörende Aktion auslöst.

Erstellen Sie als ersten Akt einen Cube, der im Inspector markiert und um die Elemente *Cube* und *Box Collider* erleichtert wird. Schreiben Sie dem Würfel sodann ein Element vom Typ

Cardboad, UI und *Cardboard Reticle* hinzu, um das Fadenkreuz in die Szene zu importieren. Selektieren Sie sodann die beiden unter *Assets, Cardboard, Resources* und *UI* bereitstehenden Elemente und werfen Sie sie per Drag and Drop in den Würfel. Danach müssen Sie nur noch die *Position*-Eigenschaften auf *0/0/0* setzen, um das in Bild 7 gezeigte Resultat zu erhalten.



Das Steuerelement ist am Platz (Bild 7)

Führen Sie die Szene sodann aus, um sich am in der Mitte des Bildschirms erscheinenden Fadenkreuz zu erfreuen. Wenn die Shader-Konfiguration richtig ist, schwebt es über allen anderen Objekten – wenn es nicht permanent sichtbar ist, so wurde zumeist das falsche Material ausgewählt.

Jagd auf Berührungen

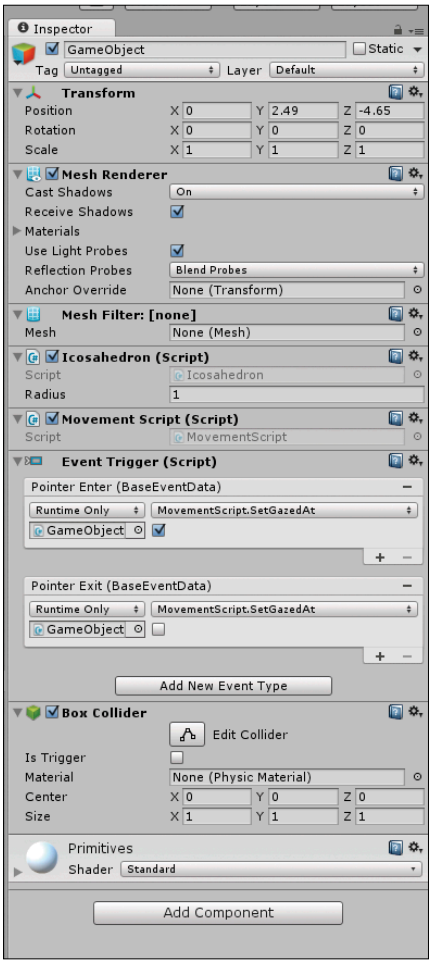
Das Einblenden eines Zielerfassungssystems ist nur dann sinnvoll, wenn der Nutzer damit auch wirklich interagieren kann. Dazu müssen wir unser Icosahederon mit einem Trigger ausstatten, der im Fall eines Kontakts zwischen Fadenkreuz und Element Laut gibt.

Öffnen Sie das im vorigen Abschnitt erstellte *MovementScript.cs* und ergänzen Sie es um den im Listing gezeigten Code:

```
public class MovementScript :
MonoBehaviour {
  ...
  void Start () {
    Camera aCamera = Camera.main;
    myCamPos =
    aCamera.transform.position;
    SetGazedAt(false);
  }
  void LateUpdate() {
    Cardboard.SDK.UpdateState();
  }
  public void SetGazedAt(bool
  gazedAt) {
    GetComponent<Renderer>().
    material.color = gazedAt ?
    Color.green : Color.red;
  }
```

SetGazedAt ist hierbei eine Hilfsfunktion, die beim Auftreten eines Kontakts zwischen Fadenkreuz und Ikosahederon aufgerufen wird. *LateUpdate()* ruft die Methode *Cardboard.SDK.UpdateState()* auf und gibt der Engine so Rechenleistung für Housekeeping-Tasks.

Ergänzen Sie das *GameObject* um ein Objekt vom Typ *Event Trigger (Script)*. Für das Handling der von Google Cardboard erzeugten Ereignisse sind die Methoden *Pointer Enter* und *Pointer Exit* notwendig, die beide im *BaseEventData*-Namespace sitzen.



Event Handler und Skript sind verdrahtet (Bild 8)

Die Konfiguration der Events erfolgt durch Anklicken des Feldes *None (Object)*. Wählen Sie im daraufhin erscheinenden Editor das *GameObject* aus, um das Dropdown-Menü *No Function* zu bevölkern. Falls Sie sich bei der Benennung der Elemente genau an den Vorgaben des Artikels orientiert haben, so lautet der korrekte Name wie in Bild 8 gezeigt *MovementScript->SetGazedAt()*.

Unity blendet im nächsten Schritt eine Checkbox neben dem Namen der Funktion ein. Diese Checkbox erlaubt das Festlegen des *bool*-Parameters. Setzen Sie an dieser Stelle den Wert des *Enter*-Events auf *true*.

Wer das Programm im vorliegenden Zustand ausführt, bemerkt keine Interaktion zwischen Fadenkreuz und Icosahederon. Das liegt daran, dass das GameObject im Moment keine physikalischen Dimensionen hat – ein Problem, dem sich durch Einfügen eines Box Colliders beikommen lässt.

Zudem ist der von Daniil Basmanov als Teil des *ProceduralToolkit* ausgelieferte Shader nicht in der Lage, auf Änderungen der Materialfarbe zu reagieren. Dies lässt sich durch Auswahl des Shaders *Standard* beheben.

Technische Verdrahtung

Zu guter Letzt müssen wir das Eventsystem der Szene als Ganzes anzapfen, um dem *CardboardReticle* das Ausliefern von Ereignisinformationen zu ermöglichen. Ergänzen Sie die Szene dazu um ein Objekt vom Typ *UI->EventSystem*, dem eine Instanz von *GazeInputModule* eingeschrieben wird. Achten Sie dabei darauf, dass sie in der Reihenfolge über eventuellen anderen Elementen (Stichwort Touch Input Module) zu liegen kommt.

Markieren Sie im nächsten Schritt die *Main Camera* und pflegen Sie über *Add Component* einen dreidimensionalen *Physics Raycaster* hinzu. Diese Komponente ermöglicht dem Eventsystem ▶

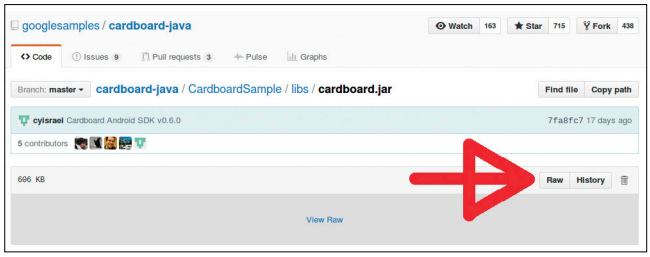
das Erkennen der Blickrichtung der Kamera. Der Lohn der Mühen ist eine Kugel, die ihre Farbe bei Kontakt mit dem Fadenkreuz verändert.

Vor Kurzem hat Google eine immens wichtige Erweiterung für das Cardboard-SDK vorgestellt: Ein als Spatial Audio bezeichnetes Feature ermöglicht die Realisierung von realistischen Tonquellen.

Auch wenn HTC und Co. seit einiger Zeit Stereolautsprecher in ihre Telefone verbauen, tritt wirklich gute räumliche Wirkung nur mit Kopfhörern auf. Nutzer akzeptieren Kopfhörer-Aufsetz-Dialoge im Allgemeinen problemlos. Eine oberflächliche Analyse des Verhaltens von Rezensenten ergab zudem, dass Blogger dem Sound von Applikationen mit einem derartigen Dialog im Allgemeinen wohlwollender gegenüberstehen.

Seine Nutzung setzt das Einpflegen eines Skripts vom Typ *Cardboard Audio Listener* voraus. In der Regel kommt dieses Skript in der *Main Camera* zu liegen. Wechseln Sie sodann in *Edit, Project Settings* und *Audio* und setzen Sie die Eigenschaft *Spatializer Plugin* auf *CardboardAudio Spatializer.*

Im nächsten Schritt müssen die einzelnen Tonquellenobjekte mit Skripts der Bauart *Cardboard Audio Source* ausgestattet werden. Der Rest der unter https://developers.google.com/cardboard/unity/guide#spatial_audio_for_vr beschriebenen Konfiguration verhält sich im Großen und Ganzen wie unter Unity gewohnt.



GitHub liefert – nach etwas gutem Zureden – auch eine Rohdatei aus (Bild 9)

Und nun mit Java

Unity erleichtert Entwicklern das Leben insofern, als es das Hantieren mit OpenGL überflüssig macht. Da es eine Vielzahl von existierenden Applikationen auf OpenGL-Basis gibt, wollen wir uns im nächsten Schritt der Nutzung des Java-API zuwenden.

Google unterstützt – wie sollte es auch anders sein – nur noch Android Studio. Die folgenden Schritte basieren auf Version 1.3.2 der IDE. Erstellen Sie ein neues Projekt, das als Minimum-SDK-Level die API-Version 16 avisiert. Witzigerweise liefert Google das Cardboard-SDK selbst nicht per Gradle aus. Öffnen Sie daher den URL https://github.com/googlesamples/cardboard-java/blob/master/CardboardSample/libs/cardboard.jar in einem Browser Ihrer Wahl und klicken Sie auf den in Bild 9 hervorgehobenen RAW-Button, um die JAR-Datei herunterzuladen.

Kopieren Sie diese in das */libs*-Unterverzeichnis des App-Moduls. Ist dieses nicht sichtbar, so müssen Sie den Darstellungsmodus des Fensters von *Android* auf *Project* umstellen. Öffnen Sie im nächsten Schritt *build.gradle* – die Datei muss unbedingt das folgende Snippet enthalten, das den Inhalt des *libs*-Ordners zur Inklusion anweist:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  testCompile 'junit:junit:4.12'
  compile 'com.android.support:appcompat-v7:23.1.1'
  compile 'com.android.support:design:23.1.1'
}
```

OpenGL ist – auch in der für Smartphones vorgesehenen ES-Variante – derart umfangreich, dass ein Versuch der Beschreibung in einem Einzelartikel von vornherein zum Scheitern verurteilt ist. Aus diesem Grund nutzen wir für unser Beispiel von Google bereitgestellte Ressourcen und besprechen das OpenGL-API nur insofern, als es für das Verständnis des Cardboard-API erforderlich ist.

Laden Sie dazu Googles *cardboard-java* auf Ihre Workstation herunter. Unter Unix arbeitende Entwickler bewerkstelligen diese Aufgabe im Idealfall auf der Kommandozeile durch Eingabe des Statements *git clone https://github.com/googlesamples/cardboard-java.git.* Für uns sind die drei unter */res/raw* liegenden Shader-Dateien von besonderer Relevanz. Kopieren Sie sie einfach eins zu eins in dasselbe Verzeichnis des erstellten Projektskeletts.

WorldLayoutData.java enthält die Koordinaten für einen Würfel und eine Bodenfläche. Importieren Sie die Datei in den */src-Ordner* und passen Sie die Package-Deklaration zwecks einfacherer Einbindung an die Gegebenheiten Ihres Projektskeletts an.

Das native Cardboard-SDK stellt umfangreiche Anforderungen an die Manifest-Datei des enthaltenden Projekts. Neben einer Gruppe von Permissions und dem Vorhandensein von GL ES 2.0 ist auch eine spezielle Activity samt *IntentFilter* erforderlich (Listing 1).

Fundamentales

Nach der Anpassung der Manifest-Datei ist die Activity an der Reihe: Cardboard-Ansichten entstehen prinzipbedingt immer in OpenGL. Aus diesem Grund ist der Inhalt der Dateien *activity_main* und *content_main* überflüssig. Das zweite File wird ersatzlos gelöscht, während die erste Datei das in Listing 2 gezeigte XML-Dokument eingeschrieben bekommt.

CardboardView ist eine von *GLSurfaceView* abgeleitete Convenience-Klasse, die das Rendern von stereoskopischen Inhalten erleichtert. Sie nimmt normalerweise den gesamten Bildschirmplatz ein. Das *RelativeLayout* ist im Großen und Ganzen nur der Form halber am Platz.

Google unterstützt mit Cardboard arbeitende Entwickler im Code Behind durch eine dedizierte Activityklasse, die einige Hilfsmethoden und eine eigene Rendering-Instanz mitbringt. Öffnen Sie *MainActivity.java* und ersetzen Sie die Ableitung von *AppCompatActivity* durch einen Verweis auf

CardboardActivity. Entfernen Sie sodann die beiden Event Handler für das Optionsmenü und reduzieren Sie den Inhalt von *onCreate* auf folgendes Snippet:

```
public class MainActivity extends CardboardActivity
{
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    CardboardView cardboardView = (CardboardView)
    findViewById(R.id.cardboard_view);
    cardboardView.setRenderer(this);
    setCardboardView(cardboardView);
  }
}
```

CardboardActivities unterscheiden sich bei der Erstellung nur wenig von normalen Formularen. Nach dem Aufrufen der *onCreate*-Methode der Superklasse und dem Laden der in der XML-Datei vorliegenden Layoutinformationen weisen wir der *CardboardView* einen Renderer zu. *SetCardboardView* informiert den GUI-Stack sodann, welche Instanz von *CardboardView* für die Steuerung des Bildschirminhalts zuständig sein soll.

Listing 1: Manifest-Datei

```
<manifest ...
  <uses-permission android:name=
  "android.permission.NFC" />
  <uses-permission android:name=
  "android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name=
  "android.permission.WRITE_EXTERNAL_STORAGE" />

  <uses-permission android:name=
  "android.permission.VIBRATE" />
  ...
  <uses-sdk android:minSdkVersion="16"
  android:targetSdkVersion="19"/>
  <uses-feature android:glEsVersion="0x00020000"
  android:required="true" />
  <application
  ...
    <activity
      android:name=".MainActivity"
      android:screenOrientation="landscape">
      ...

      <intent-filter>
        <action android:name=
        "android.intent.action.MAIN" />
        <category android:name=
        "android.intent.category.LAUNCHER" />
        <category android:name=
        "com.google.intent.category.CARDBOARD" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Platzierung des Renderers

Es gibt kein traditionelles systemtechnisches Feature, das Entwickler zu einer bestimmten Platzierung des Renderers verurteilt. Rein theoretisch könnten Sie die für die Darstellung der Szene notwendige Intelligenz auch in eine eigene Klasse auslagern. In den meisten Applikationen findet sie sich im Code Behind der Activity.

Das native Cardboard-SDK bietet dabei zwei verschiedene Interfaces an. *Renderer* ist für all jene Applikationen vorgesehen, die die Grafikdarstellung komplett von Hand erledigen wollen:

```
public interface Renderer {
  void onDrawFrame(HeadTransform var1, Eye var2,
  Eye var3);
  void onFinishFrame(Viewport var1);
  void onSurfaceChanged(int var1, int var2);
  void onSurfaceCreated(EGLConfig var1);
  void onRendererShutdown();
}
```

Das Betriebssystem ruft *onDrawFrame* auf, wenn die Animation eine Runde weitergeht. In den beiden Eye-Parametern liefert Cardboard dabei weitere Informationen über das Sehfeld des jeweiligen Auges an.

Die aus mathematischer Sicht durchaus anspruchsvolle Erstellung der stereoskopischen Ansicht ist bis auf die Korrektur der perspektivischen Verzerrung die alleinige Aufgabe des Entwicklers. *OnFinishFrame* erlaubt das Rendern von Overlays – die Methode wird erst dann aufgerufen, wenn der Frame die perspektivische Korrektur durchlaufen hat. ▶

Listing 2: activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
  android:id="@+id/ui_layout"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <com.google.vrtoolkit.cardboard.CardboardView
    android:id="@+id/cardboard_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true" />
</RelativeLayout>
```

Wir setzen in den folgenden Schritten stattdessen auf *StereoRenderer,* dessen Deklaration folgendermaßen aussieht:

```
public interface StereoRenderer {
  void onNewFrame(HeadTransform var1);
  void onDrawEye(Eye var1);
  void onFinishFrame(Viewport var1);
  void onSurfaceChanged(int var1, int var2);
  void onSurfaceCreated(EGLConfig var1);
  void onRendererShutdown();
}
```

Dieses Interface unterstützt Entwickler bei der Realisierung der Renderpipeline durch das Anbieten eines stufigen Workflows. Der Lebenszyklus eines neuen Frames beginnt mit dem Aufruf von *onNewFrame.* Die in *HeadTransform* angelieferten Daten teilen der Applikation mit, wie sich der Kopf des Benutzers bewegt hat. Im nächsten Schritt folgen zwei Aufrufe von *onDrawEye,* die der Generierung der Perspektiven der beiden Augen dienen. Zu guter Letzt gibt es abermals einen Aufruf von *onFinishFrame.*

Ergänzen Sie die *MainActivity* um einen *implements*-Verweis auf *StereoRenderer* und lassen Sie Android Studio die fehlenden Methodenskelette automatisch einfügen.

Es ist Zeit, das Auf-den-Bildschirm-Bringen des Würfels zu avisieren:

```
public class MainActivity extends CardboardActivity
implements CardboardView.StereoRenderer {
```

Unity erleichterte uns die Darstellung des umherwandernden Icosahederons, weil sich die Engine um die Berechnung der Flächen, das Shading und die Einfärbung kümmerte. Bei der Arbeit mit OpenGL ist die Sache insofern anders, als diese Aufgabe nun allein auf den Schultern des Entwicklers lastet.

Als ersten Akt müssen wir uns mit dem Laden der Shader-Programme befassen. Google implementiert dies in seinem Beispiel durch zwei in *MainActivity.java* befindliche Methoden. Kopieren Sie sowohl *loadGlShader* als auch *readRawTextFile* in den Korpus der Activity. Die beiden Methoden sind konventioneller Ressourcenzugriffscode, der den eingelesenen Text unter Nutzung einer OpenGL-Funktion in ein ausführbares Programm umwandelt.

Unsere Renderingklasse wird durch Abfeuern des *onSurfaceCreated*-Ereignisses darüber informiert, dass die OpenGL-Umgebung nun zur Verfügung steht. Wir nutzen diese Information zum Laden und der Verkapselung einiger weiterer Ressourcen:

```
@Override
public void onSurfaceCreated(EGLConfig eglConfig) {
  GLES20.glClearColor(0.1f, 0.1f, 0.1f, 0.5f);
  ByteBuffer bbVertices = ByteBuffer.allocateDirect
  (WorldLayoutData.CUBE_COORDS.length * 4);
  bbVertices.order(ByteOrder.nativeOrder());
  myCubeVertices = bbVertices.asFloatBuffer();
  myCubeVertices.put(WorldLayoutData.CUBE_COORDS);
  myCubeVertices.position(0);
  ...
```

OpenGL bleibt – auch bei Nutzung des Java-Interfaces – eine extrem systemnahe Programmiersprache. Eine gravierende Auswirkung davon ist das verwendete Datenformat: GPU-Code kann mit Java-Arrays nichts anfangen. Wir umgehen dieses Problem durch manuelles Umkopieren der Daten, die fortan in an C-Arrays erinnernden FloatBuffern auf ihren Einsatz warten.

Als nächster Akt folgen das Laden der Shader und die Erstellung der eigentlichen, für das Rendering erforderlichen Pipeline-Elemente. Wir geben den Code hier absichtlich gekürzt an. Gehen Sie davon aus, dass die Konfiguration nach dem Abarbeiten der Methode *onSurfaceCreated* erfolgreich abgeschlossen wurde:

```
int vertexShader = loadGLShader(GLES20.GL_VERTEX_SHADER,
R.raw.light_vertex);
int passthroughShader = loadGLShader
(GLES20.GL_FRAGMENT_SHADER, R.raw.passthrough_fragment);
cubeProgram = GLES20.glCreateProgram();
GLES20.glAttachShader(cubeProgram, vertexShader);
GLES20.glAttachShader(cubeProgram, passthroughShader);
GLES20.glLinkProgram(cubeProgram);
GLES20.glUseProgram(cubeProgram);
cubePositionParam = GLES20.glGetAttribLocation
(cubeProgram, "a_Position");
...
GLES20.glEnableVertexAttribArray(cubePositionParam);
```

Von XNA und anderen Managed-Frameworks umsteigende Entwickler wundern sich hier mitunter über die Rückgabewerte. OpenGL arbeitet mit numerischen IDs. So gut wie alle Methoden liefern einen Integer zurück, der das betreffende Element im internen Speicher der Laufzeitumgebung eindeutig identifiziert.

Listing 3: View in den GUI-Stack

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout ...>
  <com.google.vrtoolkit.cardboard.CardboardView
    android:id="@+id/cardboard_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true" />
  <com.tamoggemon.nmgcardboard1.COverlayView
    android:id="@+id/overlay"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true" />
</RelativeLayout>
```

Damit sind wir an dieser Stelle fertig und können uns der Realisierung der eigentlichen Frame-Zeichenlogik zuwenden. In OpenGL sind Modelle nur ein Haufen von Vertizes, die durch eine Serie von Matrixmultiplikationen auf den Bildschirm gebracht werden. Konzepte wie Position, Rotation oder axonometrische Streckung sind nicht als dedizierte Eigenschaften angelegt. Sie werden stattdessen über die Weltmatrix abgebildet.

Hierzu eine kleine Erklärung: Android geht davon aus, dass Modelle im Zahlenraum von -1 bis 1 leben. Die Weltmatrix hat die Aufgabe, das Modell an seinen endgültigen Aufenthaltsort in der Spielwelt zu bringen. Neben Rotationen und Skalierungen wird hierbei auch eine Transformation angewendet.

Im zweiten Schritt folgt die View-Matrix. Sie verschiebt die Objekte in einen anderen Koordinatenraum, der von der für die Anzeige zuständigen Kamera bestimmt wird. Bei der Programmierung von VR-Applikationen spielt diese Matrix insofern eine besondere Rolle, als sie die Verschiebung zwischen den beiden Augen abbildet. Zu guter Letzt folgt die Projektionsmatrix, die die Elemente aus dem dreidimensionalen Raum in eine zweidimensionale Ebene befördert.

Jeder Frame beginnt mit einem Aufruf von *onNewFrame*. Unsere erste Amtshandlung besteht darin, die für die Rotation zuständigen Daten zu inkrementieren. Im nächsten Schritt wird die für die Modelldarstellung zuständige Weltmatrix normalisiert und mit einer Verschiebeoperation ausgestattet, die die Quader vor den Kopf des Nutzers verschiebt. Als nächster Akt wird die Methode *rotateM* aufgerufen, die das Gesamtkonvolut um den in Grad anzugebenden Winkel verschiebt:

```
@Override
public void onNewFrame(HeadTransform headTransform) {
  modelPosition = new float[]{0.0f, 0.0f, -7.0f / 2.0f};
  degreeCounter+=1.4;
  Matrix.setIdentityM(modelCube, 0);
  Matrix.translateM(modelCube, 0, modelPosition[0],
  modelPosition[1], modelPosition[2]);
  Matrix.rotateM(modelCube, 0, degreeCounter,
  1.0f, 0f, 0f);
  Matrix.setLookAtM(camera, 0, 0.0f, 0.0f, 0.1f,
  0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
}
```

Nach dem Aktualisieren des Aufenthaltsorts und der Generierung einer neuen Weltmatrix folgen zwei Aufrufe der für das eigentliche Zeichnen zuständigen Methode *onDrawEye*. Sie sieht in unserem Beispiel folgendermaßen aus:

```
@Override
public void onDrawEye(Eye eye) {
  GLES20.glEnable(GLES20.GL_DEPTH_TEST);
  GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT |
  GLES20.GL_DEPTH_BUFFER_BIT);
  Matrix.multiplyMM(view, 0, eye.getEyeView(), 0,
  camera, 0);
  Matrix.multiplyMV(lightPosInEyeSpace, 0, view, 0,
  LIGHT_POS_IN_WORLD_SPACE, 0);
  float[] perspective = eye.getPerspective( 0.1f,
  100.0f);
  Matrix.multiplyMM(modelView, 0, view, 0,
  modelCube, 0);
  Matrix.multiplyMM(modelViewProjection, 0,
  perspective, 0, modelView, 0);
  drawCube();
}
```

Nach dem Setzen einiger für den Betrieb der Grafikpipeline wichtigen Konfigurations-Flags folgt die Transformation der *view*-Matrix mit der von *eye.getEyeView()* zurückgelieferten Konstante. Das Resultat dieser Operation ist eine View-Matrix, die das Blickfeld des gerade aktiven Auges beschreibt.

Als nächste Aufgabe folgt die Beschaffung einer für das jeweilige Sehorgan zuständigen Perspektivmatrix. Sie wird mit der *ModelView* und der View-Matrix zur *ModelViewProjection*-Matrix vereint, die für die Grafikdarstellung bereit ist.

Der eigentliche Anzeigeprozess ist sodann in *drawCube()* enkapsuliert. Er nutzt die im Rahmen der Initialisierung des Formulars erzeugten Programme und Shader-Fragmente zur Anzeige des Würfels. Da es sich dabei um reines OpenGL handelt, wird der Code an dieser Stelle nicht abgedruckt.

Zielerfassung von Hand

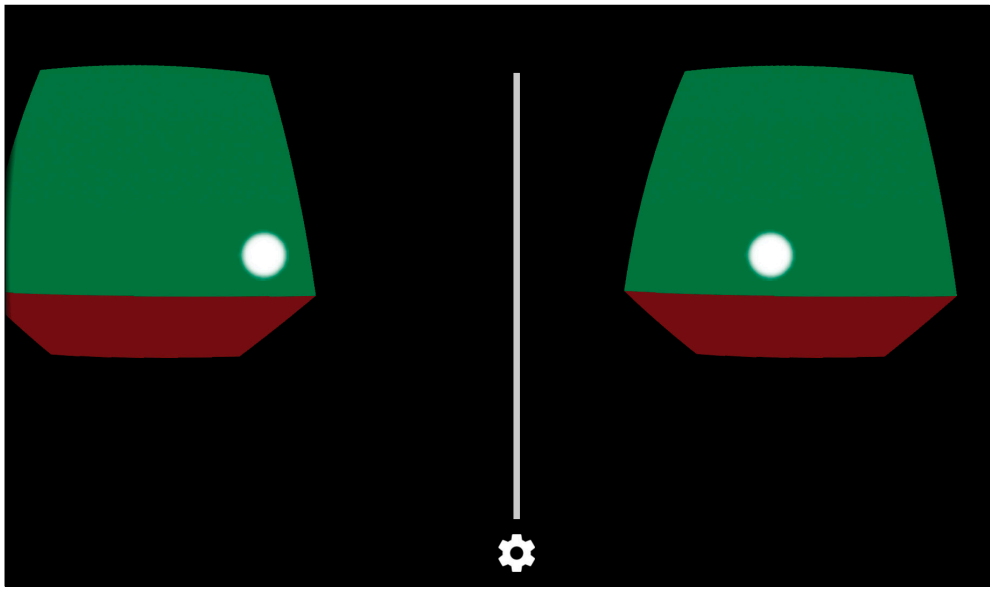
Unter Unity entfiel ein Gutteil der Arbeit auf die Realisierung einer Art Zielerfassung. Neben dem Einpflegen eines ▶

Listing 4: onLayout-Methode

```
@Override
protected void onLayout(boolean changed, int left,
int top, int right, int bottom) {
  final int width = right - left;
  final int height = bottom - top;
  final float imageSize = 0.1f;
  final float verticalImageOffset = -0.07f;
  float adjustedOffset = myOffset;

  if (width > 1000) {
   adjustedOffset = 3.8f * myOffset;
  }
  float imageMargin = (1.0f - imageSize) / 2.0f;
  float leftMargin = (int) (width * (imageMargin +
  adjustedOffset));
  float topMargin = (int) (height * (imageMargin +
  verticalImageOffset));
  imageView.layout(
    (int) leftMargin, (int) topMargin,
    (int) (leftMargin + width * imageSize), (int)
    (topMargin + height * imageSize));
  }
}
```

Eventsystems mussten wir unsere Kamera auch um einen PhysicsRaycaster ergänzen.

Mit dem nativen API arbeitende Entwickler beginnen ihre Reise normalerweise mit der Erstellung eines Fadenkreuzes. Dies funktioniert idealerweise durch eine OverlayView, die über das per OpenGL angelieferte Rendering-Resultat gezeichnet wird. Als ersten Akt müssen Sie das Formular um das XML-Snippet in Listing 3 erweitern, um die View in den GUI-Stack einzubinden.



Das Zielvisier ist einsatzbereit (Bild 10)

Dieses Snippet ist vor allem deshalb interessant, weil die beiden Ebenen identische Layoutparameter erhalten. Das bedeutet, dass die Views am Bildschirm eins zu eins übereinanderliegen. Erstellen Sie im nächsten Schritt eine Klasse namens *CoverlayView*. Sie ist ein gewöhnliches LinearLayout, das im Rahmen seiner Erzeugung einen Layoutparametersatz generiert:

```
public class COverlayView extends LinearLayout {
  private final COverlayHelper myLeft;
  private final COverlayHelper myRight;
  public COverlayView(Context context,
  AttributeSet attrs) {
    super(context, attrs);
    setOrientation(HORIZONTAL);
    LayoutParams params = new LayoutParams(
      LayoutParams.MATCH_PARENT,
      LayoutParams.MATCH_PARENT, 1.0f);
    params.setMargins(0, 0, 0, 0);
```

Im zweiten Teil des Konstruktors folgt die Erstellung der beiden Instanzen von *COverlayHelper*. Sie werden per *setLayoutParams* mit den im vorigen Schritt erzeugten Layoutregeln ausgestattet, um die korrekte Darstellung zu gewährleisten:

```
    myLeft = new COverlayHelper(context, attrs);
    myLeft.setLayoutParams(params);
    addView(myLeft);
    myRight = new COverlayHelper(context, attrs);
    myRight.setLayoutParams(params);
    addView(myRight);
    myLeft.myOffset=0.01f;
    myRight.myOffset=-0.01f;
    setVisibility(View.VISIBLE);
  }
```

COverlayView ist im Grunde genommen nur ein Container für zwei weitere Klassen, die aus ein- und derselben Basisklasse entstehen. *COverlayHelper* sieht in unserem Beispiel folgendermaßen aus:

```
public class COverlayHelper extends ViewGroup {
  private final ImageView imageView;
  public float myOffset;
  public COverlayHelper(Context context,
  AttributeSet attrs) {
    super(context, attrs);
    imageView = new ImageView(context, attrs);
    imageView.setScaleType
    (ImageView.ScaleType.CENTER_INSIDE);
    imageView.setAdjustViewBounds(true);
    imageView.setImageResource(R.drawable.reticle);
    addView(imageView);
  }
```

Der Konstruktor erstellt eine neue *ImageView*, die mit einem im *Drawable*-Ordner abzulegenden Bild ausgestattet wird. Die eigentliche Intelligenz der Klasse findet sich jedoch in der *onLayout*-Methode, deren Korpus von Google im Sinne bestmöglicher stereoskopischer Darstellung wie in Listing 4 vorparametriert wird.

Führen Sie das Programm im nächsten Schritt aus, um die in Bild 10 gezeigten Darstellung eines Fadenkreuzes zu sehen.

Visier, die Zweite

Als letzte Aufgabe wollen wir uns der Ermittlung von Kontakt zwischen Visier und Würfel zuwenden. Dazu ist etwas Ma-

Link zum Thema

- Google Cardboard-Projektseite
  https://www.google.com/get/cardboard

thematik erforderlich. Wir müssen den Differenzwinkel zwischen Blick- und Modellachse ermitteln, um so festzustellen, ob das Objekt im heißen Bereich zu liegen kommt.

Als ersten Akt müssen wir dazu die in *onNewFrame* errechnete View-Matrix in einer Variablen zwischenspeichern, über die sie später abgerufen werden kann:

```
@Override
public void onNewFrame(HeadTransform headTransform)
{
  ...
  Matrix.setLookAtM(camera, 0, 0.0f, 0.0f, 0.1f,
  0.0f, 0.0f, 0.0f, 0.0f,
  1.0f, 0.0f);
  headTransform.getHeadView
  (myHeadView, 0);
}
```

Im zweiten Schritt folgt das Einfügen einer Methode, die die eigentliche mathematische Berechnung durchführt. Im ersten Schritt wird das Objekt in den Kameraraum transformiert. *atan2* dient sodann zur Ermittlung der Differenzwinkel zwischen den Achsen. Wenn sie innerhalb eines gewissen Grenzwerts zu liegen kommen, so geht die Engine von Kontakt aus:

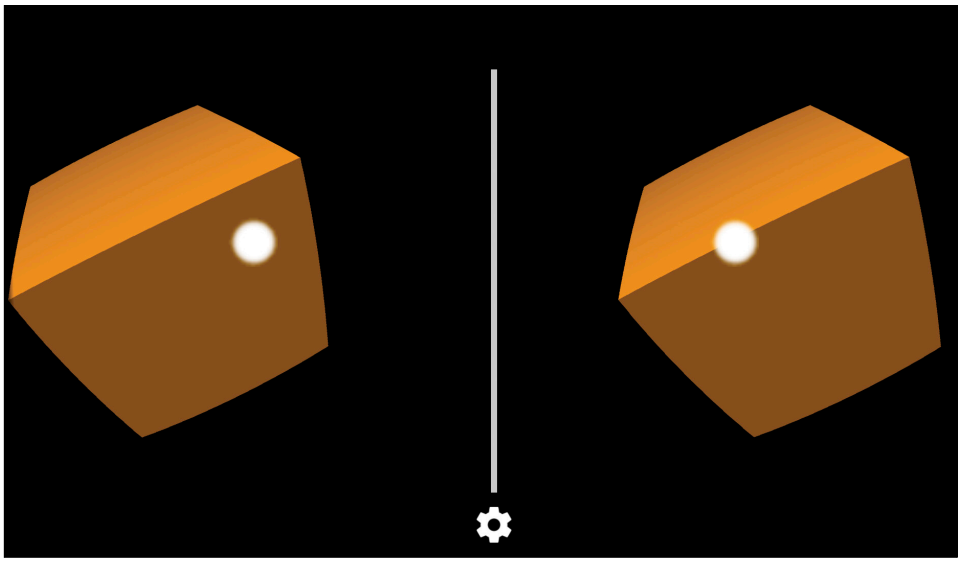
```
private static final float
YAW_LIMIT = 0.12f;
private static final float
PITCH_LIMIT = 0.12f;
private boolean isLookingAt-
Object() {
  float[] initVec = {0, 0, 0, 1.0f};
  float[] objPositionVec = new float[4];
  Matrix.multiplyMM(modelView, 0, myHeadView, 0,
  modelCube, 0);
  Matrix.multiplyMV(objPositionVec, 0, modelView,
  0, initVec, 0);
  float pitch = (float) Math.atan2(objPositionVec[1],
  -objPositionVec[2]);
  float yaw = (float) Math.atan2(objPositionVec[0],
  -objPositionVec[2]);
  return Math.abs(pitch) < PITCH_LIMIT &&
  Math.abs(yaw) < YAW_LIMIT;
}
```

Damit müssen wir nur noch die Zeichenmethode des Würfels anpassen, um den verwendeten Farbvektor bei bestehendem Kontakt durch einen anderen zu ersetzen:

```
public void drawCube() {
  ...
  GLES20.glVertexAttribPointer(cubeNormalParam, 3,
  GLES20.GL_FLOAT, false, 0, myCubeNormals);
  GLES20.glVertexAttribPointer(cubeColorParam, 4,
  GLES20.GL_FLOAT, false, 0,
    isLookingAtObject() ? myCubeFoundColors :
    myCubeColors);
  GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, 36);
}
```

Der Lohn der Mühen ist der Bild 11 gezeigte Farbwechselprozess. Die Auswahl der Grenzwerte für Pitch und Yaw ist dabei eine Wissenschaft für sich.

Es ist in höchstem Maße empfehlenswert, nicht zu strikt zu sein. Das genaue Ausrichten des Kopfes ist nicht nur ob der systemimmanenten Parallaxfehler alles andere als einfach.



Dieser Würfel hat Kontakt mit dem Visier (Bild 11)

Fazit

Cardboard könnte Google den Erfolg bringen, der Glass bisher verwehrt blieb. Das geradezu genial einfache API sorgt dafür, dass die Integration in vorhandenes geistiges Eigentum rasch von der Hand geht.

Wer 3D-Content für Unity oder OpenGL besitzt, sollte die mit weniger als 10 Euro außerordentlich preiswerte Brille unbedingt einem eingehenden Test unterziehen. ■

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Es lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter:
www.tamoggemon.com

RECHT FÜR BLOGGER UND CO.

Rechtlicher Rahmen

Bloggen ist heutzutage so einfach … und doch so kompliziert.

Will man sich regelmäßig online mitteilen, stehen heute viele unterschiedliche Werkzeuge zur Verfügung. Dazu zählen klassische Blogs (mittels WordPress, Joomla et cetera) ebenso wie Status-Updates in sozialen Netzwerken (Facebook, Twitter, Whatsapp, Instagram), Vlogs (Video-Blogs, etwa via YouTube oder Twitch) oder auch Podcasts (Bild 1).

Unabhängig vom gewählten Medium haben alle diese Dinge einen gemeinsamen Nenner: Neben den regelmäßigen Inhalten muss auch der juristische Rahmen stimmen.

Mit Open-Source-Software wie WordPress oder Joomla lassen sich schnell eigene Blogs im Web realisieren (Bild 1)

Inhalte

Ohne eigene beziehungsweise fremde Inhalte wäre jeder Blog, jedes Profil in sozialen Netzwerken oder jeder Podcast nur eine leere Hülle. Um mit einem Online-Projekt erfolgreich zu sein, ja um überhaupt in ausreichender Weise wahrgenommen zu werden, gilt es laufend neue und im Idealfall interessante Inhalte online bereitzustellen.

Einen entsprechend hohen Stellenwert sollte daher das Urheberrecht haben. Denn nur, weil es im Internet aus technischer Sicht denkbar einfach ist, fremde Texte, Bilder, Videos oder Musikstücke zu übernehmen, bedeutet das noch lange nicht, dass dies in jedem Fall auch zulässig ist.

Grundsätzlich verwenden kann man folgenden Content:

- Eigene Inhalte (die nicht gegen Gesetze verstoßen oder sittenwidrig sind).
- Fremde Inhalte mit Einwilligung für den konkreten Verwendungszweck (etwa Print- und Online-Nutzung).
- Inhalte ohne ausreichende Schaffenshöhe (Abgrenzung schwierig).
- Sogenannte gemeinfreie Inhalte (zum Beispiel Urteils- oder Gesetzestexte).
- Zitate (als Beleg für eine eigene Leistung, mit Kennzeichnung und Quellennachweis).
- Fotos von Personen mit deren Einwilligung.

Im Online-Bereich spielen Zitate eine besondere Rolle. Generell darf man Teile fremder Werke, also etwa einen bestimmten Abschnitt eines Textes, für eigene Zwecke verwenden. Dabei sind jedoch folgende Aspekte zu beachten:

- Jedes Zitat muss ein Beleg für eigene Leistung sein.
- Die eigene Leistung muss im Vordergrund stehen.
- Das Zitat muss als solches gekennzeichnet werden.
- Es können ganze Werke oder einzelne Werkteile zitiert werden (Groß-/Kleinzitat).
- Das Zitat muss eine Quellenangabe enthalten.

Ein Beispiel für ein korrektes Zitat sieht also wie folgt aus: *»Ein Zitat […] ist eine wörtlich übernommene Stelle aus einem Text oder ein Hinweis auf eine bestimmte Textstelle.« (Quelle: Wikipedia, de.wikipedia.org/wiki/Zitat).* Bei Zitaten aus Online-Quellen ist es ideal, wenn die entsprechende Site direkt verlinkt wird.

Fremde Werke dürfen auch ohne Kennzeichnung als Zitat übernommen werden, wenn sie so bearbeitet oder verändert werden, dass dadurch ein komplett neues, eigenständiges Werk entsteht. Dazu hat der Bundesgerichtshof die sogenannte Verblassen-Formel entwickelt. Die besagt, dass die Bearbeitung eines fremden Werkes zulässig ist, soweit die Züge des ursprünglichen Werks nur noch schwach durchschimmern. Als Faustformel kann man sich also merken: Je mehr das Originalwerk als solches noch zu erkennen ist, desto eher ist von einer unzulässigen Bearbeitung auszugehen.

Eine exakte Differenzierung, ab wann eine Bearbeitung zulässig und bis wohin sie noch als unzulässig zu bewerten ist, gestaltet sich oft auch für fachkundige Juristen schwierig. Im Zweifel gilt daher: Die selbst erstellten Inhalte sind noch immer die besten – jedenfalls aus urheberrechtlicher Sicht.

Äußerungen

Die Aussage, dass die eigenen Inhalte immer noch die besten sind, erfährt eine wichtige Einschränkung. Die Inhalte selbst dürfen natürlich nicht gegen geltende Gesetz verstoßen oder sittenwidrig sein. In erster Linie geht es hier um die Differenzierung von Tatsachen- und von Meinungsäußerungen.

Grundsätzlich erlaubt sind wahre Tatsachenbehauptungen und die Wiedergabe der eigenen (subjektiven) Meinung; unwahre Tatsachenbehauptungen sind hingegen untersagt.

Die Meinungsfreiheit ist insoweit eingeschränkt, als sie nicht aus reiner Schmähkritik bestehen darf. Zwar hat niemand einen Anspruch darauf, ausschließlich positiv oder freundlich behandelt zu werden. Auch überzogene oder gar ausfällige Kritik macht für sich genommen eine Äußerung noch nicht rechtswidrig. Schmähkritik ist erst dann anzunehmen, wenn nicht mehr die Auseinandersetzung in der Sache, sondern die Diffamierung der Person im Vordergrund steht.

Haftung

Grundsätzlich besteht für eigene Inhalte eine uneingeschränkte Haftung – diesbezüglich gilt online nichts anderes als im richtigen Leben auch. Weil aber das Medium Internet naturgemäß einige Besonderheiten mit sich bringt, kann unter Umständen auch eine Haftung für Inhalte Dritter bestehen. Eine solche kann unter den folgenden Voraussetzungen angenommen werden, nämlich

- bei Kenntnis der rechtswidrigen Inhalte und gleichzeitiger Untätigkeit,
- bei Zueigenmachen der rechtswidrigen Inhalte oder auch
- bei der Provokation von rechtswidrigen Äußerungen Dritter.

Faustregel: Erfährt man von möglichen Rechtsverstößen, muss schnellstmöglich reagiert werden. Die betreffenden Inhalte müssen also gelöscht oder zumindest gesperrt werden.

Impressum

Jede nicht nur rein privat genutzte Internetpräsenz muss ein Impressum und dort bestimmte Pflichtangaben bereitstellen. Die Grenze der reinen Privatnutzung ist bereits dann überschritten, wenn der Blog-Betreiber selbst Werbung einbindet, also beispielsweise Google AdWords.

Die von einem Dritten, etwa Facebook oder YouTube, veranlasste Werbung fällt nicht darunter; andernfalls wären im Grunde alle Profile in den sozialen Medien als nicht mehr rein privat einzustufen. Als Zweifelsregelung gilt die Entscheidung pro Impressum.

Im Idealfall gibt es also einen eigenständigen Menüpunkt *Impressum,* der von jeder einzelnen Unterseite aus gleich gut erreichbar ist. Dazu dürfen nicht mehr als zwei Mausklicks erforderlich sein, eine Platzierung des Impressums auf der dritten Menü-Unterebene verbietet sich also.

In sozialen Medien wie auch auf anderen Plattformen, wie zum Beispiel YouTube, ist von Seiten des jeweiligen Betreibers nicht immer ein eigenständiger Menüpunkt vorgesehen, der die gesetzlichen Vorgaben hierzulande erfüllt. Bisweilen ist dies aufgrund einer Zeichenbegrenzung, wie etwa bei Twitter, auch schon rein technisch nicht zu realisieren.

Insofern kann es vorkommen, dass man sich mit Notlösungen behelfen muss. Gut und auch zulässig ist es etwa, bei den eigenen Kontaktangaben einen sprechenden Link einzubinden, der wiederum auf das eigene Website-Impressum verweist. Das setzt aber natürlich voraus, dass auch eine eigene Website existiert und dass das dortige Impressum einen Rückverweis auf den Blog, die sozialen Medien, YouTube et cetera enthält. Ein solcher Verweis kann beispielsweise folgendermaßen lauten: *»Verantwortlich im Sinne des Telemedienrechts für die Domain xyz.de sowie für die Facebook-Seite Facebook.com/xyz, das Twitter-Profil Twitter.com/xyz und für die Xing-Präsenz Xing.com/profile/xyz:«*

Inhaltlich müssen diverse Pflichtangaben berücksichtigt werden:

- **Name:** vollständiger, ausgeschriebener Vor- und Nachname beziehungsweise Unternehmensbezeichnung,
- **Adresse:** ladungsfähige Anschrift,
- **Handelndes Organ:** Vertretungsberechtigter des Unternehmens (zum Beispiel GmbH-Geschäftsführer),
- **Haftung:** inhaltlich Verantwortlicher,
- **Kontaktdaten:** Telefonnummer und Mail-Adresse,
- **Registerangaben:** Registergericht und -nummer,
- **Umsatzsteuer-ID:** falls beantragt und zugeteilt,
- **Wirtschafts-ID:** sobald zugeteilt,
- **Reglementierte Berufe:** zusätzliche Regelungen für freie beziehungsweise reglementierte Berufe.

Oft sind im Rahmen des Impressums noch weitere Angaben zu finden, wie etwa Urhebernachweise für Werke Dritter.

Datenschutz

Bei Profilen in den sozialen Medien oder anderen Plattformen muss man sich auch als nicht mehr nur rein privat Agierender wenige Gedanken zum Thema Datenschutz machen. Die zentral vom Portalbetreiber bereitgestellte Datenschutzerklärung ist ausreichend.

Allerdings haben alle Betreiber eines nicht nur rein privaten Internetauftritts seit Inkrafttreten des IT-Sicherheitsgesetzes Mitte 2015 zusätzliche Pflichten. Spätestens seit diesem Zeitpunkt müssen die Abläufe und Prozesse im eigenen Unternehmen, auch wenn dieses nur aus einer Person besteht, datenschutzkonform sein. Insbesondere gilt es, die eigenen technisch-organisatorischen Maßnahmen gemäß Anlage zu § 9 des Bundesdatenschutzgesetzes entsprechend den Vorgaben des Gesetzgebers anzupassen.

Werbung

Eingeblendete Werbung, ganz gleich ob Werbetexte oder Bannergrafik, muss als solche gekennzeichnet werden. Redaktionelle Inhalte sind von werblichen Inhalten eindeutig zu trennen. Der deutlich platzierte und gut lesbare Begriff »Anzeige« bewerkstelligt genau dies. ■

Michael Rohrlich
ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.
www.rechtssicher.info

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Trends am Arbeitsmarkt

Rosige Aussichten

Die IT-Unternehmen blicken optimistisch ins Jahr 2016, und das sind gute Aussichten für den Arbeitsmarkt. Fast drei Viertel der Unternehmen erwarten für das erste Halbjahr steigende Umsätze, nur acht Prozent rechnen mit rückläufigen Geschäften, berichtet der Branchenverband Bitkom. Insbesondere Software-Anbieter und IT-Dienstleister sind positiv gestimmt (Bild 1).

Sechs von zehn Unternehmen planen, im laufenden Jahr zusätzliche Stellen zu schaffen, nur acht Prozent gehen von einem Personalabbau aus. »In den vergangenen fünf Jahren haben die ITK-Unternehmen rund 135.000 neue Arbeitsplätze geschaffen. Zum Jahreswechsel gab es in der Bitkom-Branche erstmals mehr als eine Million Beschäftige«, so Bitkom-Chef Dr. Bernhard Rohleder. Dabei wird es für die Unternehmen immer schwieriger, geeignete Bewerber zu finden. 61 Prozent aller Unternehmen geben an, dass die Fachkräftesituation ihre Geschäfte behindert; vor einem Jahr betrug der Anteil noch 52 Prozent.

Jobs für Webentwickler

Die meisten Angebote für Webentwickler gab es Ende Januar in den Großstädten München, Berlin und Hamburg, wobei auffällt, dass für München mit 27,2 Prozent fast so viele Ausschreibungen in der Datenbank von Jobkralle.de zu finden waren, wie für Berlin und Hamburg zusammen. Mit Blick auf die Bundesländer lagen Bayern, NRW und Baden-Württemberg auf den ersten drei Rängen. Für die 15 größten Städte in Deutschland wurden knapp über 70 Prozent der Webentwickler-Stellen ausgeschrieben, nur 29,8 Prozent der Anzeigen suchten Mitarbeiter für kleinere Städte beziehungsweise ländliche Gebiete. Die geringste Nachfrage nach Webentwicklern war Ende Januar für das Saarland und Brandenburg zu verzeichnen.

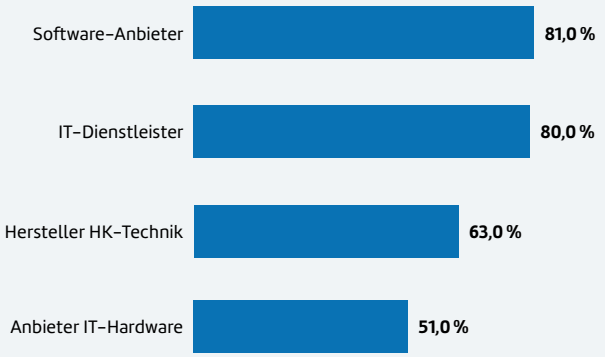
Entwickler von Apps für Mobilgeräte fanden Anfang Januar die meisten Jobangebote in den Bundesländern Bayern, NRW und Baden-Württemberg. Auch bei diesen Jobs entfallen über 70 Prozent auf die 15 größten deutschen Städte (Bild 2). Lediglich 28,5 Prozent der Angebote sind für kleinere Städte ausgeschrieben. Während München, Hamburg und Berlin das Städte-Ranking anführen – zusammen waren für diese drei Städte 56,3 Prozent der Großstadt-Angebote ausgeschrieben –, liegen Bremen, Essen und Duisburg am Ende des Rankings. In Tabelle 1 finden Sie die aktuelle Nachfrage nach ausgewählten Technologien. Ende Januar waren nur kleine Veränderungen zu verzeichnen. So überholten die Nennungen von iOS in Stellenanzeigen diejenigen von Microsoft SQL Server. Zugleich wurde häufiger nach SharePoint-Kenntnissen gefragt als nach HTML5.

Tabelle 1

Rang	Technologie	Anteil *
1	Cloud	14,9 %
2	MySQL	12,9 %
3	SharePoint	12,6 %
4	HTML5	10,3 %
5	Android	6,3 %
6	Big Data	6,1 %
7	iOS	5,6 %
8	Microsoft SQL Server	5,6 %
9	Responsive Web	5,6 %
10	CSS3	4,9 %
11	Angular.js	4,5 %
12	Windows 10	3,1 %
13	WPF	2,6 %
14	NoSQL	2,3 %
15	WCF	1,5 %
16	Azure	1,2 %

* Prozentualer Anteil der Treffer

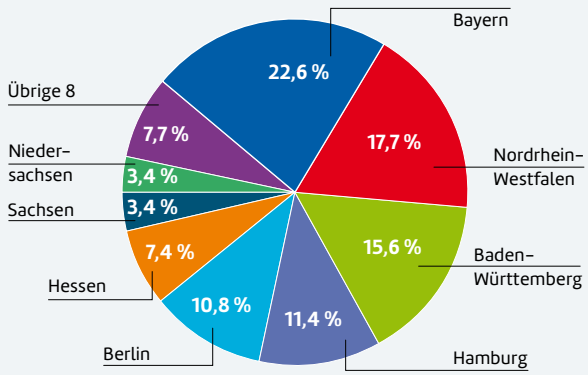
Steigende Umsätze erwartet



Acht von zehn Software-Anbietern erwarten im ersten Halbjahr 2016 steigende Umsätze (Bild 1)

web & mobile developer 4/2016 Quelle: Bitkom.org

Jobs für Mobile-Entwickler



55,9 Prozent der Angebote für Mobile-Entwickler suchen Mitarbeiter für Bayern, NRW und BW (Bild 2)

web & mobile developer 4/2016 Quelle: Jobkralle.de

Zahl des Monats

Für das Jahr 2016 erwarten **81 %** aller IT- und TK-Unternehmen ein Umsatzplus, gerade einmal **5 %** gehen von einem Minus aus. Diese positiven Erwartungen werden sich wie in den Vorjahren auch auf den Arbeitsmarkt auswirken.

Quelle: Bitkom

Textio.com

Die perfekte Stellenanzeige

Für den Text englischsprachiger Stellenanzeigen kann der Service des Start-ups Textio.com eine große Hilfe sein. Der Dienst analysiert den Text der Anzeige auf die optimale Länge, die Nutzung aktiver und positiver Sprache, checkt den Text auf starke Verben, merkt an ob Frauen ebenso wie Männer angesprochen werden, und meckert gegebenenfalls über ausufernde Aufzählungen. Die schlaue Maschine generiert im Hintergrund Vorschläge auf Basis von sehr vielen Auswertungen von Stellenanzeigen. Dabei wird für jeden Text eine Punktezahl zwischen 1 und 100 generiert. Und so lassen sich die eigenen Anzeigen im Vergleich zum Wettbewerb einordnen. Die perfekte Stellenanzeige käme auf 100 Punkte. Eine nahezu perfekte Ausschreibung mit 98 Punkten kann man bei Textio einsehen.

LinkedIn

Nachgefragte Fähigkeiten

Um herauszufinden, welche Fähigkeiten von den Arbeitgebern besonders gefragt sind, hat LinkedIn alle auf der Plattform laufenden Rekrutierungs- und Einstellungsprozesse unter die Lupe genommen und die am meisten nachgefragten Kenntnisse zusammengestellt. Dabei wurde nicht speziell nach IT-Kenntnissen gesucht, dennoch werden auffällig viele technische Fertigkeiten nachgefragt. Bild 3 zeigt die Top 10 der von LinkedIn für Deutschland gelisteten Skills. An der Spitze des Rankings stehen Kenntnisse im Cloud- und Distributed Computing. Unter den Top-25-Skills finden sich zudem die folgenden zehn weiteren IT-Kenntnisse: Software Revision/Steuerungssystem (12), Web-Architektur und -Entwicklung (13), User Interface Design (16), Software-Modellierung und Prozess-Design (17), Java-Entwicklung (18), Perl/Python/Ruby (20), Shell Scripting (22), Datenaufbereitung (23), Mac, Linux und Unix Systeme (24) und Business Intelligence (25).

CA Technologies

Initiative Deploy your Talents

Die Initiative Deploy your Talents möchte dazu beitragen, die Lücke zwischen dem Bedarf an neuen Arbeitskräften in den MINT-Fächern (Mathematik, Informatik, Naturwissenschaften und Technik) und dem Interesse von jungen Leuten an diesen Themen zu schließen. Dazu besuchen CA-Mitarbeiter Schulen und laden die Schüler in die Büros des Unternehmens ein. Die Mitarbeiter sprechen über ihre eigenen Karrieren und leiten Workshops, in denen die Schüler mehr über Berufsprofile, die steigende Bedeutung von Technologie, neue Jobs, die durch technologische Entwicklungen entstehen, sowie die Zukunft der Arbeit erfahren.

Solcom

Rechtsunsicherheit bei Freiberuflern

Eine überwiegende Mehrheit der Freiberufler schätzt die eigene Kompetenz in Rechtsfragen als mindestens gut ein. Um auf dem neuesten Stand bei Änderungen der rechtlichen Gegebenheiten zu bleiben, informieren sich Freiberufler im Wesentlichen über Webseiten oder bei Dienstleistern. Dabei war schon jeder Fünfte mindestens einmal während seiner Tätigkeit als Freiberufler in eine Rechtsstreitigkeit verwickelt.

Knapp die Hälfte der Umfrageteilnehmer nimmt zumindest gelegentlich die juristische Unterstützung eines Anwalts bei Rechtsfragen in Anspruch, jeder Fünfte sogar immer. Dass dies notwendig ist, zeigt sich beim nächsten Ergebnis: Fast zwei Drittel der Umfrageteilnehmer empfindet die Rechtssicherheit als Freiberufler zumindest als unsicher, nur jeder Zehnte als sicher (Bild 4).

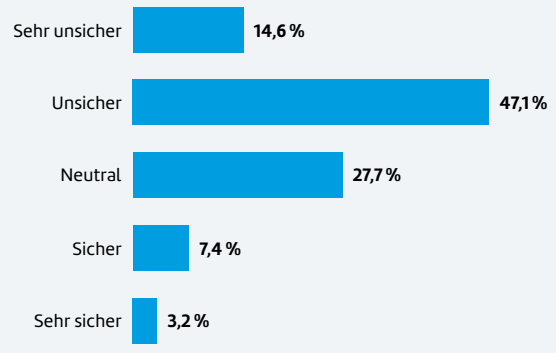
Top 10 Kenntnisse

Rang	Kenntnis
1	**Cloud und Distributed Computing**
2	Elektrotechnik
3	SEO/SEM Marketing
4	**Statistische Analyse und Data Mining**
5	**Software-Qualitätssicherung und Anwendertests**
6	**Entwicklung im Bereich mobile Geräte**
7	**Database Management und Software**
8	Channel Marketing
9	**Data Engineering und Data Warehousing**
10	Logistik im Einzelhandel

Sechs von zehn der gefragtesten Fertigkeiten sind Kenntnisse, die in IT-Berufen anzutreffen sind (Bild 3)

web & mobile developer 4/2016 Quelle: LinkedIn

Rechtssicherheit als Freiberufler

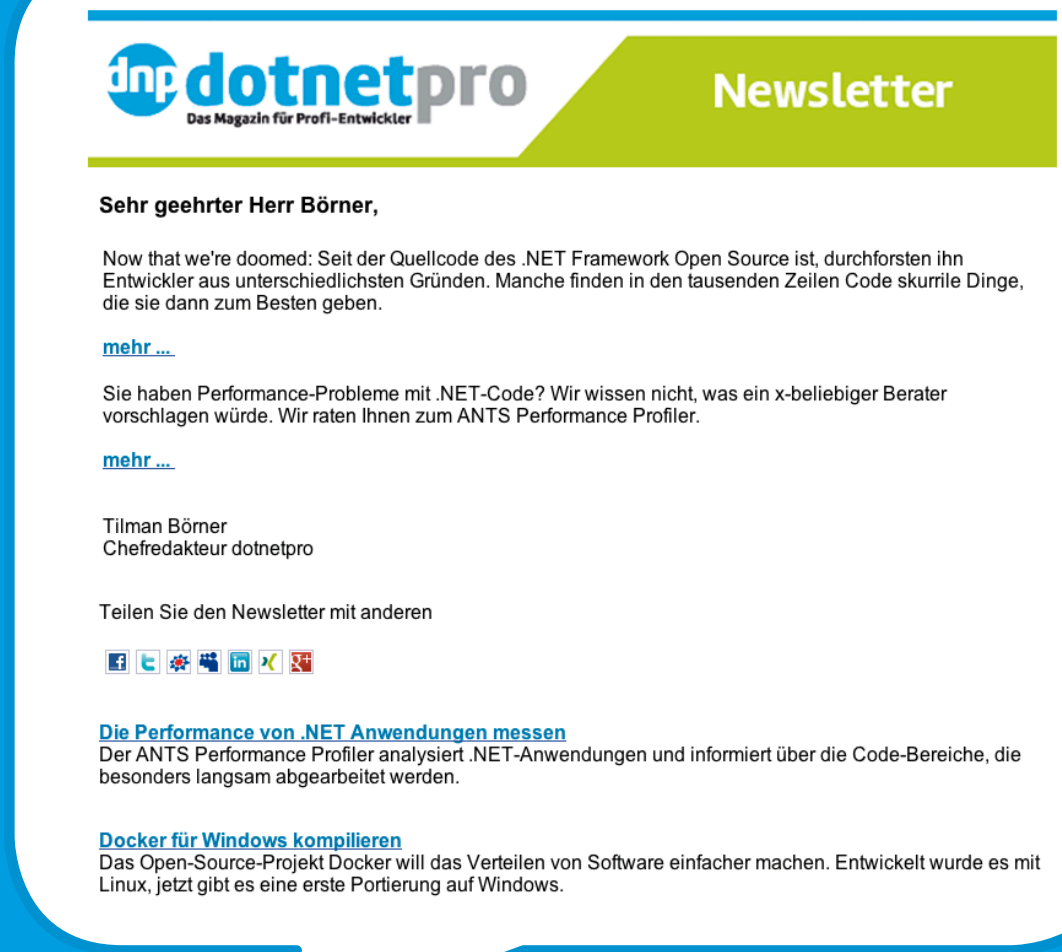


Deutlich mehr als die Hälfte der Freiberufler empfinden ihren rechtlichen Status als unsicher oder sehr unsicher (Bild 4)

web & mobile developer 4/2016 Quelle: Solcom

dotnetpro Newsletter

Top-Informationen für den .NET-Entwickler.
Klicken. Lesen. Mitreden.


dotnetpro
Das Magazin für Profi-Entwickler
Newsletter

Sehr geehrter Herr Börner,

Now that we're doomed: Seit der Quellcode des .NET Framework Open Source ist, durchforsten ihn Entwickler aus unterschiedlichsten Gründen. Manche finden in den tausenden Zeilen Code skurrile Dinge, die sie dann zum Besten geben.

mehr ...

Sie haben Performance-Probleme mit .NET-Code? Wir wissen nicht, was ein x-beliebiger Berater vorschlagen würde. Wir raten Ihnen zum ANTS Performance Profiler.

mehr ...

Tilman Börner
Chefredakteur dotnetpro

Teilen Sie den Newsletter mit anderen

Die Performance von .NET Anwendungen messen
Der ANTS Performance Profiler analysiert .NET-Anwendungen und informiert über die Code-Bereiche, die besonders langsam abgearbeitet werden.

Docker für Windows kompilieren
Das Open-Source-Projekt Docker will das Verteilen von Software einfacher machen. Entwickelt wurde es mit Linux, jetzt gibt es eine erste Portierung auf Windows.

Jetzt kostenlos anmelden:

dotnetpro.de

twitter.com/dotnetpro_mag

facebook.de/dotnetpro

gplus.to/dotnetpro

Anbieterverzeichnis

für Deutschland, Schweiz und Österreich.

Consulting / Dienstleister

ANEXIA Internetdienst-leistungs GmbH
Feldkirchner Straße 140
9020 Klagenfurt / AUSTRIA
T +43-50-556
F +43-50-556-500
info@anexia-it.com

ANEXIA wurde im Juni 2006 von Alexander Windbichler als klassischer Internet Service Provider gegründet. In den letzten Jahren hat sich ANEXIA zu einem stabilen, erfolgreichen und international tätigen Unternehmen entwickelt, das namhafte Kunden rund um den Globus mit Standorten in Wien, Klagenfurt, München, Köln und New York City betreut. ANEXIA bietet ihren Kunden hochwertige und individuelle Lösungen im Bereich Web- und Managed Hosting, sowie Individualsoftware und App Entwicklung.

---

prodot GmbH
Schifferstraße 196
47059 Duisburg
T: 0203 – 346945 – 0
F: 0203 – 346945 – 20
info@prodot.de
https://prodot.de

Intelligente Software für internationale Konzerne und mittelständische Unternehmen: prodot stärkt Kunden im weltweiten Wettbewerb – mit effizienten, stabilen und kostensenkenden Lösungen.
Durch das Zusammenspiel aus Know-how, Kreativität und Qualitätsmanagement leisten wir einen Beitrag zum langfristigen Erfolg unserer Auftraggeber.
Seit über 15 Jahren vertrauen uns deshalb Marktführer wie Aldi Süd, Microsoft und Siemens.
prodot – People. Passion. Performance..

eCommerce / Payment

Payone GmbH & Co. KG
Fraunhoferstraße 2-4
24118 Kiel
T: +49 431 25968-400
F: +49 431 25968-1400
sales@payone.de
www.payone.de

PAYONE ist einer der führenden Payment Service Provider und bietet modulare Lösungen zur ganzheitlichen Abwicklung aller Zahlungsprozesse im E-Commerce. Das Leistungsspektrum umfasst die Zahlungsabwicklung von allen relevanten Zahlarten mit integriertem Risikomanagement zur Minimierung von Zahlungsausfällen und Betrug. Standardisierte Schnittstellen und SDKs erlauben eine einfache Integration in bestehende IT- und mobile Systemumgebungen. Über Extensions können auch E-Commerce-Systeme wie Magento, OXID eSales, Demandware, Shopware, plentymarkets und viele weitere unkompliziert angebunden werden.

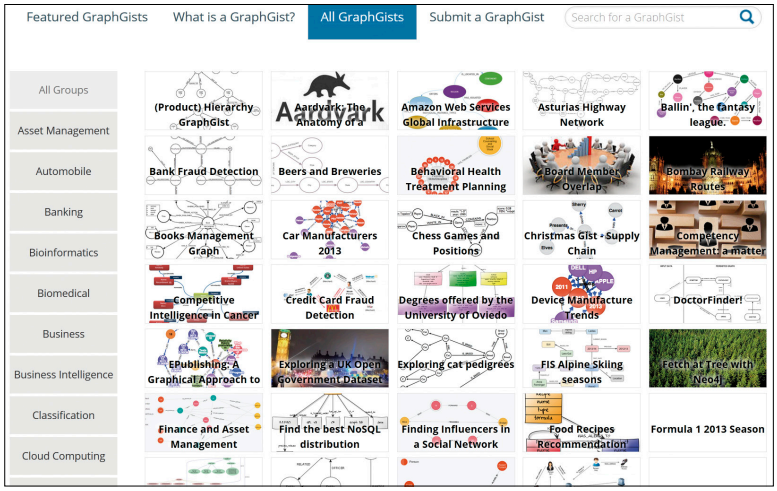
Web- / Mobile-Entwicklung & Content Management

digitalmobil GmbH & Co. KG
Bayerstraße 16a, 80335 München, T: +49 (0) 89 7 41 17 760, info@digitalmobil.com, www.digitalmobil.com

In allen Fragen rund um das Dienstleisterverzeichnis berät Sie Frau Roschke gerne persönlich!
Juliane Roschke ■ 089 / 7 41 17 – 283 ■ juliane.roschke@nmg.de

Die Ausgabe 5/2016 erscheint am 14. April 2016

Graph-Datenbanken mit Neo4j



Mit relationalen Datenbanken stößt man bei der Verarbeitung von Graphen mit vielen Beziehungen schnell an Grenzen, da diese in ihrem Datenmodell den Fokus vollkommen auf Attributwerte, also Eigenschaften, und nicht auf die modellierte Struktur (den Graphen) legen. Für die Speicherung von Graphen benötigt man ein Datenbanksystem, das besonderen Wert auf die Beziehungen zwischen den beteiligten Entitäten legt – wie es bei einer Graph-Datenbank der Fall ist. Sie stellt die Beziehungen zwischen den Entitäten in den Fokus ihres Datenmodells. Im Unterschied zur relationalen Datenbank, bei der sich die Verknüpfungen erst implizit durch Interpretation der Attributwerte ergeben, sind sie bei einer Graph-Datenbank expliziter, das heißt, bereits inhärenter Bestandteil der Daten.

Scripting auf Anwendungsebene

Zeitgemäße Betriebssysteme und Applikationen im Desktopbereich haben eine grafische Bedienoberfläche. Normalbenutzer erwartet ein solches interaktives GUI von lokal installierten Anwendungen und Webseiten. Im Serverbereich und auch bei Entwicklungswerkzeugen verschiebt sich das Bild. Das primäre Interesse von Admins, Entwickler oder Softwaretestern liegt in einer möglichst effizienten Erledigung der jeweiligen Arbeitsaufgabe.

Das Constraint Validation API

Noch vor einigen Jahren war die Validierung von Formularfeldern nur mit JavaScript möglich. Mittlerweile bietet HTML schon von Haus aus gewisse Möglichkeiten zur Überprüfung von Formularfeldern an. Allerdings ist die Validierung ohne JavaScript nicht sonderlich flexibel. Doch es gibt Abhilfe: Dank des sogenannten Constraint Validation API lässt sich die native Validierung von HTML5 nämlich auch per JavaScript steuern.

Sandbox-Umgebung Tonic

Bei Tonic handelt sich um eine Sandbox-Umgebung für JavaScript/Node.js im Webbrowser. Visuell ist Tonic sehr ähnlich wie Mathematica-Arbeitsblätter und sehr interaktiv gestaltet. Der Dienst ist (zumindest bislang) kostenlos und bietet eine ganze Reihe sehr interessanter Features. So kann man beispielsweise beliebige Node.js-Pakete einfach mittels *require()* einbinden. Ebenso kann man in eigenen Arbeitsblättern HTTP-Endpunkte definieren.

dotnetpro

Foto: shutterstock / studiostoks

Ausgabe 4/2016 ab 17. März am Kiosk

Der Schwerpunkt der dotnetpro 4/2016 befasst sich mit Continuous Integration. CI ist ein zentraler Baustein moderner und vor allen Dingen agiler Software-Entwicklung. Der Schwerpunkt zeigt, wie Sie zur CI-Pipeline kommen.

www.dotnetpro.de

Unsere digitalen Angebote

Wöchentlicher Newsletter
www.webundmobile.de/newsletter-1022034.html

Shop
https://shop.webundmobile.de

YouTube
www.youtube.com/user/developermedia

Facebook
www.facebook.com/webundmobile

Google +
gplus.to/webundmobile

Twitter
twitter.com/webundmobile

Stellenmarkt

dotnetpro + web & mobile Developer

- 25.800 Exemplare Gesamtauflage
- 25.300 Newsletter-Empfänger
- 66.600 PI'S

.NET Architektur HTML5/JavaScript iOS/Android

Kontakt:
Jens Schmidtmann, Klaus Ahlering • Tel. 089/74117-125 • sales@nmg.de

Ihr Partner für mehr Online-Wachstum

Wir planen, entwickeln und steuern
Websites, **Apps** und **Kampagnen**.

Unsere Ziele sind Ihre Ziele:

- **Mehr Sichtbarkeit**
- **Mehr Traffic**
- **Mehr Leads**
- **Mehr Conversions**

- **Mehr Kunden**

Besuchen Sie uns unter
www.digitalmobil.com

digitalmobil