

**Responsive E-Mails
entwerfen** S. 44

**QR-Codes mit Swift
lesen und erzeugen** S. 76

**Facebook API mit
PHP ansprechen** S. 112

web & mobile
DEVELOPER

web & mobile

webundmobile.de

DEVELOPER

Bezahlungssysteme integrieren

So steigern Sie den
kommerziellen
Erfolg Ihres
Online-Shops S. 14



INFO-
Programm
gemäß
§ 14
JuSchG



Auf der Heft-CD

Eine Sammlung interessanter JavaScript-Frameworks sowie
eine Auswahl leistungsfähiger Tools für Entwickler S. 50

Ausgabe 2/16

**Deutschland
14,95 EUR**

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





Upgrades für Ihr Entwickler-Know-How

Ab sofort in unseren Shops erhältlich!



<https://shop.dotnetpro.de>
<https://shop.webundmobile.de>



Bezahlungssysteme

Der wirtschaftliche Erfolg eines Online-Shops hängt stark von den angebotenen Payment-Systemen ab.

Der kommerzielle Erfolg eines Webshops mag von einer Vielzahl verschiedener Faktoren abhängen, doch keine einzige Bestimmungsgröße wirkt sich derart nachhaltig auf die Konversionsrate aus wie der Ablauf des Zahlungsvorgangs aus Sicht des potenziellen Käufers. Wer maßgeschneiderte E-Commerce-Anwendungen nach individuellen Kundenwünschen entwickeln möchte, ist gut beraten, nicht bloß Hals über Kopf in die Fußstapfen der Pioniere zu treten, sondern von den erfolgreichsten Webshops zu lernen, wie sich mit einem schmerzlosen Bezahlvorgang die Loyalität der Käufer gewinnen lässt. Was dabei zu beachten ist, zeigt unser Schwerpunktthema ab Seite 14.

»Beim Codieren von HTML-Mails gelten andere Regeln als bei Webseiten.«

Bei der Codierung von HTML-Mails gelten andere Regeln als bei Webseiten. Doch auch hier empfiehlt sich responsives Design. Was beim Entwurf und bei der Realisierung von responsiven HTML-Mails zu beachten ist, zeigt ein Artikel von Dr. Florence Maurice ab Seite 44.

QR-Codes sind im Internet und im realen Leben omnipräsent. In einem Artikel ab Seite 76 zeigt Christian Bleske, wie QR-Codes sowohl erzeugt als auch gelesen werden können. Vor iOS 7 war die Verarbeitung von QR-Codes ausschließlich mittels externer Bibliotheken möglich. Seit Version 7 gehören Klassen, die das Lesen und Erzeugen von QR-Codes ermöglichen, zu den Bordmitteln von iOS.

Facebook bietet für PHP ein leistungsfähiges SDK an, über das man zahlreiche Facebook-Features direkt ansprechen kann. Patrick Lobacher zeigt in einem Artikel ab Seite 112, was alles möglich ist.

Ihr Max Bold
chefredakteur@maxbold.de



Sebastian Eschweiler

stellt die neuen Features von Meteor 1.2 vor (S. 32)



Thomas Sillmann

erläutert die Speicherverwaltung mit ARC in Swift (S. 70)



Jens Geyer

stellt die In-Memory-Datenbank Redis vor (S. 90)



INHALT

Aktuell

Apple

Die Programmiersprache Swift wird Open Source

6

Feature

Zahlungssysteme in E-Commerce-Anwendungen integrieren

Der erste Goldrausch des E-Commerce mag vorbei sein. Dennoch sind sich Experten einig: Dank neuartiger Zahlungssysteme steht die Internetwirtschaft vor einer neuen Revolution

14

HTML, CSS & JavaScript

Meteor 1.2 und React

Das JavaScript-Framework Meteor 1.2 umfasst den gesamten Anwendungs-Stack

32

Responsive E-Mails

Responsive HTML-Mails sind wegen der unterschiedlichsten Zielplattformen ein komplexes Thema. Der Artikel zeigt, wie man solche Mails entwirft und realisiert

44

Regressionstests unter Node.js mit WebDriverIO

Regressionstests dienen in der Software-Entwicklung dazu, sicherzustellen, dass Änderungen an einer Software keine neuen Fehler für bestehende Komponenten verursachen

52

Web Animations API

Über das Web Animations API lassen sich Animationen definieren und steuern

58

Dependency Injection und Unit-Testing mit Angular 2.0

DI und gute Testbarkeit waren schon immer ein Alleinstellungsmerkmal für AngularJS

64

Mobile Development

Speicherverwaltung mit ARC in Swift

Auch wenn die Speicherverwaltung mittels ARC vieles automatisch regelt, gibt es einige Dinge, auf die Entwickler beim Programmieren mit Swift achten müssen

70

QR-Codes mit Swift

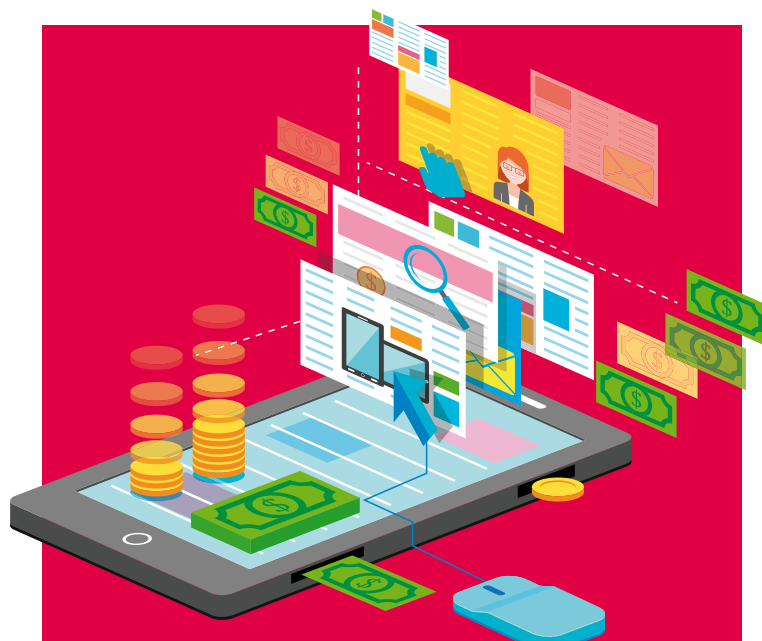
Vor iOS 7 war die Generierung von QR-Codes unter iOS eine Qual. Seit Version 7 gibt es eine eigene Bibliothek dafür

76

Cross-Plattform-Entwicklung mit Xamarin

Mit Xamarin steht eine Laufzeitumgebung zur Verfügung, die .NET-Code unter Android und iOS ausführen kann

80



Zahlungssysteme sind entscheidend für den kommerziellen Erfolg von E-Commerce-Applikationen

S. 14

Foto: Mathias Vietmeier



Responsiv designte HTML-Mails sind machbar und auch empfehlenswert

S. 44

Experten in dieser Ausgabe



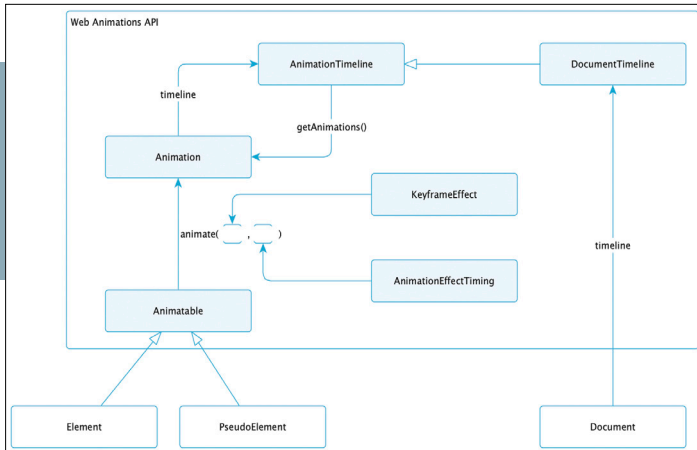
Patrick Lobacher erläutert, wie man aus PHP heraus auf die Funktionen von Facebook zugreifen kann.

112



Katharina Sckommodau zeigt, wie man den Schärfeindruck bei Bildern durch Nachbearbeitung optimiert.

124



Web Animations API: Bei der Unterstützung sind die Browser Chrome, Firefox und Opera vorne mit dabei **S. 58**



Bildschärfe: Physikalische Bildschärfe wird mit Hilfe von Linien pro Millimeter gemessen, die noch voneinander unterschieden werden können **S. 124**



Mitarbeiter in Unternehmen nutzen immer häufiger Mobile Devices für ihre Arbeit **S. 138**

Foto: Samsung

Jetzt abonnieren

Sichern Sie sich jetzt die **web & mobile developer** im Jahresabo und profitieren Sie von exklusiven Services und Angeboten für Abonnenten.
<https://shop.webundmobile.de>

Backend

In-Memory-Datenbank Redis

Redis ist nicht nur ein einfacher RAM-basierter Key-Value-Store, sondern ein hervorragend skalierendes Multitalent für viele Einsatzfälle **90**

Professionelle Datenanalyse mit BI-Tools

In dem Artikel werden die besten Werkzeuge zur Verarbeitung und Visualisierung von Unternehmensdaten unter die Lupe genommen und bewertet **104**

Facebook API mit PHP ansprechen

Facebook bietet für PHP ein leistungsfähiges SDK an, über das man Facebook-Features direkt ansprechen kann, wie zum Beispiel das Open Graph API und den Upload von Bildern **112**

PHP-Bibliothek ShellWrap

Eine kompakte Bibliothek vereinfacht die Nutzung der Linux-Shell aus PHP heraus **120**

Beyond Dev

Grafik für Entwickler: Bildschärfe

Bildschärfe entspricht nicht dem subjektiven Schärfeeindruck – dieser kann durch Nachbearbeitung erzielt werden **124**

Raspberry-Konkurrent Orange Pi

Aus China kommen mittlerweile zahlreiche Raspberry-Konkurrenten, die für weniger Geld mehr Hardwareleistung als das Original bieten **128**

Mit Mobility-Plattformen ins mobile Zeitalter

Wie die digitale Transformation die Entwicklung von Enterprise-Software verändern wird **138**

Standards

Editorial	3
Heft-CD	50
Impressum	137
Online-Recht	140
Arbeitsmarkt	142
Dienstleisterverzeichnis	145
Vorschau	146

NEWS & TRENDS

AKTUELLE NEWS AUS DER ENTWICKLERSZENE

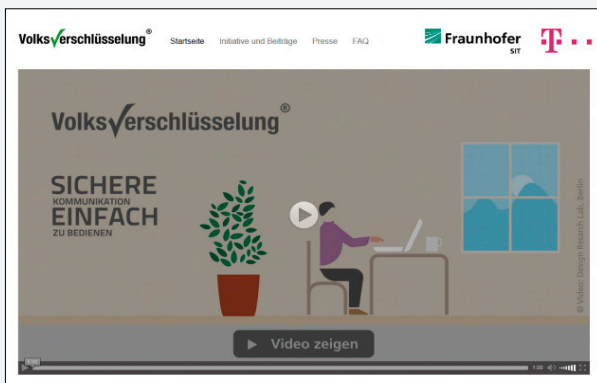
Volksverschlüsselung

Verschlüsselte E-Mails für jedermann

Die Deutsche Telekom und das Fraunhofer-Institut für Sichere Informationstechnologie SIT wollen eine einfache Möglichkeit zur Verschlüsselung von E-Mails für jedermann anbieten.

Die Volksverschlüsselung ist eine Entwicklung von Fraunhofer, die Deutsche Telekom betreibt die Lösung in einem Hochsicherheits-Rechenzentrum. Die Software ist besonders benutzerfreundlich, soll im ersten Halbjahr 2016 verfügbar sein und wird nach ihrem Start sukzessive ausgebaut und erweitert.

»Die Volksverschlüsselung ist kostenlos, unkompliziert und transparent. Für uns das beste Werkzeug, um eine Ende-zu-Ende-Verschlüsselung von E-Mails in der breiten Bevölkerung zu verankern. Mit jedem Entwicklungsschritt der Software wollen wir die Volksverschlüsselung für weitere Nutzer zugänglich machen und damit die Nutzerbasis verbreitern«, sagt Dr. Thomas Kremer, Datenschutzvorstand der Deutschen Telekom.



Die Volksverschlüsselung ist eine Entwicklung von Fraunhofer. Die Deutsche Telekom betreibt die Lösung in einem Hochsicherheits-Rechenzentrum

Prof. Michael Waidner, Leiter des Fraunhofer SIT: »Mit der Volksverschlüsselung setzen wir die Prinzipien Security by Design und Usability by Design in die Praxis um. Beide haben in jeder Phase des Entwurfs eine wichtige Rolle gespielt. Durch die Volksverschlüsselung wollen wir kryptografische Methoden, die in der Forschung etabliert sind, endlich allen Menschen zugänglich machen. Als Institut der Fraunhofer-Gesellschaft sehen wir das als Teil unseres gesellschaftlichen Auftrags an.«

Die Volksverschlüsselung ist eine Software, die sowohl die notwendigen Verschlüsselungsinformationen generiert als auch die E-Mail-Programme der Benutzer entsprechend vor-konfiguriert.

www.volksverschlueselung.de

Apple

Swift wird Open Source

Apple hat bekannt gegeben, dass seine Programmiersprache Swift jetzt Open Source ist. Die neue Ausrichtung soll dafür sorgen, dass eine große Gemeinschaft talentierter Entwickler zu neuen Funktionen und Optimierungen von Swift beitragen wird und dabei helfen soll, Swift auf neue Computerplattformen zu bringen.

Swift wurde 2014 eingeführt und ist die am schnellsten wachsende Programmiersprache der Geschichte. Sie kombiniert die Leistung und Effizienz der kompilierten Sprachen mit der Einfachheit und Interaktivität von Skriptsprachen. Apple hat außerdem die Website Swift.org eingeführt, die detaillierte Informationen über Swift als Open Source, einschließlich technischer Dokumentationen, Community-Ressourcen und Links zum Herunterladen des Swift-Quellcodes, bereitstellt.

»Indem wir Swift als Open Source anbieten, kann die gesamte Entwicklergemeinde zur Programmiersprache beitragen und dabei helfen, diese auf noch mehr Plattformen zu bringen«, sagt Craig Federighi, Senior Vice President Software Engineering von Apple. »Die Leistung und Benutzerfreundlichkeit von Swift wird eine neue Generation dazu inspirieren, mit dem Programmieren zu beginnen. Mit der heutigen Ankündigung werden sie in der Lage sein ihre Ideen überall einzubringen, von mobilen Geräten bis hin zur Cloud.«



Craig Federighi ist der Senior Vice President für Software-Entwicklung bei Apple

Der Swift Open-Source-Code ist über GitHub verfügbar und bietet Unterstützung für alle Apple-Softwareplattformen – iOS, OS X, watchOS und tvOS – und auch für Linux. Verfügbare Komponenten beinhalten Swift Compiler, Debugger, Standard Library, Foundation Libraries, Package Manager und REPL. Swift ist unter der bekannten Apache-2.0-Open-Source-Lizenz mit einer Runtime Library Exception lizenziert, was Nutzern hilft, Swift auf einfache Weise in ihre eigene Software zu integrieren und die Sprache auf neue Plattformen zu portieren.

www.swift.org

IBM Bluemix

App-Übersetzer für neun Sprachen

Mit dem neuen Cloud-Service IBM Globalization Pipeline können Entwickler ihre mobilen Apps und Anwendungen automatisch in neun Sprachen übersetzen. Verfügbar ist der cloud-basierte Service über die IBM-PaaS-Umgebung Bluemix. Die Beta-Version von IBM Globalization Pipeline basiert auf Englisch und übersetzt direkt in die Sprachen Französisch, Spanisch, Portugiesisch, Italienisch, Japanisch, Chinesisch, Koreanisch und Deutsch.

www.ibm.com

Zahl des Monats

München, Berlin, Hamburg und Frankfurt sind für Webentwickler die erste Wahl. **38%** der Angebote suchen aktuell Webentwickler für eine dieser vier Städte. Die meisten Jobs für Mobile-Entwickler gibt es derzeit in Bayern, Nordrhein-Westfalen und Baden-Württemberg. Das Einstiegsgehalt für Java-Entwickler soll in diesem Jahr **43.250 Euro** betragen.

Quelle: www.roberthalf.de/gehalt

Openers & IBM

Neue Räume für Start-ups in Berlin

Mit dem IBM Innovation Space in Berlin erhält das virtuelle Netzwerk der deutschsprachigen Gründerszene einen realen Treffpunkt. Dort können junge Start-ups zukünftig nicht nur Arbeitsplätze und Büros mieten, sondern auch vom Know-how und der Technologie von IBM profitieren. Vor Ort tauschen sich Existenzgründer mit IBM-Experten aus und entwickeln gemeinsam aus Ideen innovative Lösungen. Fundament

der IT-Infrastruktur sind die IBM SoftLayer Cloud und die PaaS-Umgebung Bluemix. Der IBM Innovation Space ist Teil der Initiative Digital City Berlin mit dem zentralen Informationsportal TechBerlin.

Der IBM Innovation Space fungiert als Anlaufstelle für den direkten Austausch mit IBM-Experten, die über den Nutzen und Einsatz aktueller IT-Trends wie Cloud, Big Data, Social oder Mobile informieren. Start-ups erhalten die Möglichkeit, neue Technologien auszu-



Der IBM Innovation Space in Berlin will der deutschsprachigen Gründerszene einen realen Treffpunkt bieten

testen und sie in ihre Anwendungen und Lösungen zu integrieren. Basis der IT-Infrastruktur sind unter anderem die IBM

SoftLayer Cloud und die Innovationsplattform Bluemix.

Darüber hinaus finden im IBM Innovation Space auch Workshops für Existenzgründer statt und es werden Veranstaltungen wie Hackathons und Meetups organisiert. Außerdem will IBM junge Existenzgründer aus der gesamten DACH-Region nach Berlin einladen, um

digitale Geschäftsmodelle zu diskutieren.

<https://developer.ibm.com/cities/locations/berlin>

MOBILE IS
EVERYTHING

#MWC16

What is mobile?

Is it the latest communications device? The health monitor on our wrist? The key to our digital security? Is it the means to connect the unconnected or is it the screen that entertains us? Mobile is all of this. But it's also so much more. Mobile powers our lives. It's an extension of who we are. Mobile is connectivity. Mobile is identity. Mobile is commerce. Mobile is inclusive. There is no clearer way to say it. Everything is mobile, but more importantly:

Mobile Is Everything. See the phenomenon for yourself in Barcelona at Mobile World Congress 2016.

 **MOBILE**TM
WORLD CONGRESS

BARCELONA 22-25 FEB 2016

WWW.MOBILEWORLDCONGRESS.COM

AN EVENT OF
 **MOBILE
WORLD CAPITAL
BARCELONA**

Cyber Security Report 2015

Risiken für Industrie 4.0

Fast 90 Prozent der Entscheider aus Politik und Wirtschaft sehen IT-Sicherheit laut dem aktuellen Cyber Security Report der Deutschen Telekom als die größte Herausforderung für die flächendeckende Umsetzung von Industrie 4.0.

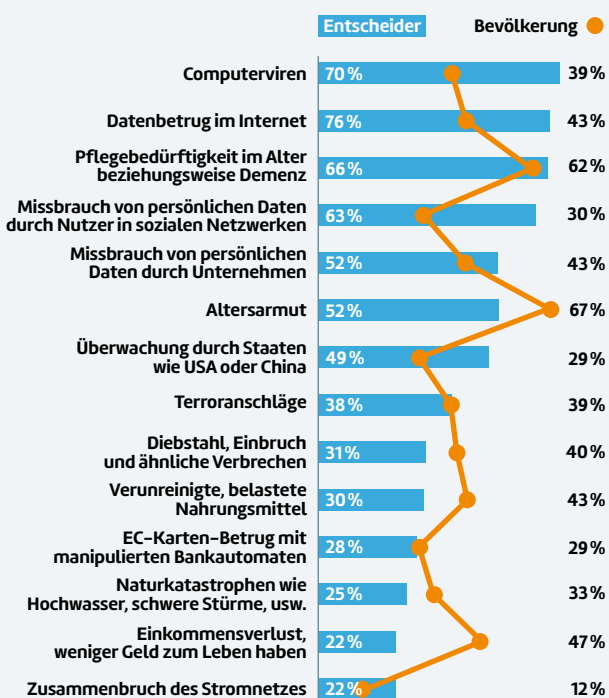
Denn die große Mehrheit der Führungskräfte (84 Prozent) ist davon überzeugt, dass mit Industrie 4.0, also der intelligenten Vernetzung von Menschen, Maschinen und Produktionsprozessen, das Risiko von Angriffen steigen wird. Weitere Industrie-4.0-Hausaufgaben sind laut den Führungskräften die flächendeckende Versorgung mit schnellem Internet sowie die Einigung auf weltweit einheitliche Standards. Der Cyber Security Report 2015 der Deutschen Telekom, erstellt vom Institut für Demoskopie Allensbach und dem Centrum für Strategie und Höhere Führung Bodman, erscheint in diesem Jahr zum vierten Mal in Folge.

War der Begriff »Industrie 4.0« im Jahr 2014 erst 38 Prozent der Unternehmen bekannt, sind es heute schon fast doppelt so viele. Nicht einmal ein Viertel der Entscheider hat sich aber intensiver mit Industrie 4.0 beschäftigt. Neun von zehn Entscheidern werten die vierte industrielle Revolution als bedeutende Entwicklung für den Wirtschaftsstandort Deutschland.

www.telekom.com

Risikowahrnehmung von Bevölkerung und Entscheidern im Vergleich

Das stellt für die Menschen in Deutschland ein großes Risiko dar



Quelle: Allensbacher Archiv, IHD-Umfragen 11040 (Juni 2015), 7231 (Sept. 2015)

Basis: Bundesrepublik Deutschland, Bevölkerung ab 16 Jahren, Abgeordnete und Führungskräfte in mittleren und großen Unternehmen.



Der aktuelle Cyber Security Report der Deutschen Telekom sieht Risiken

Nuclos

Webclient für Open-Source-ERP

Der Open-Source-ERP-Baukasten Nuclos bietet in der Version 4.6 einen Webclient. Anwender können damit unkompliziert mit einem Webbrowser auf Nuclos zugreifen – auch über mobile Endgeräte. Technologisch basiert der neue Webclient auf AngularJS.

Die Nutzer profitieren von einem sehr schnellen Zugriff auf ihre Anwendungen. Zudem können sie mit dem Webclient auch bei einer langsamen Internetverbindung effizient arbeiten. Nuclos 4.6 steht unter www.nuclos.de zum kostenlosen Download bereit.

Mit Nuclos realisieren Unternehmen sehr einfach schlanke und individuelle Softwarelösungen beispielsweise für Warenwirtschaft, Ein- und Verkauf, Auftragsabwicklung, Kundenmanagement, Lager oder Produktion. Der Open-Source-ERP-Baukasten bietet dafür effiziente Werkzeuge und generische Mechanismen. Dazu zählen Editoren für Businessobjekte zur Abbildung von Entitäten, zur Erstellung von Maskenlayouts sowie zur Konfiguration von Business-Prozessen, Workflows oder Geschäftsregeln. Administratoren steht für die Entwicklung und Konfiguration weiterhin der Java-basierte Rich Client zur Verfügung. Der Webclient hingegen bietet sich an, um Mitarbeiter im Außendienst oder Home-Office anzubinden.

www.nuclos.de



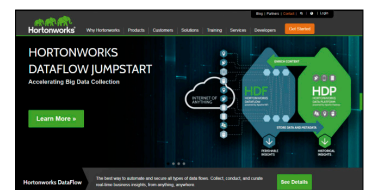
Nuclos: In der neuen Version nun mit Webclient

Hortonworks

Fünf Big-Data-Trends für 2016

In einer Welt stetig wachsender Datenmengen ist es wichtig, den Überblick über diese Daten zu behalten. Hadoop-Anbieter Hortonworks sieht für das Jahr 2016 fünf große Trends im Big-Data-Umfeld.

Jenseits von Endgeräten, Sensoren oder Maschinen kann



Hortonworks sieht für das Jahr 2016 fünf große Trends im Big-Data-Umfeld

ein Mehrwert aus allen Daten, inklusive Serverlogs, geographischer Standortbestimmung oder dem Internet selbst geschaffen werden.

Wenn Unternehmen immer mehr Daten jenseits ihrer Rechenzentren erschließen, bedarf es im entstehenden IoT-Umfeld auch neuer Konzepte der Datensicherung.

Datenplattformen werden von Data-At-Rest zu Data-In-Motion übergehen, da die Nachfrage für hocheffektive Echtzeitverarbeitung steigt.

Big-Data-Technologien müssen in der Bedienung einfacher werden, damit jeder Nutzer – vom Cluster-Operator zum Datenanalysten und darüber hinaus – sie im vollen Umfang anwenden können.

Hadoop wächst weiter und wird 2016 auch bei traditionellen Unternehmen für immer mehr kritische Prozesse verwendet werden. Man muss nicht Yahoo, Xing oder Spotify sein, um neue Analytics-Anwendungen im Bereich BI nutzen zu können.

www.hortonworks.com

Developer Newsletter



Top-Informationen für Web- und Mobile-Entwickler.
Klicken. Lesen. Mitreden.

web & mobile
DEVELOPER

Newsletter

Probleme mit der Darstellung | Aktuelles Heft

// news



Stellenbörse für Open Source-Unternehmen

Der Open Source-Branche geht es gut, und mit dem Erfolg wächst der Bedarf an weiteren Mitarbeitern. Dem trägt die Open Source Business Alliance (OSB Alliance) nun Rechnung und startet auf ihrer Website eine Stellenbörse und veröffentlicht in einem ersten Schritt Stellenangebote ihrer Mitgliedsunternehmen.



Add-on-Marktplatz für Node.js-Entwickler

Die Progress-Tochtergesellschaft Modulus hat eine Reihe von Zusatzprodukten auf ihrem Add-on-Marktplatz veröffentlicht. Damit ist es für Node.js-Entwickler einfacher, neue Funktionalitäten schneller in ihre Applikationen einzubauen.



HPI will Benchmarks für Big-Data-Leistungsvergleiche erarbeiten

Das Hasso-Plattner-Institut (HPI) ist Gastgeber des fünften internationalen Workshops zu Leistungsvergleichen im Bereich Big Data, dem so genannten Big Data Benchmarking. Das Treffen, zu dem rund 80 Teilnehmer erwartet werden, findet am 5. und 6. August am HPI in Potsdam statt.

Jetzt kostenlos anmelden:



webundmobile.de



twitter.com/webundmobile



facebook.de/webundmobile



gplus.to/webundmobile

Digitalisierung der Finanzbranche

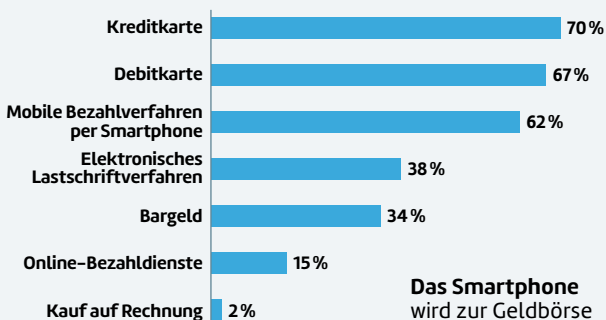
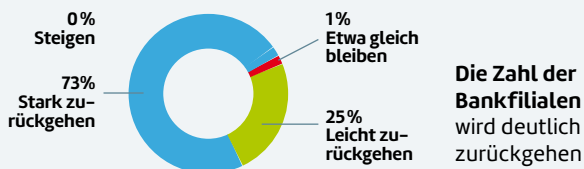
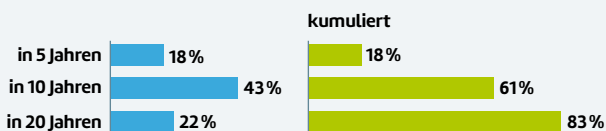
Das Smartphone als Geldbeutel und Bankfiliale

Die Digitalisierung wird die Finanzbranche in den kommenden zehn Jahren gravierend verändern. Das betrifft sowohl die Art und Weise, wie Kunden im stationären Handel bezahlen, als auch Geschäftsmodelle und Arbeitsweisen der Banken, so eine repräsentative Bitkom-Studie. Die Studie zeigt, dass die Experten vor allem dem Smartphone eine große Rolle bei den Veränderungen zusprechen. In zehn Jahren wird das Smartphone der Umfrage zufolge hinter Kreditkarte (70 Prozent) und Debitkarte (67 Prozent) das dritt wichtigste Zahlungsmittel (62 Prozent) im Handel sein. Zugleich sind 6 von 10 Befragten (61 Prozent) der Meinung, dass bis

2025 Bargeld in Deutschland nicht mehr das dominierende Zahlungsmittel sein wird.

Mehr als ein Drittel (37 Prozent) der Finanzunternehmen geht davon aus, dass es in zehn Jahren üblich sein wird, dass es im Einzelhandel keine Kassen mehr gibt. Kunden werden zum Beispiel über ihr Smartphone erkannt, nehmen die Produkte aus dem Regal und der Bezahlvorgang wird automatisch beim Verlassen des Geschäfts abgewickelt. Aber auch die Bankenwelt selbst verändert sich durch die Digitalisierung. So rechnen drei Viertel der Finanzexperten (74 Prozent) damit, dass es in zehn Jahren verbreitet sein wird, seine Bankgeschäfte mit dem Smartphone abzuwickeln.

www.bitkom.org

Welche Zahlungsmittel werden in zehn Jahren im stationären Handel am stärksten verbreitet sein?**Wie wird sich die Zahl der Bankfilialen in den kommenden zehn Jahren entwickeln?****Wann wird Bargeld in Deutschland nicht mehr das dominierende Zahlungsmittel sein?**

Eine Mehrheit sieht einen Bedeutungsverlust des Bargelds

Basis: Alle befragten Finanzunternehmen

Eurodata

E-Book zum Umgang mit Big Data

Der Software-Anbieter Eurodata geht in einem kostenlos angebotenen E-Book der Frage nach, warum Industrie 4.0 und das Internet der Dinge zu einer wachsenden Datenflut und -vielfalt führen.



Ein E-Book erläutert den richtigen Umgang mit Big Data

Anhand konkreter Beispiele wird aufgezeigt, wie Unternehmen erfolgreich mit diesen Herausforderungen umgehen können. Im Zentrum stehen dabei Softwareplattformen, die Daten über Systemgrenzen hinweg integrieren und visualisieren. So sorgen sie nicht nur für Überblick in der Datenflut, sondern tragen auch signifikant zur Verbesserung von Business-Prozessen, Transaktionen und der Zusammenarbeit bei. Im ersten Teil des E-Books werden die aktuellen Entwicklungen in der IT-Welt analysiert und dargestellt, wie die zunehmende Vernetzung physischer und virtueller Welten die Wirtschaft umwälzt und zu einer Neuausrichtung in Industrie und Handel führen wird. Strategische Aufgabe der Unternehmen muss es daher sein, sich spätestens jetzt auf weiter zunehmende, komplexe Datenmengen vorzubereiten und diese so zu nutzen, dass Prozesse und Transaktionen noch effizienter werden.

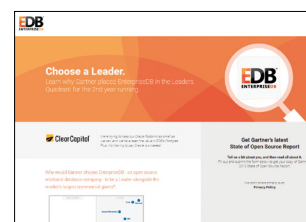
Um Informationen und Daten zu konsolidieren, die in Echtzeit, Batch-Form, strukturiert oder unstrukturiert vorliegen, wird

im E-Book die Einführung einer smarten Plattform beschrieben, auf der Daten zusammenlaufen, integriert, analysiert und so weiterverarbeitet werden, dass kontrollierte Prozesse automatisiert ablaufen können. »Wenn physische und digitale Dienstleistungen zu neuen, intelligenten Produkten und digital veredelten Geschäftsmodellen verknüpft werden, sprechen wir von Smart Services«, sagt Frank Reinelt, Geschäftsbereichsleiter der Eurodata AG. <http://ebook-smartservices-der-zukunft.eurodata.de>

EnterpriseDB

Postgres statt Oracle SE2

EnterpriseDB (EDB) bietet Oracle-SE-Nutzern für einen begrenzten Zeitraum die Möglichkeit, die EDB-Postgres-Lösung für den Betrag der jährlichen Wartungskosten, die sie bisher bei Oracle zahlen, zu nutzen. Oracle hat seine beliebte SE-Datenbank-Lizenzoption abgeschafft und durch eine neue Lizenz namens Oracle SE2 ersetzt.



EnterpriseDB (EDB) bietet Oracle-SE-Kunden eine Alternative zur Oracle-SE2-Lizenz

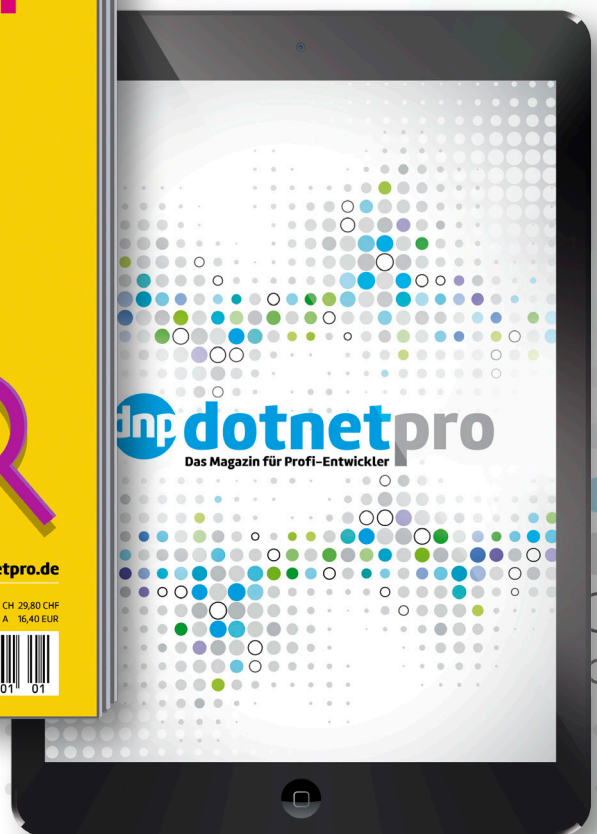
setzt. Als Folge davon sind viele bisherige SE-Nutzer gezwungen, neue Hardware anzuschaffen, Applikationen neu zu konfigurieren und sich gegebenenfalls auch mit Beeinträchtigungen bei der Performance abzufinden. Das neue Programm EDB bietet Oracle-SE-Kunden dazu eine Alternative.

www.enterprisedb.com

Jetzt kostenlos testen!



**2x
gratis!**



Das Fachmagazin für .NET-Entwickler

Testen Sie jetzt 2 kostenlose Ausgaben und erhalten Sie unseren exklusiven Newsletter gratis dazu.

www.dotnetpro.de/probeabo

AppSphere

PowerShell-Skripts per App ein- und auschecken

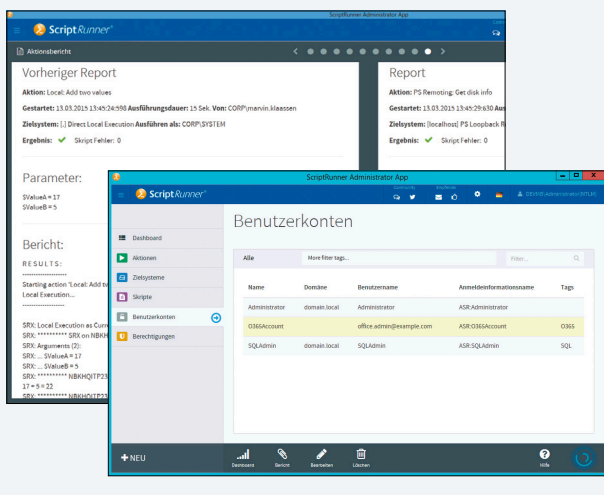
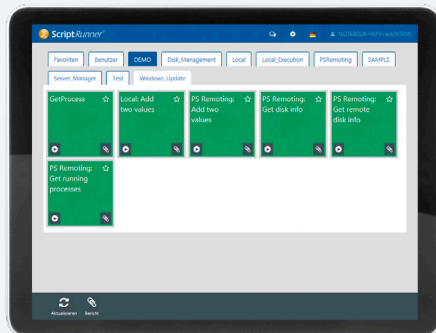
AppSphere hat sein PowerShell-Automatisierungs-Tool ScriptRunner erweitert. Die neue Version R3 der Plattform bietet nun Möglichkeiten, um Skripts im Team zu entwickeln. Entwickler und DevOps profitieren mit den Funktionen der PowerShell-ISE-App von der zentralen Verwaltung, dem Check-in/Check-Out und der Versionierung. Die neue ISE-App unterstützt den kompletten Prozessschritt zur Entwicklung von Skripts. Sie integriert sich nahtlos in die ISE und wird über die Rechte in ScriptRunner gesteuert.

DevOps oder PowerShell-Entwickler sind damit unter anderem in der Lage, für Änderungen vorgesehene Skripts zentral in der zur Verwaltung ebenfalls in ScriptRunner enthaltenen Admin-App an Dritte zur Bearbeitung freizugeben (Check-out). Dabei wird der Quellcode aus der zentralen Bibliothek geladen und in einem Tab der ISE angezeigt. Im Anschluss können Änderungen vorgenommen werden, während der Quellcode für andere User gesperrt ist. Nach Freigabe steht das Skript wieder allen Entwicklern zur weiteren Bearbeitung oder – nach dem Zurückspielen in die zentrale Bibliothek (Check-in) – zur direkten Nutzung zur Verfügung. Die Änderungen können mittels Versionshistorie jederzeit nachvollzogen werden.

Darüber hinaus ist per ISE-App auch die reine Anzeige des Skript-Quellcodes möglich, sodass Funktionen kontrolliert oder Teile des Inhalts anderweitig wiederverwendet werden können. Im Zusammenspiel mit der Admin-App kann zudem eine erweiterte Liste der Skripts dargestellt werden.

www.appsphere.com www.scriptrunner.de

AppSphere hat sein PowerShell-Automatisierungs-Tool ScriptRunner erweitert



PHP-Kräfte gebündelt

Kooperation von PHP Meetup und PHP Usergroup in Karlsruhe

Die ITstrategen GmbH, Software-Entwicklungsunternehmen und Organisator der Event-Reihe PHP Meetup Karlsruhe, und die PHP Usergroup Karlsruhe sind eine neue Kooperation



Entspannte Atmosphäre im ITstrategen-Garten

eingegangen. Regelmäßige, gemeinsam ausgerichtete Veranstaltungen sollen die bislang parallel laufenden Treffen auflösen und Entwicklern und IT-Entscheidern eine zentrale Anlaufstelle für den Wissensaustausch rund um PHP bieten.

Die Software-Entwicklung ist in der Technologieregion Karlsruhe seit jeher ein zentrales Thema. Immer mehr Business- und User-Groups sind im Lauf der Jahre entstanden, die parallel arbeiteten und um Interessenten und Veranstaltungsteilnehmer buhlten. Das Software-Entwicklungsunternehmen ITstrategen GmbH, Organisator der Event-Reihe PHP Meetup Karlsruhe, hat nun einen Schritt in die entgegengesetzte Richtung unternommen: In Kooperation mit der PHP Usergroup Karlsruhe sollen künftig gemeinsame Events ausgerichtet und der Wissensaustausch damit an zentraler Stelle gebündelt werden. Eine Auftaktveranstaltung Ende Oktober hat bereits gezeigt, dass das Konzept ankommt. Insgesamt 20 Teilnehmer äußerten sich positiv zur neuen Kooperation. Neben den Vorträgen »Multi-

Threading mit PHP« von Michael Döhler, IntelliShop AG, und »Mocks in Unit-Tests« von Claudio Zizza, Flagbit GmbH & Co. KG, wurde auch die Zeit beim Imbiss genutzt, über codebasierte Probleme und deren Lösungen zu diskutieren.

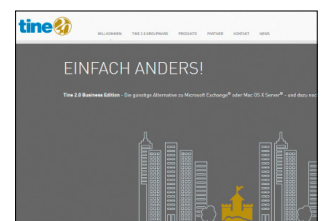
Die Weiterführung des neuen PHP Meetups Karlsruhe ist nun zunächst mit einem Event pro Quartal geplant.

www.meetup.com/de/PHP-Meetup-Karlsruhe

Open-Source-Groupware

Neue Version von Tine 2.0 vorgestellt

Metaways hat ein neues Release seiner Business Edition von Tine 2.0 veröffentlicht. Für die neue Version 2015.11 hat der IT-Dienstleister vor allem die Benutzerfreundlichkeit der beliebten Open-Source-Groupware weiter erhöht. Zu den zentralen Neuerungen von Tine 2.0 gehört die Möglichkeit, Dateien zu veröffentlichen. Die Groupware erstellt dazu Download-URLs, über die ohne Authentifizierung auf Files zugegriffen werden kann. Die Gültigkeit der



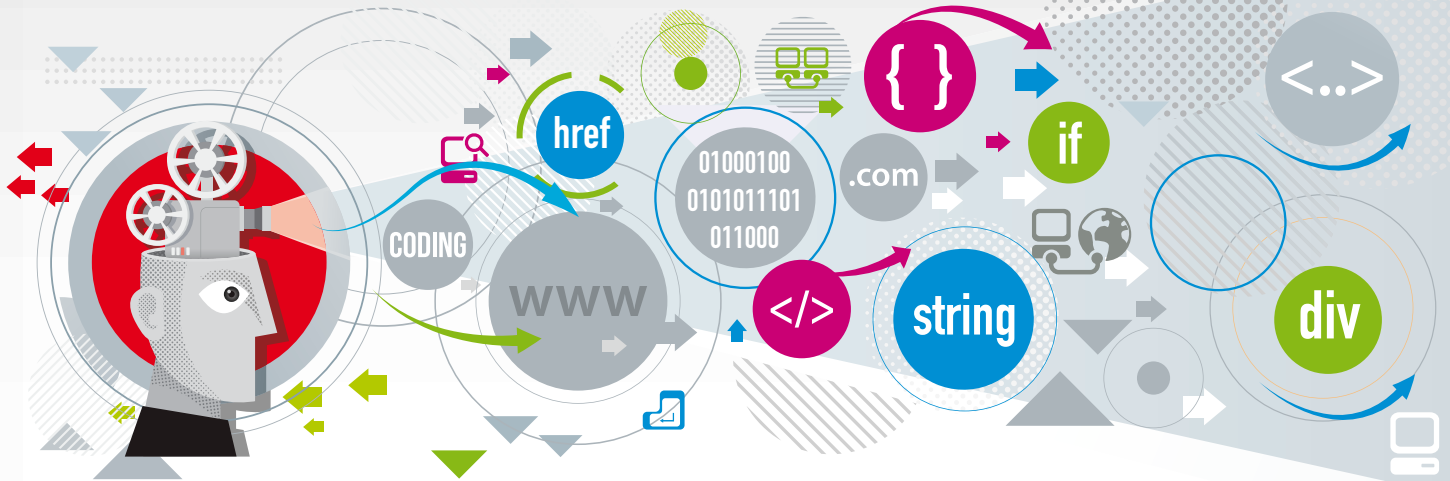
Metaways hat bei der Version von Tine 2.0 die Benutzerfreundlichkeit verbessert

URLs lässt sich zeitlich eingrenzen und die Anzahl der Downloads wird mitgezählt. Durch diese Neuerung können Anwender auch größere Dateien einfach miteinander teilen. Dropbox oder ähnliche File-Sharing-Lösungen aus der Public Cloud sind damit überflüssig. www.tine20.com



Developer Week 2016

20.-23. Juni 2016,
Messe Nürnberg



Mit
Very Early Bird
€ 200,-
sparen!
Bei Anmeldung bis
27.02.2016!

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

- **3-Tages-Konferenz**
200 Vorträge | 30 Thementracks |
Networking auf Abendveranstaltungen
- **Top-Referenten**
150 Experten aus den Themenbereichen
.NET-, Web- und Mobile-Entwicklung
- **Specials**
Maker Spaces | Community Night |
Büchercafé u. v. m.
- **Themen (u.a.)**
API | Integration | IoT | DevOps |
Industrie 4.0 | Cloud/Server
- **10 parallele Workshops am 23.06.2016**
Ganztägig | Praxisorientiert | Vertiefend |
Aktive Teilnahme
- **Fachausstellung**
Mehr als 50 Partner präsentieren sich in der
Fachausstellung vom 20.-22. Juni 2016.

developer-week.de



DeveloperWeek

Aussteller &
Sponsoren:

adorsys



brainLight®
LIFE IN BALANCE



GPU Cyrus AG

MATHEMA

SPARX
SYSTEMS

TEXTCONTROL

Veranstalter:



Neue
Mediengesellschaft
Ulm mbH

Präsentiert von:



web & mobile
DEVELOPER



Foto: shutterstock/Vectormart

ZAHLUNGSSYSTEME IN E-COMMERCE-ANWENDUNGEN INTEGRIEREN

Schmerzloser Bezahlvorgang

Neuartige Zahlungssysteme bescheren der Internetwirtschaft eine neue Revolution.

Der kommerzielle Erfolg eines Webshops mag von einer Vielzahl verschiedener Faktoren abhängen, doch keine einzige Bestimmungsgröße wirkt sich derart nachhaltig auf die Konversionsrate aus wie der Ablauf des Zahlungsvorgangs aus Sicht des potenziellen Käufers.

Dabei stellt sich heraus, dass weder Suchmaschinenoptimierung der einzelnen Produktseiten, noch A/B- beziehungsweise MV-Tests verschiedener Design-Varianten zur Maximierung der Konversionsrate, noch der Einsatz eines CDN (Content Delivery Networks) zur Minimierung der Ladezeiten einem Webshop zum Erfolg verhelfen können, wenn der Besucher das unterstützte Bezahlungssystem grundlegend ablehnt. Diese relevante Tatsache haben unabhängig voneinander der Bundesverband Digitale Wirtschaft e.V. (BVDW) und das E-Commerce-Center Handel (kurz: ECC Handel) in di-

versen Studien festgestellt. Daher sind Online-Zahlungssysteme mit einer breiten Akzeptanz für ein Unternehmen nahezu überlebenswichtig.

Wer maßgeschneiderte E-Commerce-Anwendungen nach individuellen Kundenwünschen entwickeln möchte, ist gut beraten, nicht bloß Hals über Kopf in die Fußstapfen der Pioniere zu treten, sondern von den erfolgreichsten Webshops zu lernen, wie sich mit einem schmerzlosen Bezahlvorgang die Loyalität der Käufer gewinnen lässt. Am Beispiel des Online-Riesen Amazon wird das Prinzip deutlich.

Bezahlen mit Amazon

Amazons Ein-Klick-Check-out-Prozedur gilt als der goldene Standard für eine hohe Kundenzufriedenheit. Amazons Zahlungssystem ist für die Betreiber externer Webshops mit Sitz

in Deutschland, Luxemburg, im Vereinigten Königreich, in den Niederlanden, in Schweden, Portugal, Ungarn, Dänemark und den Vereinigten Staaten als ein Dienst verfügbar. Die Anmeldung im deutschsprachigen Raum erfordert ein deutsches oder ein österreichisches Bankkonto oder eine Kreditkarte. Eine breite Akzeptanz der Amazon-Marke unter den Online-Käufern verspricht eine hohe Konversionsrate. Mit robusten APIs, einer einfachen Integration in Webapplikationen auf der Basis von Webstandards und einer leistungsstarken Sandbox zum Testen und Debuggen bietet Amazon Payments auch so ziemlich alles, was das Herz eines Webentwicklers begehrt (Bild 1).

Ansicht der Transaktionsdaten

Beim Auschecken mit Amazon Payments können Käufer auf die Lieferadressen und Zahlungsinformationen aus ihrem eigenen Amazon-Account zugreifen, um Transaktionen abzuschließen. Die Käufer können ihre Transaktionen in ihrem Amazon-Konto nicht einsehen; stattdessen müssen sie zu der Amazon Payments-Webseite unter <https://payments.amazon.com/home> navigieren. Der Shop-Betreiber verfügt über eine vollständige Ansicht der Transaktionsdaten.

Amazons Versprechen unveränderlicher APIs ist allerdings im Kontext der realen Erfahrungswerte deutlich übertrieben. Amazon hat die Dienste Simple Pay (ASP) und Flexible Payments Service (FPS) am 1. Juni 2015 eingestellt und unterstützt künftig nur noch »Login and Pay with Amazon«.

Checkout-by-Amazon-Dienste bleiben nur für bereits geöffnete Merchant-Accounts bestehen. Nach der Einführung wiederkehrender Zahlungsvorgänge im Rahmen von Login and Pay with Amazon im Mai 2014 unter dem Namen Automatic Payments hatte der Anbieter bald wieder kalte Füße

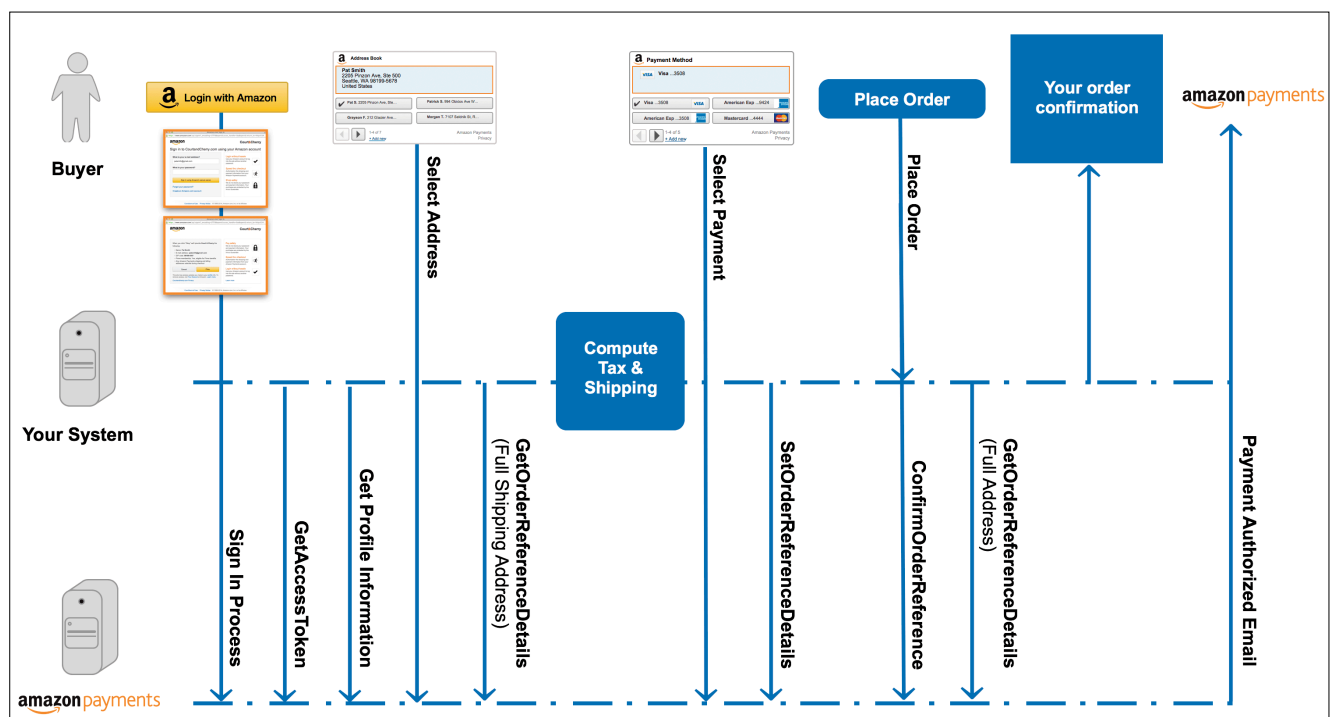
bekommen und stellte diese Funktionalität ein. Sie können Abonnements zwar nach wie vor realisieren, doch lassen sich die Buchungen nun nicht mehr via Amazons APIs nach Zeitplan auslösen. Stattdessen sind jetzt externe Lösungen vonnöten, die sich um zeitgerecht ausgeführte automatische Abbuchungen kümmern.

Die API-basierte Integration taufte Amazon auf den Namen Login und Bezahlen mit Amazon (Login and Pay with Amazon). Die Anbindung ist in zwei Ausbaustufen realisierbar:

- als Express-Integration,
- als vollständige Integration via Amazon Payments Advanced mit Unterstützung für geteilte Lieferungen und Rückstandslieferungen.

Zu den Voraussetzungen für die Umsetzung von Login und Bezahlen mit Amazon zählt eine Merchant-ID für einen Amazon Payments Seller-Account, in dem der Dienst bereits provisioniert wurde. Letzteres lässt sich unter der Adresse <https://payments.amazon.com/signup> nachholen. Es sind SDKs für PHP, C#, Java, Ruby und Python verfügbar.

Als Testumgebung zum Debuggen der API-Aufrufe bietet Amazon die sogenannte Sandbox. Mit Hilfe von Amazons Webportal für Verkäufer unter der Adresse <https://sellercentral-europe.amazon.com> können Sie Benutzer-Accounts imaginärer Käufer anlegen und nach Belieben fiktive Bestellungen aufgeben, um Zahlungsvorgänge zu testen, ohne echte Zahlungen auszulösen. Die benötigte Funktionalität findet sich im Menü *Integration, Test Accounts* der horizontalen Navigation; die gebotenen Zahlungsmethoden funktionieren in der Sandbox wie ihre Entsprechungen in der Realität. Damit Sie die Sandbox für Ihren Webshop nutzen können, müssen Sie Ihre Webapplikation in der Console registrieren. ►



Teamwork: Flowdiagramm eines Bezahlvorgangs mit dem Amazon Payments API (Bild 1)

Listing 1: Bezahlen mit Amazon

```

<head>
  // <head>-Abschnitt Ihrer Webseite
  <script type='text/javascript'>
    window.onAmazonLoginReady = function() {
      amazon.Login.setClientId('IHRE_CLIENT_ID');
    };
  </script>
  <script type='text/javascript' src='https://
static-eu.payments-amazon.com/OffAmazonPayments/
de/sandbox/lpa/js/Widgets.js'></script>
  // <head>-Abschnitt Ihrer Webseite
</head>
<div id="AmazonPayButton"></div>
  <script type="text/javascript">
    var authRequest;
    OffAmazonPayments.Button("AmazonPayButton",
    "IHRE_SELLER_ID", {
      type: "PwA",
      authorization: function() {
        loginOptions = {
          scope: "ENTER_SCOPES",
          ENTER_POPUP_PARAMETER};
          authRequest = amazon.Login.authorize
            (loginOptions,
              "IHRE_REDIRECT_URL");
        },
        onError: function(error) {
          // Fehlerhandhabung
        }
      });
  </script>
</div>

```

Um die Integration möglichst problemlos verlaufen zu lassen, sei Ihnen empfohlen, Bestellschaltflächen (**Listing 1**) und Widgets (**Listing 2**) aus der Sandbox in Ihre Webseite zu integrieren.

Solange Sie in der Sandbox arbeiten, sind die Design-Elemente mit einem Wasserzeichen gekennzeichnet. Danach können sich Beta-Tester mit den zuvor erstellten Identitäten fiktiver Käufer für Bezahlvorgänge authentifizieren. Mit Hilfe von Amazons APIs können Sie dann die gewünschten Aktivitäten simulieren und gegebenenfalls Fehler auslösen.

Nach Abschluss der Testphase können Sie von der Sandbox auf die Produktionsumgebung umschalten. Hierzu müssen Sie die Konfiguration Ihres Merchant-Accounts um alle benötigten Daten vervollständigen und alle Verweise auf die Sandbox aus Ihrem Code entfernen. Danach können Sie Ihre Lösung veröffentlichen und mit ein paar echten Testbestellungen auf die Probefahrt nehmen.

Scratchpad

Zusätzlich steht Ihnen auch noch eine Webapplikation namens Scratchpad unter der Adresse <https://sellercentral-europe.amazon.com/hz/me/apia> zur Verfügung (**Bild 2**). Mit Hilfe dieser Anwendung können Sie Aufrufe der APIs tätigen, um die Integration anhand der Response-Meldungen zu erlernen. Ihre Aktivitäten in Scratchpad reflektieren sich in Seller Central (sofern Sie den Standardbetrag nicht verändern und keine Fehlermeldung auslösen); hier können Sie die fiktiven Transaktionen ganz reell verwalten.

Für die Sandbox stellt Amazon ein MWS-Schlüsselpaar bestehend aus einem Zugangsschlüssel und einem geheimen Schlüssel direkt in Seller Central zur Verfügung (zu finden unter *Integration* und *MWS Access Key* Ihrer Sandbox).

Um API-Aufrufe in der Produktionsumgebung tätigen zu können, benötigen Sie ein echtes MWS-Schlüsselpaar aus dem Amazon Marketplace Web Service. Um das benötigte

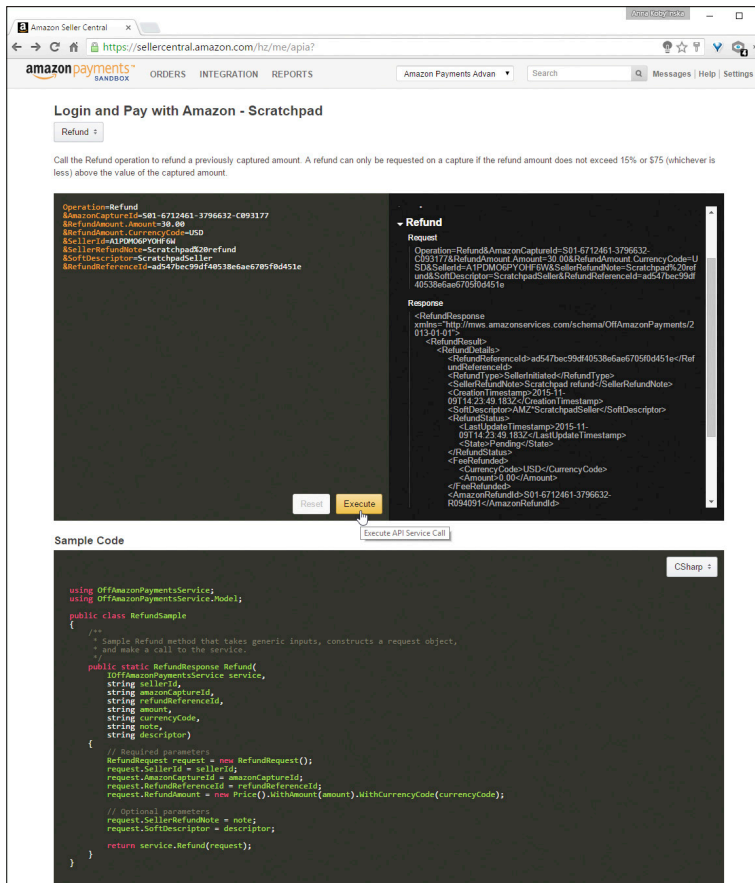
Listing 2: Das Zahlungsarten-Widget

```

<head>
  // <head>-Abschnitt Ihrer Webseite
  <script type='text/javascript'>
    window.onAmazonLoginReady = function() {
      amazon.Login.setClientId('YOUR_CLIENT_ID_HERE');
    };
  </script>
  <script type='text/javascript' src='https://
static-eu.payments-amazon.com/OffAmazonPayments/de/
sandbox/lpa/js/Widgets.js'></script>
  // <head>-Abschnitt Ihrer Webseite
</head>

<!--der folgende Code gehört in die CSS-Scriptdatei
Ihrer Webseite -->
<style type="text/css">
  #walletWidgetDiv{width: 400px; height: 228px;}
</style>
<div id="walletWidgetDiv"></div>
  <script>
    new OffAmazonPayments.Widgets.Wallet({
      sellerId: 'IHRE_SELLER_ID',
      onPaymentSelect: function(orderReference) {
        // Code zu verarbeiten nach der Auswahl
        // der Zahlungsmethode durch den Käufer
      },
      design: {
        designMode: 'responsive'
      },
      onError: function(error) {
        // Fehlerhandhabung
      }
    }).bind("walletWidgetDiv");
  </script>

```



Experimentierkasten: Per Amazon Scratchpad tätigen Sie Aufrufe der APIs, um die Integration anhand der Response-Meldungen zu erlernen (Bild 2)

Schlüsselpaar für die Produktionsumgebung zu erhalten, müssen Sie sich unter der Adresse <https://developer.amazon-services.de> als ein Amazon-Entwickler registrieren lassen.

Wählen Sie bei der Anmeldung die erste Option. Sie erhalten anschließend gleich zwei Schlüsselpaare – einmal als Verkäufer und einmal als Developer. Wenn Sie Amazon Pay-

ments in den Webshop eines Auftraggebers (statt in Ihren eigenen Webshop) integrieren, nutzen Sie das zweite der beiden Schlüsselpaare.

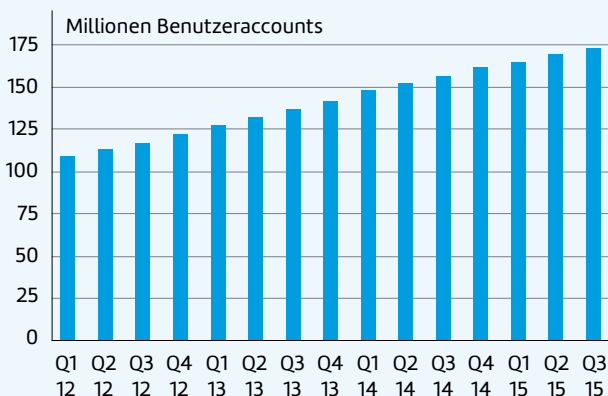
Amazons ansonsten benutzerfreundliche Lösung ist leider nicht frei von lästigen Bugs. So lehnt das Scratchpad gültigen Währungscode aus einer anderen Wirtschaftszone mit einer Fehlermeldung ab, obwohl laut der Dokumentation ein dreistelliger Währungscode gemäß ISO 4217 eigentlich funktionieren müsste. Die Widgets versagen beim Testen mit Benutzeridentitäten, die noch keine Adressen beinhalten; zwar wird die neue Adresse in Seller Central aufgenommen, aber Amazons Webanwendung dreht dabei endlos am Rad und erholt sich davon nicht wieder.

Auch lassen sich Test-Accounts aus Seller Central nicht wieder entfernen. Insgesamt macht die Lösung einen etwas unausgereiften Eindruck. Auch Städtenamen mit einem Freiraum (zum Beispiel Bad Nauheim) verhindern das Aufgeben einer Bestellung: Das Adressbuch-Widget in der Sandbox beschwert sich über die unvollständigen Angaben und deaktiviert gleich die Schaltfläche zum Fortfahren. Wer sich an diesen kleinen, wenn auch zahlreichen Ungereimtheiten nicht sonderlich stört, kann sich an Amazons geringen Transaktionskosten erfreuen.

Ein wesentlicher Nachteil von Amazon Payments besteht im strengen Verbot der Nutzung der APIs im Rahmen von nativen Apps (unter anderem für iOS, Android, RIM und Windows).

Wer diese Richtlinie mit Tricks umgehen sollte, wird aus dem Dienst ausgesperrt. An nicht nativen mobilen Apps stört sich der Online-Riese nicht, lässt die Entwickler jedoch im Unklaren darüber, welche Technologien außer reinen HTML-5-Apps sich als nicht nativ qualifizieren. Es ist somit Vorsicht bei der Anwendung geboten. ▶

Benutzeraccounts von PayPal

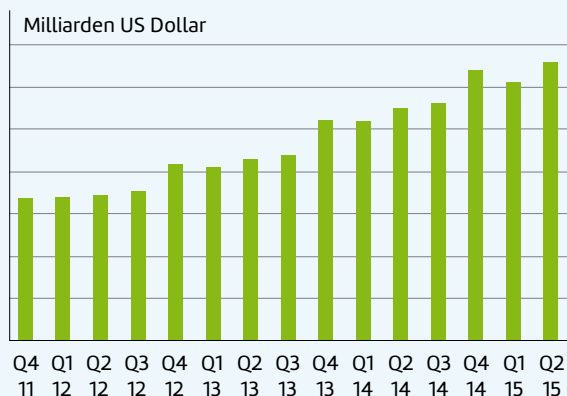


Wachstum: Die Anzahl aktiver registrierter Benutzeraccounts von PayPal (I/2010 bis II/2015, weltweit) (Bild 3)

web & mobile developer 2/2016

Quelle: Statista

Transaktionsvolumen von PayPal



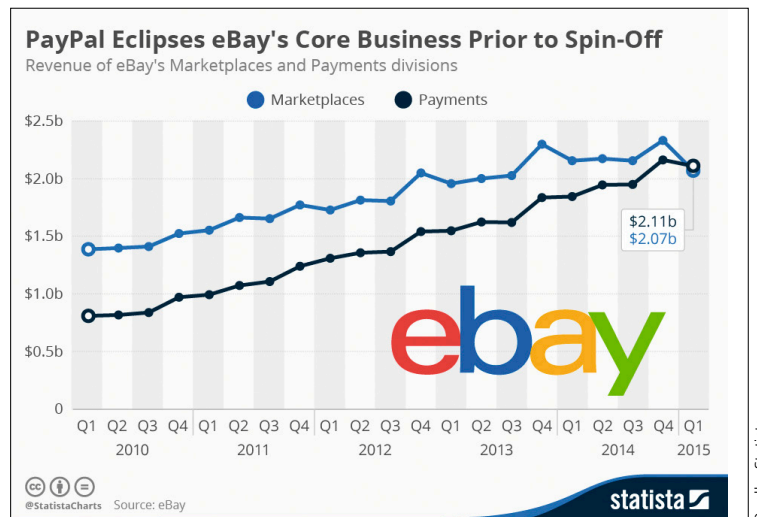
Im Trend: Entwicklung des Transaktionsvolumens von PayPal (pro Quartal, I/2010 bis II/2015) (Bild 4)

web & mobile developer 2/2016

Quelle: Statista

Mit seinen 169 Millionen aktiven Benutzern, einem Gesamttransaktionsvolumen von über 60 Milliarden Dollar pro Quartal und einer jährlichen Wachstumsquote von 22 Prozent gilt PayPal nach wie vor als der Platzhirsch des Internethandels (Bild 3, Bild 4).

In der Vergangenheit hat sich der Pionier von Internetzahlungen unbeliebt gemacht. Für viele Kunden steht das blaue Logo für rigide Beschränkungen der zulässigen Zahlungsvorgänge, unangenehm hohe Wechselkursgebühren und einen stressig-bürokratischen Kundendienst. Diese Nutzer dürften auch heute noch beim Anblick einer PayPal-Schaltfläche den Rückwärtsgang einlegen. Die Vorurteile mögen nicht immer oder nicht mehr zutreffen. Dennoch ist PayPal trotz seiner schieren Größe (Bild 5) mit seinem ausgewachsenen Käuferschutz auch nicht für alle Anbieter zu empfehlen.



Aufholjagd: PayPal überholt eBays Kerngeschäft vor dem Spin-off (Bild 5)

Revolvierende Unternehmenskredite

Auf der Money2020-Konferenz in Las Vegas (Bild 6) betonte Daniel H. Dan Schulman, der Geschäftsführer von PayPal, in seiner Keynote, PayPal habe über 60.000 Unternehmen die stattliche Summe von über eine Milliarde US-Dollar geliehen. Revolvierende Unternehmenskredite seien zwar ein nur kleiner, aber sehr wichtiger Teil von PayPals Unternehmensstrategie, die darauf ausgerichtet ist, Fehler der Vergangenheit auszumergen.

Zu den interessantesten Aufgabenstellungen beim Entwickeln von E-Commerce-Lösungen zählen wiederkehrende Zahlungen, im PayPal-Jargon ganz einfach *Recurring Payments* genannt. Leider ist eben diese Zahlungsart bei PayPal für Verkäufer in Deutschland nicht vorgesehen. Als Workaround können Sie auf sogenannte Referenztransaktionen ausweichen, um wiederkehrende Transaktionen dennoch zu realisieren.

Eine Referenztransaktion ist eine finanzielle Transaktion, von der sich nachfolgende Transaktionen ableiten lassen. Hat ein Besucher ein Produkt erst einmal gekauft, lässt sich innerhalb von zwei Jahren aus diesem Vorgang eine weitere, darauf basierende Zahlung ableiten. Erst nach zwei Jahren verschwindet die hierzu benötigte ursprüngliche Transaktions-ID aus dem System, weswegen es dann unmöglich wird, den ursprünglichen Kaufvorgang als eine Referenztransaktion zu nutzen.

Es gibt drei Methoden, um Referenztransaktionen unter Verwendung des sogenannten *Express Checkouts* von PayPal zu realisieren (Bild 7):

- Zuerst richten Sie eine Zahlungsautorisierung ein, danach erstellen Sie eine Zahlungsvereinbarung.
- Zuerst richten Sie eine Zahlungsvereinbarung ein, danach lösen Sie eine Zahlung aus, sobald Ihnen eine Zahlungsvereinbarung vorliegt.
- Sie richten wiederkehrende Zahlungen (*Recurring Payments*) mittels Referenztransaktionen ein.

Im ersteren dieser drei Szenarien richten Sie durch den Aufruf der Methode *SetExpressCheckout* eine Aufforderung zur Zahlungsautorisierung ein und leiten den Benutzer einmal an PayPal weiter, damit dieser den Zahlungsvorgang genehmigt.

Nach Erhalt der Zahlungsautorisierung erstellen Sie eine Zahlungsvereinbarung mit Hilfe der *CreateBillingAgreement*-Methode, welche die sogenannte *BILLINGAGREEMENTID* zurückgibt; diese Identifikationsnummer der Zahlungsvereinbarung können Sie nutzen, um weitere Zahlungen anzufordern, ohne den Benutzer erneut zu einer Zahlungsautorisierung zu bemühen.

Im zweiten Szenario rufen Sie die Methode *DoReferenceTransaction* unter Angabe der Parameter *REFERENCEID* (hierbei handelt es sich entweder um eine Zahlungsvereinbarung oder um eine zuvor durchgeführte Zahlung) auf, des gewünschten Typs des Vorgangs (*PAYMENTACTION*, zum Beispiel *Authorization* oder *Sale*). Den Aufruf der Methode *Do-*



Unter Zugzwang: Der PayPal-Stand auf der Money2020-Konferenz in Las Vegas (Bild 6)

ReferenceTransaction können Sie unabhängig davon tätigen, ob sich der betreffende Käufer gerade in Ihrem Shop-System befindet. Ihr Server könnte diese Art wiederkehrender Zahlungen nach Zeitplan auslösen. Möchte man wiederkehrende Zahlungen mittels sogenannter Referenztransaktionen (Szenario 3) durchführen, läuft das Ganze auf die Bezahlung wechselnder Beträge zu variierenden Zeitpunkten hinaus (also nicht periodisch).

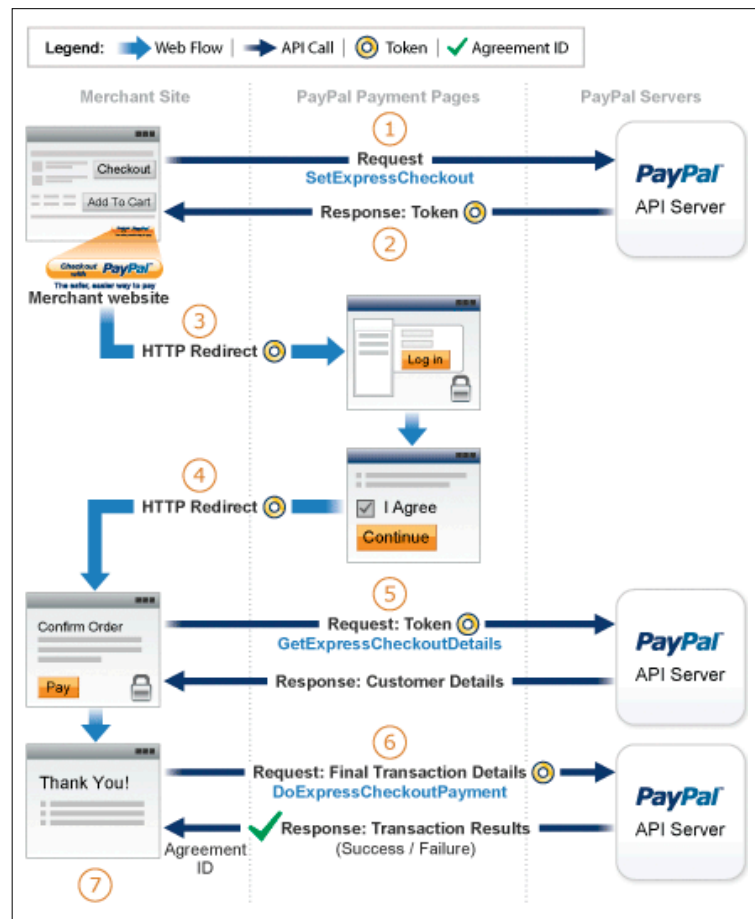
Drei Szenarien aus dem Alltag veranschaulichen die Komplexität der Abwicklung und die große wirtschaftliche Bedeutung wiederkehrender Transaktionen:

- Szenario 1: Der Käufer erwirbt ein Smartphone unter Verwendung von Express Checkout und geht einen Vertrag mit einem Mobilfunkanbieter für zwei Jahre an.
- Szenario 2: Der Käufer erwirbt VoIP-Minuten und stimmt künftig automatischen Abbuchungen zu, sobald sein Guthaben einen bestimmten Betrag unterschreitet.
- Szenario 3: Der Käufer erwirbt ein nutzungsabhängiges Softwareabonnement mit kostenlosem 30-Tage-Probeabo; die erste Zahlung erfolgt erst nach 30 Tagen.

Diese Arten wiederkehrender Zahlungsvorgänge eignen sich besonders gut für die Implementierung unter Verwendung von Referenztransaktionen. Die Fähigkeit, Referenztransaktionen auszulösen, ist standardmäßig deaktiviert. Falls Sie diese Funktionalität benötigen müssen Sie bei PayPal die Freischaltung beantragen.

PayPal Adaptive Payments API

Anpassungsfähige Zahlungen erfolgen zwischen dem zahlenden Kunden und einem oder mehreren Empfängern der Zahlung als eine einfache, parallele oder verkettete Transaktion unter Verwendung der Adaptive Payments API. Der Besitzer der Webanwendung benötigt zur Abwicklung der Zahlung einen PayPal-Business-Account. Bei den (sonstigen) Empfängern der Zahlung besteht hingegen weitgehende Flexibilität, denn diese Teilnehmer können über ein beliebiges PayPal-Konto verfügen.



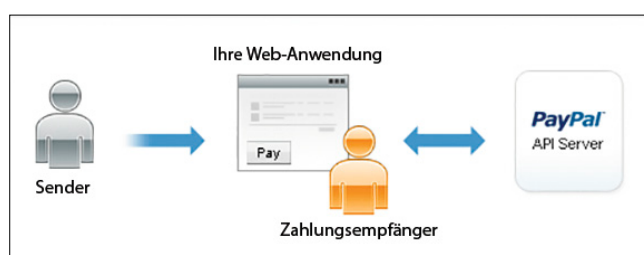
Im Schnellverfahren: Flowchart des ExpressCheckouts von PayPal (Bild 7)

Als Empfänger einer Zahlung kommen zum Beispiel die folgenden Teilnehmer infrage: der Besitzer der Website, ein Widget einer sozialen Netzwerkseite oder ein Dienstleister einer Zahlungsapplikation für Mobiltelefone.

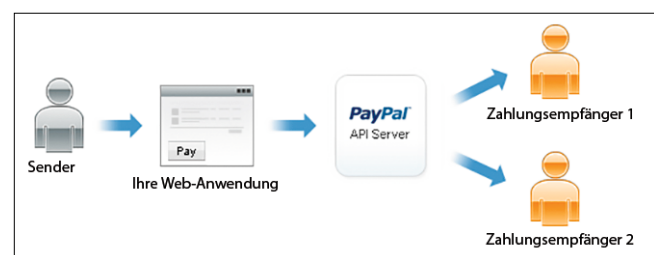
Bei vielen Applikationen kann es sich beim Applikationsbesitzer und -empfänger der Zahlung um eine und dieselbe Person handeln (Bild 8). Dies ist im Übrigen nicht zwingend nötig, denn es geht auch durchaus anders.

Adaptive Zahlungen erlauben drei Arten von Transaktionen: einfache Zahlungen, parallele Zahlungen und verkettete Zahlungen.

Einfache Zahlungen erlauben es dem Absender einer Zahlung, die Zahlung des gesamten Betrags an genau einen ►



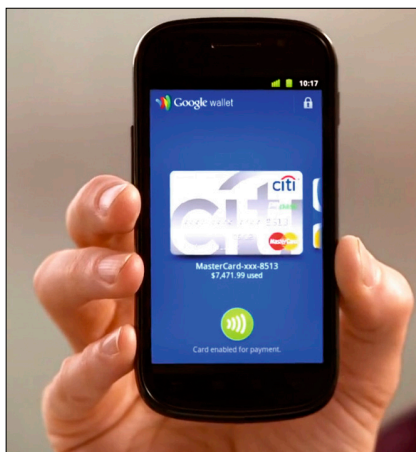
Direkt: Eine einfache Zahlung bietet keine Spielräume. Wenn Sie jedoch das Adaptive Payments API einsetzen, können Sie die Zahlungs-Engine später einmal aufbohren (Bild 8)



Geteilte Freude ist doppelte Freude: Produkte von verschiedenen Händlern werden in einer einzigen Bestellung zusammengefasst und mit einer einzigen Zahlung vergütet (Bild 9)

Empfänger auszuführen. Der Geldfluss vom PayPal-Account des Kunden an den PayPal-Business-Account des Shopbesitzers ergibt sich als Resultat des (einmaligen) Verkaufs von Gütern oder Dienstleistungen.

Parallele Zahlungen sind zum Beispiel dann von Interesse, wenn der Website-Besucher in Ihrem Webshop Waren von mehreren verschiedenen Händlern in den Warenkorb legt und eine einzige Bestellung aufgibt (wie es zum Beispiel im Fall von Amazon möglich ist). Das Diagramm in **Bild 9** zeigt eine solche Zahlung, bei der der Absender mehrere Empfänger in einer parallelen Transaktion vergütet. Ihr Shopping-Cart empfängt das Geld in Ihrem PayPal-Business-Account und leitet die gesplitteten Teilbeträge an die



Quelle: Citibank

Digitalisiert: Google Wallet auf einem Smartphone (**Bild 11**)

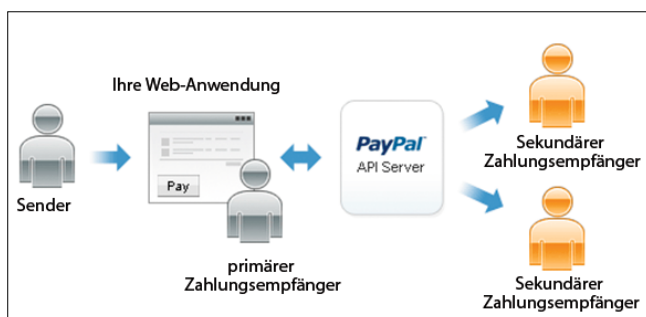
Der Programmierer eines Shopping-Carts ist an dem Verkauf in der Regel nicht selbst finanziell beteiligt. Die Applikation führt Zahlungen zwischen dem Absender und dem Empfänger der Zahlung durch. Mit Hilfe von verketteten Zahlungen ließe sich aber durchaus ein Check-out-System einrichten, das von jedem Kauf eine Kommission an den Entwickler als Sekundärempfänger sendet. Dies ist unter Verwendung von PayPal ohne Weiteres möglich. Dem Käufer ist dies im Übrigen nicht bekannt; der Sender kann die Sekundärempfänger nicht sehen (**Bild 10**).

Google Wallet und Android Pay

Der Gedanke, bei dem Suchriesen

Google durch eine Integration des eigenen Webshops mit dem Check-out-System Google Wallet (**Bild 11**) Punkte zu holen klingt durchaus verlockend. Nicht zuletzt dank des hohen Wiedererkennungswerts des Google-Namens versprechen sich viele Website-Betreiber einen durchschlagenden Verkaufserfolg.

Googles Zahlungs-APIs sind noch unübersichtlicher als die von PayPal. Die Hauptschnittstelle der Google-Wallet-Integration bildet die *Wallet-Klasse* aus Googles APIs für Android. Für In-App-Zahlungen zeichnet Android Pay verantwortlich. Zu den unterstützten Engines zählen unter anderem Braintree (eine PayPal-Tochter) und Stripe. Loyalitätsprogramme, Sonderangebote, Gutscheine und ähnliche Marketing-Aktionen lassen sich mit Hilfe des Wallet Objects API und mit Hilfe des Save to Android Pay API implementieren. Infrage kommen in beiden Fällen Java, Ruby, PHP, Python und C#.NET. Mobile Zahlungen aus einer Android-App heraus sollen mit Googles Bezahl dienst Android Pay möglich werden (der kürzlich vorgestellte Dienst ist genauso wie Apple Pay vorerst nur in den Vereinigten Staaten verfügbar). Android Pay erfordert ein NFC-fähiges Smartphone mit Android OS ab Version 4.4 (KitKat).



Abgesahnt: Der primäre Zahlungsempfänger fungiert als ein Mittelsmann (**Bild 10**)

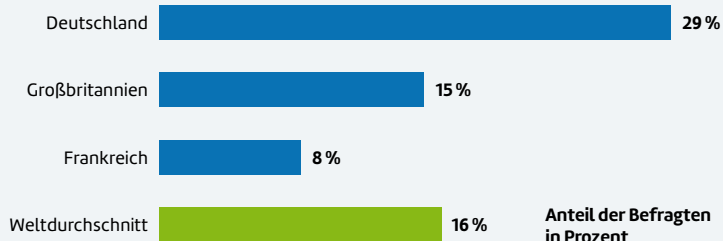
PayPal-Accounts der jeweiligen Händler automatisch weiter. In diesem Szenario ist es dem Absender bekannt, an welche Empfänger die Zahlung tatsächlich hinausgeht.

Verkettete Zahlungen mit PayPal

Ebenfalls interessant sind die sogenannten verketteten Zahlungen. In einer verketteten Zahlung überträgt der Absender dem primären Zahlungsempfänger einen Betrag, von dem der primäre Empfänger einen oder mehrere sekundäre Empfänger bezahlt. Geeignet ist eine solche Konstellation zum Abrechnen von Produktverkäufen beziehungsweise Dienstleistungen, für die eine Kommission fällig wird.

Ein denkbare Einsatzszenario ist eine Online-Reiseagentur, die Flüge, Hotelbuchungen und Autovermietung anbietet. Der Kunde sieht nur die primären Zahlungen an die Reiseagentur. Von den Zahlungen der Kunden werden dann die Kommissionen abgezogen und mit dem Rest die einzelnen Anbieter (im Beispiel also Fluglinie, Hotel und Autovermieter) bezahlt.

Funktion Mobile Wallet



Mobil: Anteil der Befragten, welche die Funktion Mobile Wallet in Anspruch genommen haben, gegenüber allen, denen sie im Smartphone zur Verfügung stand (**Bild 12**)

web & mobile developer 2/2016

Quelle: Statista 2015

Händlern gegenüber zeigt sich Google leider eher unzugänglich und geht auch oft ziemlich rau mit Entwicklern um; für viele Grund genug, sich nach Alternativen umzuschauen.

Innovative Bezahlssysteme kleiner(er) Herausforderer

Inzwischen sind weltweit circa 2,2 Milliarden Kreditkarten im Umlauf. Die Marktdurchdringung von mobilen Endgeräten ist mit schätzungsweise 7,2 Milliarden Benutzern bereits heute mehr als dreimal so groß, auch wenn es noch nicht alle verwenden, denen diese Funktionalität zur Verfügung steht (Bild 12).

Nicht nur Apple (mit Apple Pay), Google (mit Android Pay) und Samsung (mit Samsung Pay) haben in mobilen Zahlungen Umsatzwachstum gewittert. Auch hierzulande haben Anbieter wie Boku oder Zooz das enorme Marktpotenzial erkannt und arbeiten fieberhaft daran, Smartphones und Tablets in ein robustes Zahlungsinstrument zu verwandeln. Das Marktforschungsinstitut PortioResearch (<http://portio-research.com>) erwartet Ende 2016 ein Marktvolumen mobiler Zahlungsarten, einschließlich NFC, in Höhe von einer Billion US-Dollar weltweit.

Mit Zahlungen per Vorkasse oder im Lastschriftverfahren ist im E-Commerce längst kein Blumentopf mehr zu gewinnen. Mit Kreditkarten lassen sich bei Weitem nicht alle Nischen abdecken, zumal viele Käufer dieses Zahlungsinstrument scheuen. PayPal hat genauso viele Anhänger wie eingeschworene Feinde. Zahlreiche kleinere Anbieter versuchen daher, sich als Querdenker zu profilieren und mit innovativen Bezahlssystemen auf diesem rasant wachsenden Markt Fuß zu fassen.

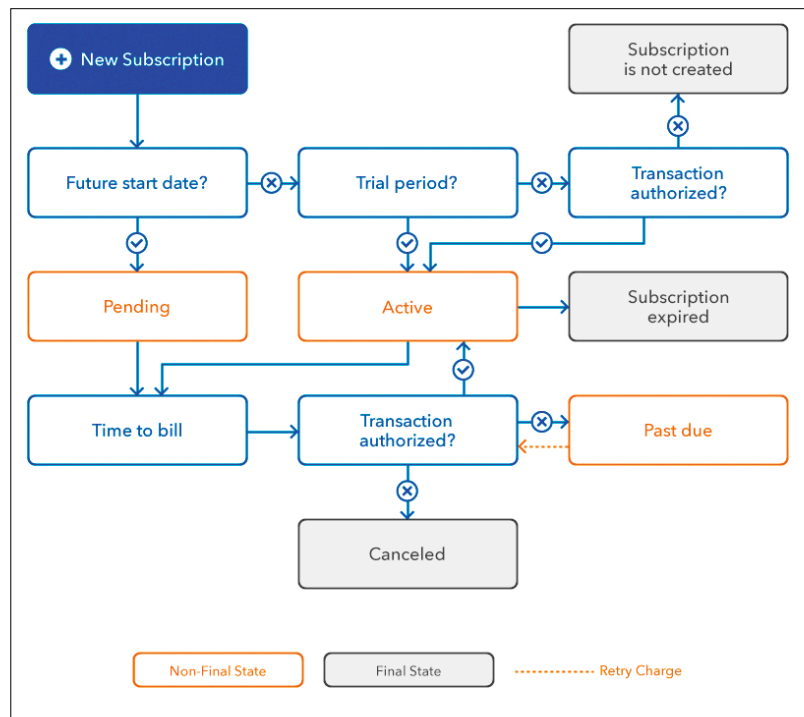
Sie experimentieren mit Lösungen wie Zahlung auf Rechnung nach einer sofortigen Bonitätsprüfung, Ratenkauf und innovativen mobilen Zahlungsmethoden. Das Abbuchen von Guthaben vom Smartphone (eine moderne Variante der Zahlung per Vorkasse) sowie das Übertragen des Zahlungsbetrags auf die mobile Monatsrechnung gehören mit dazu. Für die Betreiber von Webshops scheint sich der Spruch »Konkurrenz belebt das Geschäft« durchaus zu bestätigen; erst recht, wenn diese unter den Anbietern von Bezahlssystemen stattfindet.

Stripe

Stripe (<http://stripe.com>) ist eine einfache, entwicklerfreundliche Online-Zahlungsplattform für reguläre Websites und native Apps für iOS und Android OS mit Unterstützung für In-App-Transaktionen, inklusive Zahlung per Kredit- und Debitkarte, Crowdfunding, Mikrobeträge und Abonnements. Stripe unterstützt mehrere Entwicklungssprachen, darunter Java, Ruby und Python



Expansiv: Patrick Collison, Geschäftsführer und Mitgründer von Stripe (Bild 13)



Flexibel: Flowchart wiederkehrender Zahlungsvorgänge mit Braintree (Bild 14)

(<http://github.com/stripe>) und erfreut sich bei App-Entwicklern einer enormen Beliebtheit (Bild 13). Zu den prominentesten Nutzern zählen Foursquare, die Electronic Frontier Foundation und Reddit. Derzeit unterstützt der Dienst in einer stabilen, produktionssicheren Ausführung Unternehmen in den USA, Kanada, Australien, Dänemark, Finnland, Norwegen, Schweden, Großbritannien und Irland. In Deutschland, Österreich und der Schweiz (sowie in 11 weiteren Ländern) liegt der Dienst in einer Beta-Version vor. Eigenen Aussagen zufolge möchte das junge Start-up die eigenen Dienste um Zahlungen in weiteren Ländern ergänzen.

(PayPal) Braintree v.zero

Braintree (<http://braintreepayments.com>), einst eine Alternative zu Stripe mit vergleichbarer Funktionalität, wurde im Jahr 2013 durch den Zahlungsriesen PayPal akquiriert. Das Resultat der Übernahme ist Braintree v.zero, ein clientseitiges SDK von PayPal, das bestehende Zahlungsmethoden reibungslos erweitert. Braintree v.zero ist unter anderem auch in Deutschland verfügbar.

Braintree zeichnet sich durch leistungsstarke Anti-Fraud-Tools, bemerkenswert flexible wiederkehrende Zahlungsabläufe (Bild 14) und Unterstützung für 3D Secure, ein Protokoll zur Betrugsvermeidung, aus.

Braintree speichert Zahlungsmethoden eines Käufers in dem sogenannten Braintree Vault, einem PCI-konformen Speicher, um die Sicherheit zu maximieren. Jeder Zahlungsmethode wird dann ein eigener Identifikationstoken zugewiesen (*payment_* ►

Listing 3: Zahlungsmethode in Braintree

```
PaymentMethodRequest request = new
PaymentMethodRequest()
    .customerId("7132596")
    .paymentMethodNonce(nonceFromTheClient);

Result<? extends PaymentMethod> result =
gateway.paymentMethod().create(request);
```

Listing 4: Braintree-Transaktion aufrufen

```
TransactionRequest request = new
TransactionRequest()
    .amount(new BigDecimal("100.00"))
    .paymentMethodNonce(nonceFromTheClient)
    .options()
        .submitForSettlement(true)
        .done();

Result<Transaction> result =
gateway.transaction().sale(request);
```

Listing 5: Kreditkartenprüfung in Braintree

```
PaymentMethodRequest request = new
PaymentMethodRequest()
    .customerId("Kaeufer_ID")
    .paymentMethodNonce(nonceFromTheClient)
    .options()
        .verifyCard(true)
        .verificationMerchantAccountId
        ("the_merchant_account_id")
        .done();

Result<PaymentMethod> result =
gateway.paymentMethod().create(request);
```

Listing 6: Beispielresultat

```
Result<PaymentMethod> result =
gateway.paymentMethod().create(request);
result.isSuccess();
// false

CreditCardVerification verification =
result.getCreditCardVerification();

verification.getStatus();
// gateway_rejected

verification.getGatewayRejectionReason();
// GatewayRejectionReason.CVV
```

method_token). Braintree stellt Ihnen eben diesen Token anstelle der eigentlichen Zugangsdaten der jeweiligen Zahlungsmethode zur Verfügung. Diesen Token können Sie auf Ihren eigenen Servern sorglos aufbewahren und künftig nutzen, um (wiederkehrende) Zahlungsvorgänge reibungslos durchzuführen.

Um eine Transaktion auszulösen, müssen Sie zuerst die Zahlungsmethode erzeugen (Listing 3).

Mit dem Aufruf der Methode *Transaction.sale()* unter Angabe des zu zahlenden Betrags führen Sie die Transaktion aus (Listing 4).

Indem Sie die Methode *transaction.sale()* mit dem Parameter *paymentMethodToken()* aufrufen, können Sie eine zuvor gesicherte Zahlungsmethode aus dem Braintree Vault aufrufen und nutzen (Listing 5).

Eine Kreditkartenprüfung unter Verwendung von Braintree in Java illustriert Listing 6. Sollte ein Fehler auftreten, liefert die Methode *PaymentMethod.create()* ein Objekt, dem sich weitere Informationen über den abgebrochenen Vorgang entnehmen lassen (entweder *processor_declined* oder *gateway_rejected*).

PayNearMe, BarPay und Barzahlen.de

PayNearMe (www.paynearme.com) ist eine neuartige Zahlungsplattform in den USA zum Begleichen von Rechnungen für Online-Käufe mit Bargeld. Der Dienst richtet sich an die 24 Prozent der Amerikaner, die über kein traditionelles Bankkonto verfügen beziehungsweise keine Kreditkarte benutzen.

Der Kaufvorgang verläuft einfach und relativ unbürokratisch. Der Online-Käufer wählt beim Check-out PayNearMe als Zahlungsmethode aus und erhält einen Code, den er entweder ausdruckt oder auf einem mobilen Gerät mitnimmt. Im Besitz dieses Codes geht der Käufer zum Beispiel zu einem Fast-Food-Restaurant der 7-Eleven-Kette in seiner Nähe und zahlt dort in bar. Der Kassierer scannt den Code vom Ausdruck oder vom Display des Mobilgeräts ein, bucht die Transaktion und händigt dem Kunden eine Zahlungsbestätigung aus. Benutzer des PayNearMe-Zahlungssystems können so ihre Online-Käufe oder auch Rechnungen für Strom und die Miete ohne ein Bankkonto und ohne eine Kreditkarte mit sofortiger Wirksamkeit begleichen.

PayNearMe setzt sich dank der deutlich niedrigeren Gebühren von den Banken wohltuend ab. Die lokalen Geschäfte geben sich mit den niedrigen Gebühren gerne zufrieden, da sie dank der zusätzlichen Besucherströme, die sie sonst niemals sehen würden, in ihrem Kerngeschäft höhere Umsätze erwirtschaften können. Für kleine und mittlere Unternehmen bringt PayNearMe zusätzlichen Fußgängerverkehr zahlungsfreudiger Kundschaft.

PayNearMe unterscheidet sich von PayPal durch die unkomplizierte und schnelle Abwicklung von Transaktionen. Der Dienst kann auch diejenigen Online-Nutzer erreichen, die über nicht ein PayPal-Konto verfügen.

PayNearMe bereitet sich gerade auf die Internationalisierung vor und plant, auch in Deutschland präsent zu sein. Mit BarPay der Berliner EZV Gesellschaft für Zahlungssysteme

mbH (<http://ezv-gmbh.de>) und Barzahlen (www.barzahlen.de) der ebenfalls in Berlin beheimateten Cash Payment Solutions GmbH gibt es hierzulande aber schon zwei andere etablierte Dienste, sodass sich PayNearMe in Deutschland auf Wettbewerb freuen darf.

Square

Square (<http://squareup.com>) hat sich zum Ziel gesetzt, die technischen Hürden für den Einsatz elektronischer Zahlungsmittel in Kleinunternehmen zu senken. Die Strategie von Square basiert auf einem winzigen Kreditkartenleser für das iPhone, iPad und für Android-Geräte. Square ermöglicht die Zahlung von Mikrobeträgen unter Nutzung eines Smartphones als mobile Kasse via Mikrochip, EMV oder NFC.

Ein Kleinunternehmer im Besitz eines Durchzugslesers von Square, der an ein Smartphone oder ein Tablet angeschlossen ist, kann jederzeit eine Kreditkarte als Zahlungsmittel akzeptieren. Die Transaktionsgebühr beträgt lediglich 2,75 Prozent und trägt zu der hohen Beliebtheit von Square insbesondere unter jungen Start-ups im Einzelhandel bei. Leider ist der Kreditkartenleser derzeit nur in den USA, in Kanada und in Japan verfügbar. In Deutschland funktioniert die mobile App von Square, Square Register, vorerst als eine mobile Kasse für Bargeldtransaktionen mit Fähigkeiten zur Personalverwaltung und Möglichkeiten zur Integration mit dem Webshop der jeweiligen Niederlassung.

Virtuelle Kreditkartennummern

Inzwischen sind sogar einige konventionelle Banken aufgewacht.

Als Reaktion auf das Wachstum innovativer Online-Zahlungssysteme haben einige der konventionellen US-Banken, darunter die Citibank, virtuelle Kreditkartennummern eingeführt. Der Bankkunde kann im Rahmen der bestehenden Kreditlinie selbst eine neue Web-Kreditkarte mit dem gewünschten Verfügungsrahmen und der gewünschten Gültigkeitsdauer direkt im Online-System des Kreditkartenanbieters erzeugen. Eine solche virtuelle Kreditkartennummer gilt typischerweise nur für Transaktionen mit einem bestimmten Online-Händler, um die Sicherheit weiter zu verbessern. Ein Hacker könnte diese Information nämlich schlecht erraten und falls doch, wäre die Kreditkarte beim zweckfremden Einsatz ohnehin unbrauchbar. Mit einer virtuellen Kreditkartennummer können Bankkunden PayPal mit Leichtigkeit umgehen.

Wesentlich interessanter ist Square für deutsche App-Entwickler, da sie mit Square-kompatiblen Apps Millionen von Händlern weltweit adressieren können. Eine Anmeldung ist unter der Adresse <https://squareup.com/signup> möglich. Die Mitgliedschaft im Entwicklerprogramm, das Ihnen eine ►



Upgrades für Ihr Know-How



Node.js & Co. – Entwickeln für die Cloud

Trainer: Golo Roden

3 Tage, Termin & Ort nach Absprache
EUR 2.399,- zzgl. MwSt.

JavaScript für C#-Entwickler

Trainer: Golo Roden

3 Tage, Termin & Ort nach Absprache
EUR 2.399,- zzgl. MwSt.

Weitere Informationen zu den Inhalten der Trainings finden Sie auf unserer Website.

Ihr Wunschtraining ist nicht dabei?
Kein Problem – wir bieten Ihnen maßgeschneiderte Schulungen an.

developer-media.de/trainings

Listing 7: Verkaufsberichte unter Verwendung des Square Connect API

```

<?php
require_once 'Pfad/zur/Unirest-Bibliothek.php';
$accessToken = 'WERT_DES_TOKENS';
$connectHost = 'https://connect.squareup.com';

function formatMoney($money) {
    return money_format('%+.2n', $money / 100);
}

function get2015Payments() {
    global $accessToken, $connectHost;
    # Jahr 2015 in der Zeitzone Berlin
    # alle Parameter werden URL-codiert
    $parameters = http_build_query(
        array(
            'begin_time' => '2015-01-01T00:00:00+01:00',
            'end_time'   => '2016-01-01T00:00:00+01:00'
        )
    );
    $requestHeaders = array (
        'Authorization' => 'Bearer ' . $accessToken,
        'Accept' => 'application/json',
        'Content-Type' => 'application/json'
    );
    $payments = array();
    $requestPath = $connectHost . '/v1/me/payments?' .
    $parameters;
    $moreResults = true;
    while ($moreResults) {
        $response = Unirest\Request::get($requestPath,
            $requestHeaders);
        $payments = array_merge($payments,
            $response->body);

        <https://connect.squareup.com/v1/MERCHANT_ID/
        payments?batch_token=BATCH_TOKEN>;rel='next'
        if (array_key_exists('Link', $response->headers)) {
            $paginationHeader = $response->headers['Link'];
            if (strpos($paginationHeader, "rel='next'") !==
                false) {
                $requestPath = explode('>', explode('<',
                    $paginationHeader)[1])[0];
            } else {
                $moreResults = false;
            }
        } else {
            $moreResults = false;
        }
    }
    $seenPaymentIds = array();
    $uniquePayments = array();
    foreach ($payments as $payment) {
        if (array_key_exists($payment->id,
            $seenPaymentIds)) {
            continue;
        }
        $seenPaymentIds[$payment->id] = true;
        array_push($uniquePayments, $payment);
    }
    return $uniquePayments;
}

function printSalesReport($payments) {
    $collectedMoney = $taxes = $tips = $discounts =
    $processingFees = 0;
    $returned_processingFees = $netMoney = $refunds = 0;
    foreach ($payments as $payment) {
        $collectedMoney = $collectedMoney +
        $payment->total_collected_money->amount;
        $taxes = $taxes + $payment->tax_money->amount;
        $tips = $tips + $payment->tip_money->amount;
        $discounts = $discounts +
        $payment->discount_money->amount;
        $processingFees = $processingFees +
        $payment->processing_fee_money->amount;
        $netMoney = $netMoney +
        $payment->net_total_money->amount;
        $refunds = $refunds +
        $payment->refunded_money->amount;
    }

    $basePurchases = $collectedMoney - $taxes - $tips;
    echo '<pre>';
    echo '==SALES REPORT FOR 2014== ' . '<br/>';
    echo 'Gross Sales: ' . formatMoney($basePurchases -
        $discounts) . '<br/>';
    echo 'Discounts: ' . formatMoney($discounts) .
        '<br/>';
    echo 'Net Sales: ' . formatMoney($basePurchases) .
        '<br/>';
    echo 'Tax collected: ' . formatMoney($taxes) .
        '<br/>';
    echo 'Tips collected: ' . formatMoney($tips) .
        '<br/>';
    echo 'Total collected: ' .
        formatMoney($basePurchases + $taxes + $tips) .
        '<br/>';
    echo 'Fees: ' . formatMoney($processingFees) .
        '<br/>';
    echo 'Refunds: ' . formatMoney($refunds) . '<br/>';
    echo 'Fees returned: ' . formatMoney(
        $returned_processingFees) . '<br/>';
    echo 'Net total: ' . formatMoney($netMoney +
        $refunds + $returned_processingFees) . '<br/>';
    echo '</pre>';
}

$payments = get2015Payments();
printSalesReport($payments);
?>

```

Aufnahme in den App-Markt des Anbieters beschreiben kann, setzt die Zustimmung von Square für Ihr avisiertes App-Projekt voraus; diese Freigabe können Sie formlos per E-Mail an developers@squareup.com einreichen.

Nach dem Einholen einer Genehmigung des Anbieters können Sie in Ihrer App das Square Register API nutzen, um mit der mobilen App Square Register zu kommunizieren.

Mit dem Square Connect API können Sie mobile Apps für das Square-Ökosystem von Lösungen entwickeln, um sie um Funktionalität wie die Berechnung von Steuern, die Erstellung von Verkaufsberichten und die Auswertung von Daten aus diversen Verkaufskanälen zu erweitern.

Listing 7 illustriert die Erstellung eines Verkaufsberichts unter Verwendung des Square Connect API. Das Beispiel erfordert die PHP-Bibliothek Unirest, verfügbar unter <http://unirest.io/php.html>. Den individuellen Token Ihrer Anwendung erhalten Sie im Dashboard unter <https://connect.squareup.com/apps>.

Zu den prominentesten Vorzeigekunden für das Zahlungssystem zählt Starbucks. »Starbucks hat sich als ein Distributionskanal bewährt«, sagt Sarah Friar, Finanzvorstand von Square. Sobald sich ein Jungunternehmer im Besitz eines Kartenlesegeräts befindet, sammelt Square detaillierte Nutzungsstatistiken und zieht diese bei der Entscheidungsfindung über den Aufbau einer langfristigen Geschäftsbeziehung heran. In den USA konnte Square ein Transaktionsvolumen in Höhe von 100 Millionen US-Dollar pro Tag (15 Milliarden US-Dollar pro Jahr) erreichen und plant eigenen Aussagen zufolge eine weltweite Expansion.

RatePay

Bei RatePay (<http://ratepay.de>) handelt es sich um ein Zahlungssystem zum Abwickeln von Kreditkartentransaktionen mit Unterstützung für Ratenzahlungen und den Kauf auf Rechnung ohne Einsatz des Postident-Verfahrens (**Bild 15**).

Das Berliner Start-up RatePay GmbH bringt einen kunden nahen Service und ein tiefgehendes Verständnis für die Besonderheiten des deutschsprachigen Wirtschaftsraums mit. Im Gegensatz zu der noch jungen PayPal-Tochter BillMeLater, Inc., welche Ratenzahlungen derzeit ausschließlich in den USA bereitstellt, adressiert RatePay GmbH auch deutsche Online-Händler.

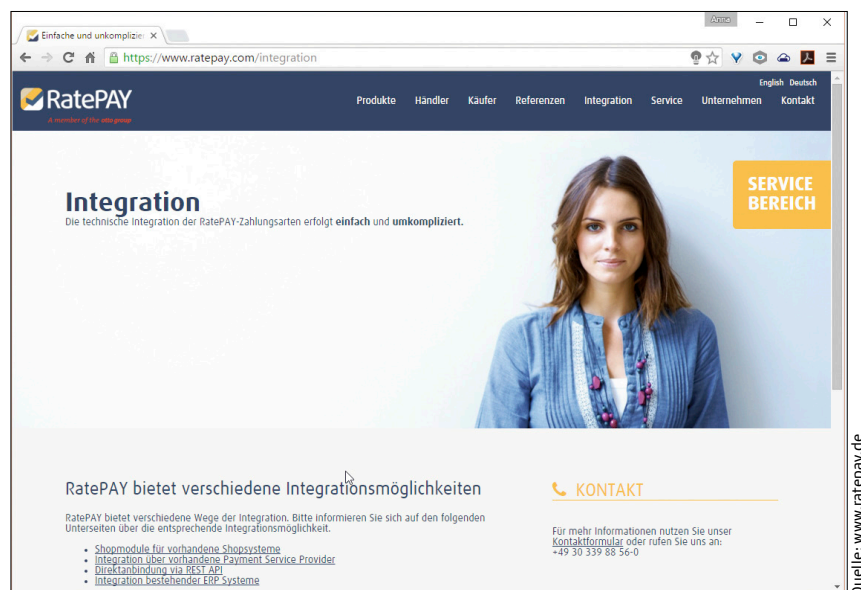
RatePay unterstützt sowohl klassische Ratenzahlungen als auch den Kauf auf Rechnung. Die Plattform bietet dem Käufer eine sichere Zahlungsmethode und sichert den Händler vor dem Risiko eines Zahlungsausfalls ab.

Möchte ein Kunde in einem Webshop auf Raten zahlen, so führt RatePay online eine Risikoabschätzung durch, indem es sowohl die Identität des Antragstellers als auch seine Bonität prüft. Bewertungs- und Prüfungsvorgänge sowie das gesam-

te Risikomanagement führt das Unternehmen in Eigenregie durch; lediglich beim Debitorenmanagement kommen handverlesene externe Dienstleister zum Einsatz.

Während die US-amerikanische PayPal-Tochter BillMeLater bei der Kreditvergabe routinemäßig auf die aktuellen Bewertungsnoten der Bonität des Kunden (das sogenannte Credit Rating) zurückgreift, nimmt die deutsche RatePay GmbH die Risikoeinschätzung unter Verwendung eines hauseigenen Bewertungssystems in die eigenen Hände.

Das Risikomanagement scheint dem Unternehmen sehr gut zu gelingen. Der Verzicht auf das zeitraubende und um-



Der Kunde ist König: Die Kaufabschlüsse erhöhen sich nachweislich bei einer größeren Auswahl an Zahlungsarten (**Bild 15**)

ständige Postident-Verfahren erlaubt es in Kombination mit dem hauseigenen Bewertungssystem völlig automatisiert und in Echtzeit, die Kreditwürdigkeit des Käufers verbindlich zu ermitteln. Gleichzeitig schützt das RatePay-System den Händler. Sollte der Käufer seine Rechnung – aus welchen Gründen auch immer – nicht begleichen können, wird der Webshop für die Lieferung dennoch vollständig bezahlt. Es fällt lediglich eine kleine Gebühr von der Bank an, an die der Webshop seine Forderung veräußert.

Das hauseigene Risikobewertungssystem stellt für die Berliner RatePay GmbH einen wichtigen Wettbewerbsvorteil dar. RatePay integriert sich in die jeweiligen Shopsysteme über ein REST-API.

BillPay

Bei BillPay (<http://billpay.de>) handelt es sich um ein Zahlungssystem mit Unterstützung für den Kauf auf Rechnung, im Lastschriftverfahren und auf Raten (die sogenannte Teilzahlung) durch private (B2C) und gewerbliche (B2B) Kunden im DACH-Raum und in den Niederlanden.

Die Berliner BillPay GmbH wurde unter dem Namen Jade 968. GmbH von Nelson Holzner und Frank Biedka gegründet. ►

det und kann inzwischen auf insgesamt über viertausend Online-Shops verweisen. Das Besondere daran: Billpay.de übernimmt das aufwendige Risiko- und Forderungsmanagement beim Online-Verkauf an private sowie gewerbliche Kunden in Deutschland, Österreich und in der Schweiz und schützt Online-Händler mit einer hundertprozentigen Zahlungsgarantie.

Bei allen BillPay-Plug-ins und Kernintegrationen ist allerdings eine manuelle Aktivierung notwendig (diese kann je nach Art der Integration Teil eines automatischen Ablaufs sein). Unter den insgesamt 18 unterstützten Shopsystemen befinden sich neben Magento auch PrestaShop und osCommerce. Wer eine andere Lösung einsetzen oder ein eigenes Modul entwickeln möchte, kann auf BillPay-Dienste via API zugreifen. Die Core APIs gehen allerdings kaum mit der Zeit: Sie unterstützen die veraltete Version 1.6 von Java, .NET ab der Version 2.0 bis 4.5 und neben PHP 5.x das veraltete und potenziell unsichere PHP 4.x. Den Kern der Shop-Integration bildet ein JavaScript-Widget zur Einbindung der BillPay-Zahlarten im Rahmen des Check-out-Vorgangs.

Boku: Zahlen per Mobilfunkrechnung

Bei Boku (www.boku.com/de) handelt es sich um eine Zahlungsplattform für Online-Transaktionen, die über einen Mobilfunkvertrag abgewickelt werden. Boku erlaubt es deutschen Online-Händlern, mobile Zahlungen mit und ohne Mehrwertsteuer abzuwickeln.

Mit seiner Akquisition des deutschen Zahlungsanbieters Mopay im Jahre 2014 konnte Boku seine vier Millionen Verbraucher starke Kundenbasis mit 250 mobilen Anbietern in 67 Ländern um eine weitere Milliarde und 13 zusätzliche Länder ausweiten. Diese imposante Reichweite hat die Marktposition des Anbieters erheblich gestärkt.

Website-Entwicklern bietet Boku mehrere sehr flexible APIs. Durch die Integration des Dienstleistungspakets Boku Mobile Billing in einen Webshop können Online-Händler Zahlungen über die Telefonnummer des Käufers abbuchen. Im Gegensatz zu PayPal braucht der Käufer hier (anders als der Händler) weder ein Bankkonto anzugeben noch einen mühseligen Registrierungsvorgang über sich ergehen zu lassen. Der Käufer trägt in das Online-Formular seine Telefonnummer ein und bekommt umgehend eine SMS-Nachricht mit einer Zahlungsaufforderung zugeschickt. Um die Zahlung auszulösen, muss der Käufer die SMS-Nachricht beantworten. Die Transaktion belastet dann einfach die monatliche Telefonrechnung des Käufers.

In-App-Käufe lassen sich über die sogenannte Boku-1-Tap-Lösung implementieren. Das Boku 1-Tap SDK stellt Online-Händlern eine schnelle und unkomplizierte Check-out-Lösung zur Verfügung, die zu höheren Konversionsraten und damit auch zu höheren Umsätzen führen soll. Die Online-Transaktion wird umgehend und komplett innerhalb der aktiven App abgewickelt; der zu zahlende Betrag landet auf der



Fix: Martin Poschenrieder, Mitgründer und Geschäftsführer des Start-ups Testmunk.com (Bild 16)

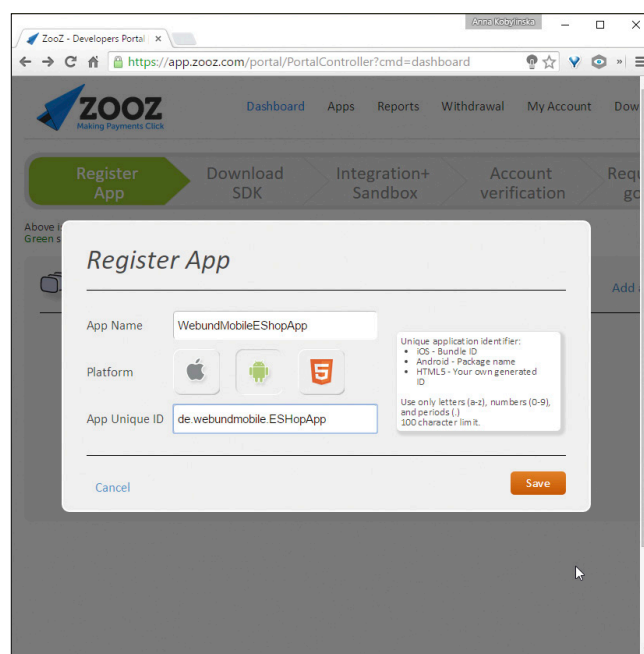
nächsten Telefonrechnung des Käufers. Da der Mobilfunkanbieter bis zur Fälligkeit der Rechnung Bank spielt und der Betrag der Transaktion bis zum Erhalt der Rechnung rein abstrakt bleibt, ist die Hemmschwelle des Nutzers beim Kauf vergleichsweise gering.

Boku bietet auch ein Online-System, das sich durch eine hohe Sicherheit auszeichnet. Die weltweite Kompatibilität verdankt das System unter anderem den flexibel anpassbaren Steuerregeln. Ein durchdachtes Preismodell sorgt dafür, dass sich die Plattform sowohl für Großunternehmen als auch für den Mittelstand und für Kleinunternehmer eignet. Die Integration von Boku auf eine sichere Art und Weise ist nahezu trivial.

Boku hat sich bemüht, die komplexe Problematik der Mehrwertsteuer-Abrechnung elegant zu lösen. Leider ist die zugehörige Funktion standardmäßig deaktiviert. Indem Sie das Feld *operator-tax-treatment* beim Callback der Resultate der Transaktion aktivieren, liefert Boku eine von drei möglichen Antworten zurück:

- **NO_LOCAL_TAX_PAID:** Der Mobilfunkanbieter hat die Mehrwertsteuer nicht gezahlt. Der E-Commerce-Anbieter haftet für die Mehrwertsteuer.
- **LOCAL_TAX_PAID:** Alle Steuern sind bereits auf die Mobilfunkrechnung des Käufers übertragen worden.
- **UNCONFIRMED:** Die Haftbarkeit des E-Commerce-Anbieters für die Zahlung der Mehrwertsteuer konnte nicht überprüft werden.

Diese scheinbar einfache Lösung für mehrwertsteuerpflichtige B2C-Anbieter geht leider am Markt vorbei. Im ersten der drei Fälle genügt es nicht, einfach einen Bruttobetrag abzu-



Einfach: Registrieren einer mobilen App bei Zooz (Bild 17)

buchen, denn dieser ergibt sich aus dem Nettopreis nach der Anwendung des korrekten Mehrwertsteuersatzes, der aber sogar bei digitalen Gütern wie E-Books vom Wohnsitz des Verbrauchers abhängt. Aus diesem Grund sollten interessierte Webshop-Betreiber unbedingt mit ihrem Steuerberater Rücksprache halten, und zwar bevor sie mit dem Gedanken spielen, Boku zu integrieren. Die scheinbare Simplität der Integration kann hier ganz schön nach hinten losgehen. Ansonsten bleibt einem nur eines übrig: für jedes Land ein eigener Webshop.

Zu den Nachteilen von Boku zählt nicht zuletzt die hohe Abhängigkeit von der Fähigkeit des Mobilfunkanbieters, beim Auftreten von Buchungsfehlern (beispielsweise falscher Mehrwertsteuer-Satz auf der Rechnung) einen zufriedenstellenden Kundendienst zu leisten.

Zu den Vorzeigekunden von Boku zählen Facebook, Nokia, Swisscom und Bigpoint SARL und Co, SCS, Europas größte Spielplattform. Zooz ging andere Wege als Boku.

Zooz

Zooz betrat den Markt für mobile Zahlungssysteme als ein wagemutiger Herausforderer mit einer einzigartigen Zahlungsplattform. Inzwischen ist diese zu einem benutzerfreundlichen Zahlungssystem herangewachsen, das mit einer extrem einfachen Integration in Webshops und mobile Apps glänzt.

Das Unternehmen hat sich auf mobile Zahlungsvorgänge spezialisiert, ohne sich jedoch auf einen einzigen Ansatz zu beschränken. Zooz unterstützt sowohl die Zahlung per Kreditkarte als auch per PayPal. Zu den unterstützten Technologien zählt unter anderem NFC. Um den Bezahlvorgang stets komfortabel zu gestalten, nutzt Zooz Erkenntnisse aus der Analyse des Kaufverhaltens.

Zooz weiß mobile Käufer mit einem sehr komfortablen Bezahlvorgang nahezu zu verwöhnen. Der Ansatz basiert auf einer eingehenden Analyse der Verhaltensweisen und Bedürfnisse der Zielgruppe und nicht ausschließlich auf den technischen Aspekten des Zahlungsvorgangs. Zooz arbeitet mit führenden Zahlungs-Gateways, Kreditkartenprozessoren und E-Wallets zusammen. Die Unterstützung von PayPal ist mit dabei (Bild 16).

Zooz mag über keine deutschsprachige Webseite verfügen, und doch zählen zu den prominentesten Anwendern von Zooz deutsche Unternehmen wie das Berliner Start-up Favor.it labs GmbH (<http://favor.it>).

Florian Hübner, Gründer & CTO von Favor.it labs GmbH, sagt dazu: »Wir legen den Schwerpunkt auf das Verständnis und die Verbesserung der Interaktionen lokaler Unternehmen mit ihren Kunden. Wir waren wirklich begeistert, dass wir mit Zooz mobile Zahlungen schnell und einfach integrieren konnten. Die Lösung funktioniert gut, der Service ist großartig, und wir sind erleichtert, solch verlässliche Partner zu haben, denn es geht ja um den Umgang mit unseren Einnahmen.«

Zugang zu dem benötigten SDK erhalten Sie nach einer Anmeldung unter <https://app.zooz.com/portal/signup.jsp> und der Registrierung Ihrer App (Bild 17). ►

Komprimiertes Know-how für Entwickler

Microsoft TFS 2015 - Neuerungen

Referent: David Tielke
On-demand, 60 Minuten



Codequalität mit JavaScript

Referent: Golo Roden
On-demand, 90 Minuten



Architektur auf der Serviette - Softwarezellen

Referent: Ralf Westphal
On-demand, 60 Minuten



MS SQL Server 2016 – Neues für Entwickler

Referent: Thorsten Kansy
On-demand, 60 Minuten



SOLID Prinzipien

Referent: David Tielke
On-demand, 60 Minuten



jQuery – Teil 1

Referent: Johannes Hoppe
On-demand, 60 Minuten



Softwarequalität für Einsteiger

Referent: David Tielke
On-demand, 60 Minuten



Die Geschäftslogik des Webshops

Hinter dem virtuellen Schaufenster eines Webshops werkelt eine Webapplikation, welche die Geschäftsabläufe leise vor sich hin abwickelt. Diese Webapplikation besteht üblicherweise aus einem Content-Management-System (CMS), das den Katalog mit Angeboten aus einer Datenbank auffüllt und das Auffinden von Produktseiten ermöglicht. Der Warenkorb lässt sich üblicherweise als ein Modul implementieren. Zum Abschließen eines Bestellvorgangs wird letztendlich ein Check-out-System benötigt, das Bestellungen entgegennimmt und Zahlungen abwickelt.

Content-Management-Systeme wie Drupal, WordPress und TYPO3 unterstützen leistungsstarke E-Commerce-Module. Darüber hinaus gibt es spezialisierte CMS mit Shop-Funktionalität. In diese Kategorie fallen unter anderem osCommerce und Zen Cart (Bild 18).

Die große Auswahl an scheinbar kostenlosen Shop-Systemen trägt, denn die meisten dieser Lösungen sind überladen mit überflüssigen Features und unsauber programmiert.

Eine positive Ausnahme stellt Magento dar, eine kostenlose E-Commerce-Plattform auf Open Source-Basis. In Magento basiert das Layout auf Vorlagen, um das Aktualisieren des Systems zu vereinfachen. Sie können dem Käufer eine Übersicht der zuletzt besuchten Produktseiten präsentieren und die Empfehlungs-Engine einspannen, um verschiedene Produkte miteinander zu verknüpfen und neue Klickströme zu generieren. Magento beinhaltet bereits einen Warenkorb, aber bei den meisten anderen Content-Management-Systemen kann man dies nicht voraussetzen. In der Regel müssen Sie diese Funktionalität nachrüsten.

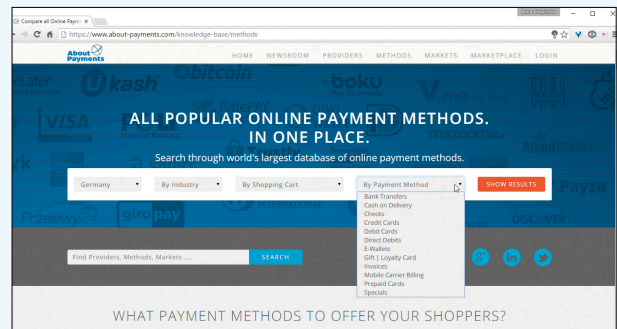
Der Warenkorb

Zu jedem erfolgreichen Webshop gehört ein verlässlicher Warenkorb (ein Einkaufswagen). Der Warenkorb besitzt eine Anbindung an den Online-Katalog auf der einen und an die

Online-Zahlungssysteme

Die schiere Vielfalt an innovativen Zahlungsarten mag überwältigend wirken.

Unter der Adresse <https://www.about-payments.com> können Sie die Kompatibilität, die geografische Verfügbarkeit und die Eignung für einen bestimmten Einsatz einer Vielzahl von Zahlungsmethoden nachschlagen.



Übersicht: Eine sehr empfehlenswerte Knowledgebase zu Zahlungssystemen und Shop-Lösungen

Transaktions-Engine auf der anderen Seite. Bei einer Bestellung landet das gewählte Produkt idealerweise immer in einem robust programmierten Warenkorb, der auch bei mehreren Sitzungen niemals Produkte vergisst. Der Kunde kann dadurch entspannt weiterstöbern. Erst wenn der Käufer mit dem Warenkorb zur Kasse geht, fragt die Webapplikation zusätzliche Daten wie die Lieferadresse und die Rechnungsadresse ab. Webshops, die zuerst eine namentliche Anmeldung erzwingen und erst dann Bestellungen annehmen, haben meist wenig Erfolg. Online-Käufer wollen sich nicht beobachtet fühlen und weigern sich, ihre persönlichen Daten preiszugeben, solange sie noch selbst nicht genau wissen, welche Produkte sie kaufen werden. Geht der Käufer aber irgendwann doch zur Kasse, dürfen Sie ruhig eine Anmeldung vorschlagen, um die Daten der Transaktion in einem Benutzerkonto zu erfassen.

Bei der Wahl eines Warenkorb-Moduls gilt es, den wirklich benötigten Funktionsumfang und Möglichkeiten der Erweiterbarkeit vorab abzuwägen. Bei Lieferungen physikalischer Güter sollte der Warenkorb nicht nur die Mehrwertsteuer, sondern unter anderem auch die Berechnung der Lieferkosten unterstützen.

Der Feinschliff: Konversionsoptimierung

Nachdem die Geschäftslogik Ihrer Webapplikation samt Warenkorb und Kassensystem steht, ist es an der Zeit, sich der Optimierung der Optik und der Navigation zu widmen. Konversionsoptimierung ist nicht nur das Tüpfelchen auf dem i, sondern eine wichtige Voraussetzung für die optimale Usability.

Wer den Bezahlvorgang insgesamt als zu umständlich oder unangenehm empfindet, wird den Kauf meiden, und nichts ist im Internetzeitalter einfacher, als

osCommerce zählt zu den Content-Management-Systemen mit ausgeprägten Shop-Funktionalitäten (Bild 18)

von dem Kauf zurückzutreten und zu einem anderen Webshop zu navigieren. Als Gegenmaßnahmen empfiehlt es sich, das Registrierungs- und Kaufformular auf ein Minimum der wirklich erforderlichen Fragen zu bringen und zusätzlich auch noch einen Gastzugang ohne eine Registrierung zu erlauben.

Die Konversionsrate (auch Konvertierungsrate genannt) misst, wie erfolgreich es Ihnen gelingt, Website-Besucher zur Umsetzung einer bestimmten Handlung zu verleiten. Diese Kennzahl definiert den Erfolg als einen Anteil der sogenannten Konversionen an dem angenommenen Gesamtpotenzial der maximal möglichen Konversionen bei der aktuellen Besucherzahl.

Die Konversionsrate kann zum Beispiel als die Anzahl abgeschlossener Verkäufe an der Gesamtzahl aller Besuche des Webshops berechnet werden.

Von je tausend Besuchern eines Webshops springen vielleicht so um die 260 aus verschiedenen Gründen sofort wieder ab; die sogenannte Absprungrate (Bounce-Rate) liegt in diesem Fall bei 26 Prozent. So um die 350 weitere Besucher des Beispiel-Shops greifen auf die Suchfunktion zu; die übrigen 390 stöbern vielleicht durch die Produktkategorien. Besucher gelangen über verschiedene Wege zu einer Produktseite, ehe sie ein Produkt in den Warenkorb legen. Diese alternativen Navigationswege, die den Besucher ans Ziel führen – im Beispiel also die Suchfunktion und Textnavigation durch Produktkategorien – nennt man Trichter (Funnel). Jeden dieser Funnel können Sie separat beobachten und optimieren.

Beim Erreichen des vordefinierten Ziels durch den Besucher – zum Beispiel beim Aufgeben einer Bestellung – spricht man von einer Konversion.

Der Anteil aller Besucher, die ein konkretes Navigationsziel erreichen und die gewünschte Handlung ausführen, gemessen an der Gesamtanzahl aller Besucher, denen diese Möglichkeit zur Verfügung stand, ist die Konversionsrate des Ziels in dem betreffenden Funnel. Zur Optimierung der Konversionsrate möchten Sie Ihre Besucher in die gewünschten Funnel hineinleiten und ohne störende Ablenkungen zum Ziel führen. Erreicht der Besucher einen Zwischenschritt (einen sogenannten Meilenstein), spricht man von einer Mikrokonversion. Der Begriff bezeichnet ein Ereignis innerhalb einer Konversion, beispielsweise einen Mausklick auf eine bestimmte Schaltfläche beim Einkaufsbummel. Klickt ein Besucher etwa auf *Jetzt kostenpflichtig bestellen*, nachdem er die betreffende Produktseite geladen hat, kommt eine Mikrokonversion zustande. Gibt er die Bestellung dann auch auf, ist die Konversion abgeschlossen. Auch dieses Ereignis können Sie messen, um zu ermitteln, an welcher Stelle des Vorgangs die meisten Besucher abspringen. Gewappnet mit dieser Information können Sie wiederum potenzielle Hürden aus dem Weg schaffen, um Ihre Annahmen anhand der neuen Messwerte zu bestätigen oder zu widerlegen.

Von je tausend Besuchern der Startseite eines Shops werden vielleicht schlussendlich nur 130 Nutzer ein Produkt in den Warenkorb legen, davon werden aber möglicherweise weniger als 90 Besucher tatsächlich eine Bestellung aufge-

ben. Ein solcher Webshop kann also bei dem Kaufvorgang auf eine Konversionsrate von etwa neun Prozent verweisen. Wer diese Zahl verdoppeln kann, verdoppelt automatisch den Umsatz.

Die Optimierung der Konversionsrate erfordert viel Feingefühl, eine Menge Beharrlichkeit und eine gehörige Dosis an Aufgeschlossenheit. Man weiß ja nie, was einen erwartet. Anhand der Konversionsrate lassen sich subtile Usability-Probleme genauso wie nicht offensichtliche Marketing-Fehler diagnostizieren – und korrigieren. Ob ein Redesign der Landeseite oder eine neue Social-Media-Kampagne: Eine umfassende Analyse der Entwicklung der eigenen Konversionsrate erlaubt es, auf Rätselraten zu verzichten, katastrophale Fehlentscheidungen zu vermeiden und das Potenzial der betreffenden Website auszuloten, so weit es geht.

Das Gegenteil der Konversionsrate ist die Ausstiegsrate: der Anteil der Besucher, die ihren Besuch einer Webseite ►

Webshop-Lösungen im Überblick

Die nachfolgende Aufstellung gibt einen Überblick über die bekanntesten Webshop-Lösungen:

- Die E-Commerce-Plattform Magento (kostenfrei in der Community-Edition)
www.magento.com
- Shop-Script 6, ein E-Commerce Framework in PHP (Preis: 249 US-Dollar, für Entwickler kostenfrei)
www.shop-script.com
- Storefronts-Demo
www.shop-script.com/demo
- Backend-Demo
<https://demo2.webasyst.com/webasyst/shop>
- CubeCart Version 6 (kostenfrei)
www.cubecart.com
- osCommerce Online Merchant v2.3 (kostenfrei)
www.oscommerce.com
- PayPal App für osCommerce Online Merchant (kostenfrei)
<http://addons.oscommerce.com/info/9184>
- Zen Cart Version 1.5.4 (kostenfrei)
www.zen-cart.com
- Plug-ins für Zen-Cart
<https://www.zen-cart.com/downloads.php>
- EllisLab Expression Engine (299 US-Dollar für die Lizenz)
<https://ellislab.com/expressionengine>
- X-Cart v4.1.11 (kostenfrei in der Free Edition, sonst ab 495 US-Dollar)
www.x-cart.com
- Warenkorb und Transaktionsabwicklung (ab 59 US-Dollar pro Monat)
www.1shoppingcart.com

abbrechen. Das Minimieren der Ausstiegsrate ist natürlich erstrebenswert, aber es ist an sich noch keine Garantie für den Erfolg. Der Besucher eines Webshops kann endlos im Kreis navigieren und dennoch nichts kaufen. Wer einen Erfolg verbuchen möchte, fokussiert in erster Linie auf die Konversionsrate.

Werden die Erkenntnisse der Konversionsoptimierung tatkräftig umgesetzt, kann sich der Feldzug scheinbar kleiner Optimierungen unmittelbar positiv auf die Geschäftszahlen auswirken. Zahlreiche Beispiele aus dem Online-Handel und dem Online-Dienstleistungssektor belegen es eindeutig.

Für die Optimierung der Konversionsrate gibt es zwei grundlegend unterschiedliche Ansätze.

Der eine Ansatz beruht auf ausgiebigen Tests unter Beteiligung echter Besucher, deren Verhalten ausgewertet wird, um daraus für die inhaltliche Zusammenstellung der Website und ihr grafisches Design Schlüsse zu ziehen. Mit Hilfe von A/B- oder MVT-Tests wird die Reaktion der Zielgruppe auf verschiedene Bilder, Überschriften oder Schaltflächen untersucht.

Der Nachteil dieses Ansatzes besteht darin, dass so nur diejenigen Variablen berücksichtigt werden, die sich in einem der Tests erfassen lassen. Die Gründe für eine niedrige Konversionsrate (und somit eine hohe Absprungrate) können je-

doch sehr vielfältig sein. Schlechtes Design mit schwacher Usability ist nur das kleinste aller Übel. Einige Besucher werden wohl den Bestellvorgang abbrechen, weil ihnen die gebotenen Liefermethoden nicht zusagen, andere empfinden vielleicht die Zahlungsbedingungen als zu restriktiv, oder sie wurden durch eine missverständlich formulierte Anzeige auf die Landing Page gelockt und gehören gar nicht zu der anvisierten Zielgruppe.

Um diesen und ähnlichen Ursachen auf die Spur zu kommen, ist ein tiefgreifendes Verständnis der Zielgruppe vonnöten. Darauf fokussiert der zweite Ansatz zur Konversionsoptimierung: Eine umfassende Zielgruppenforschung und Marktsegmentierung bildet die optimale Grundlage zur Formulierung einer Marketingstrategie und einer maßgeschneiderten Marketing-Botschaft, deren Wirkung sich anschließend anhand der Konversionsrate in Tests bestätigen, widerlegen oder verstärken lässt. Keine Website kann es allen Website-Besuchern gleichermaßen recht machen, aber zumindest lassen sich mittels Tests mit mehreren Varianten hinreichend viele alternative Designs durchtesten, um daraus Schlüsse zu ziehen. Durch eine geschickte Segmentierung der Zielgruppe können Sie Besucher mit unterschiedlichen Wünschen und Vorlieben berücksichtigen und die Konversionsrate weiter steigern.

A/B- und MVT-Tests im Vergleich

Die zwei wichtigsten Tests zur Konversionsoptimierung sind die sogenannten A/B- und MVT-Tests.

In einem A/B-Test werden zwei oder mehr Varianten eines Webshops einander gegenübergestellt (Variante A, Variante B et cetera). MVT steht für einen sogenannten Multivariate-Test. In einem solchen Experiment testen Sie das Zusammenspiel alternativer Versionen mehrerer Inhaltsabschnitte derselben Seite.

Ein klassisches A/B-Experiment geht von einem existierenden Webshop aus und ersetzt diesen zur Laufzeit durch zuvor bereitgestellte alternative Seiten mit ihren eigenen URLs. Einige Besucher bekommen die Variante A des Produktkatalogs, andere die Variante B serviert (Split-Testing), zum Beispiel einmal mit und einmal ohne die Einbindung eines Empfehlungssystems. A/B-Tests lassen sich vergleichsweise einfach implementieren und schnell durchführen, da sie nur ein relativ geringes Besucheraufkommen voraussetzen (generell genügen etwa 100 Besucher pro Variante der Seite im Experiment).

A/B-Tests liefern durchaus nicht immer eindeutige Rückschlüsse auf die Ursachen des abschließenden Urteils. Sie sind jedoch geeignet, um etwa zwei Shopsysteme direkt miteinander zu vergleichen.

Im Entstehen einer Konversion spielen erfahrungsgemäß immer viele einzelne Faktoren eine Rolle. Oft ergeben sich große Verbesserungen der Konversionsrate durch eher geringfügige Design-Optimierungen an vielen Stellen gleichzeitig; diese Informationen gehen jedoch aus einem A/B-Experiment nicht hervor. Im Gegensatz zu A/B-Tests lassen sich mit Hilfe von MVT-Experimenten die Gründe für Verhaltensmuster erforschen und Opti-

mierungspotenziale oft an unerwarteten Stellen aufdecken. Hierzu werden bei MVT-Experimenten verschiedene Kombinationen von Inhaltselementen auf einer Webseite anhand der Konversionsrate untersucht und anschließend miteinander verglichen.

Zu den Vorzügen von MVT-Experimenten zählt ihre hohe Genauigkeit. Diese kommt jedoch mit einem Preis in Form der wesentlich höheren Anforderungen an das Besucheraufkommen. Aus diesem Grund sind MVT-Experimente nicht auf jeder Webpräsenz sinnvoll realisierbar.

Je mehr Besucherzahlen eine Website aufweist, desto mehr Varianten lassen sich im selben Zeitraum testen. Websites mit einem geringeren Besucheraufkommen benötigen bei einer gleichen Anzahl von Varianten entsprechend länger, um zu statistisch relevanten Resultaten zu gelangen. Große Webportale können in einem Bruchteil der Zeit wesentlich mehr Varianten durchtesten. Die Dauer eines Experiments dürfen Sie im Übrigen nicht beliebig lange hinausstrecken, ohne dass die Resultate an Relevanz verlieren, weil Sie auf die tatsächliche Lebensdauer der Cookies nicht wirklich einen Einfluss haben.

Wer weitere Einblicke ins Besucherverhalten wünscht, kann Tools für verhaltensbezogene Zielgruppenansprache (sogenanntes Behavioral Targeting) einspannen. Hierbei werden technische Daten des Besuchers (samt geografischer Ortungsdaten) mit Verhaltensprofilen überlagert, um die Konversionsrate weiter zu erhöhen. Mit Hilfe von Heatmaps und Clickmaps können Sie den Lesefluss und das Klickverhalten untersuchen. Schließlich nutzen einige Website-Betreiber eine Testgruppe, um direktes Feedback einzuholen.

Links zum Thema

- Amazon Payments
<https://payments.amazon.com/home>
- Das PayPal-Entwicklerportal
<https://developer.paypal.com>
<https://www.paypal.com/de/webapps/mpp/partners-and-developers>
- Das Integrationscenter für Barzahlen.de
<https://integration.barzahlen.de/en/api>
- Ein Magento-Modul für BarPay
www.magentocommerce.com/magento-connect/barpay-payment-modul.html

Amazons spektakulärer Erfolg bescherte dem Online-Riesen beachtliche Synergieeffekte. Zum Glück müssen Sie nicht gleich so groß wie Amazon sein, um einen eigenen erfolgreichen Webshop aufzusetzen. Mit etwas Know-how liegt dieses Ziel in jedermanns Reichweite. Planen Sie das Projekt jedoch sorgfältig und überlassen Sie nichts dem Zufall.

Fazit

Der Erfolg eines Webshops steht und fällt mit der Verfügbarkeit geeigneter Bezahlmethoden. Die Kaufabschlüsse erhöhen sich nachweislich bei einer größeren Auswahl an Zahlungsarten. Über viele Dinge kann man geteilter Meinung sein, eines aber ist unstrittig: Mit neuen Zahlungsmethoden geht E-Commerce in die nächste Runde.

Das Herz eines jeden Webshops bildet die Webapplikation, die unter der Haube werkelt und potenzielle Käufer möglichst stressfrei und reibungslos durch den Bestellvorgang begleitet. Nur ein Webshop, der die beliebtesten Zahlungsmethoden seiner avisierten Zielgruppe abdeckt, kann für sich in Anspruch nehmen, das Marktpotenzial wirklich ausschöpfen zu können.

Erfolgreiche Webshops zeichnet außer wiederkehrenden Besuchern ein umfassendes Know-how um die technische Implementierung des Kaufvorgangs mit sorgsam ausgesuchten Zahlungsmodalitäten aus. Ein erfolgreicher Webshop erfordert eine sorgfältige Planung und die durchdachte Integration geeigneter Module. ■



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.

Trainings für Webentwickler

Hybrid-Apps mit Ionic, Cordova und Angular JS

Trainer: Hendrik Lösch
22.-23.02.2016, Hamburg



Cross-Plattform-Apps mit C# und Xamarin

Trainer: Sebastian Seidel
07. – 09.03.2016, München



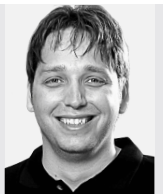
Webanwendungen mit HTML, CSS und JavaScript

Trainer: David Tielke
14.-15.03.2016, München



Webentwicklung mit ASP.NET, MVC und Web API

Trainer: David Tielke
16.-18.03.2016, München



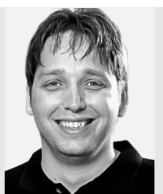
Entwicklung von Windows Universal Apps

Trainer: Lars Heinrich
22.-23.03.2016, Köln



Testen von .NET-Anwendungen

Trainer: David Tielke
21.-22.04.2016, Köln



Usability verstehen und verbessern

Trainer: Armin Reuter
10.-11.03.2016, Köln



METEOR UND REACT

Meteor auf neuen Pfaden

Das JavaScript-Framework Meteor 1.2 umfasst den gesamten Anwendungs-Stack.

Meteor ist ein echtes Full-stack-JavaScript-Framework auf der Basis von Node.js. Im Gegensatz zu vielen anderen Web-Frameworks, die jeweils nur die Client- beziehungsweise die Serverseite der Anwendung abdecken, umfasst Meteor den kompletten Anwendungs-Stack. Quellcode lässt sich hierdurch wahlweise auf dem Client, auf dem Server oder auf beiden Plattformen ausführen.

Clientseitig besitzt das Meteor-Framework eine eigene Template-Engine: Blaze. Durch den Einsatz von Blaze lassen sich HTML-Templates um dynamische Elemente ergänzen. Dies schafft die Voraussetzung dafür, dass ein reaktives Verhalten der Webanwendung umsetzbar ist. Vor dem Erscheinen von Meteor 1.2 war Blaze als View-Engine die einzige offiziell unterstützte Variante des Frameworks. Wer gewohnt war, für die Gestaltung des User Interfaces auf Frameworks wie AngularJS oder React zurückzugreifen, musste sich mit dem Einsatz von Meteor daher umorientieren.

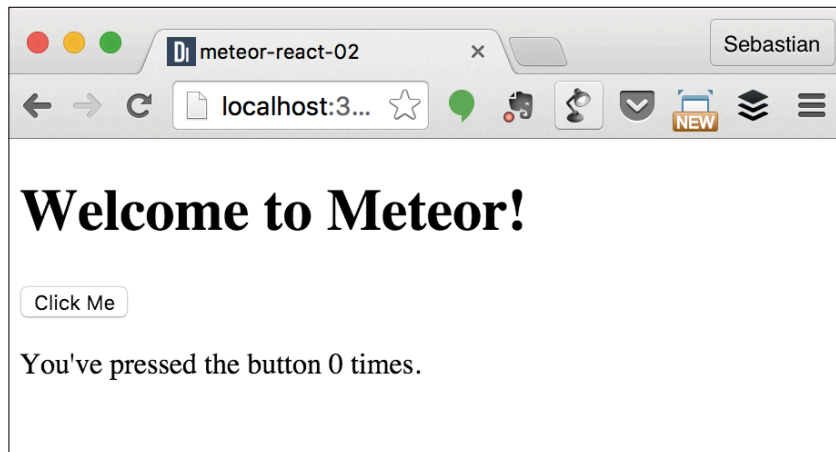
Mit der Version 1.2 des Meteor-Frameworks bietet die Meteor Development Group (MDG) erstmals die gleichwertige und offizielle Unterstützung von AngularJS und React an. Nachdem wir uns in Ausgabe 11/2015 bereits mit dem Einsatz von Angular 2.0 und Meteor beschäftigt haben, finden Sie in den nachfolgenden Abschnitten einen praktischen Einstieg in Meteor 1.2 und React.

Was ist React?

React ist eine JavaScript-Bibliothek, die von Facebook entwickelt und verwendet wird und zur Umsetzung von webbasierten User Interfaces eingesetzt wird. React setzt hierbei durchgängig auf das Konzept der Komponentenorientierung. Jede React-Anwendung ist daher aus einer oder mehreren React-Komponenten aufgebaut.

Listing 1: meteor-react-01.html

```
<head>
  <title>meteor-react-01</title>
</head>
<body>
  <div id="render-target"></div>
</body>
```



Meteor-Standardanwendung nach der ersten Ausführung im Browser (Bild 1)

React ist besonders darauf optimiert, auch in großen Anwendungen mit erheblichem Datenumfang eine gute Performance und ein reaktives Anwendungsverhalten zu liefern. Die Reaktivität äußert sich vor allem darin, dass React automatisch Änderungen an den zugrunde liegenden Datenobjekten erkennt und die Oberfläche entsprechend aktualisiert, ohne dass der Anwendung die Webseite neu laden muss. Die Aktualisierungslogik arbeitet hierbei besonders effizient und tauscht nur die Bestandteile des DOM aus, die tatsächlich von der Änderung betroffen sind. Die offizielle Projektseite von React finden Sie unter <https://facebook.github.io/react/index.html>.

JSX-Syntax

Zur Implementierung von Komponenten bietet React optional die JSX-Syntax an. Hiermit kann der Markup-Code, der zur Generierung der HTML-Ausgabe der Komponente zum Einsatz kommt, direkt in den JavaScript-Code der Komponente eingebettet werden. Der klassische Ansatz, View-Template und View-Logik zu trennen, wird daher von React aufgehoben.

Der Vorteil der JSX-Syntax lässt sich sehr einfach anhand eines kleinen Beispiels erläutern. Die Generierung eines `<a>`-Elements lässt sich in React durch den Einsatz von JavaScript erreichen:

```
React.createElement('a', {href:
'https://facebook.github.io/react/'}, 'Hello!')
```

Hierbei wird die Methode `React.createElement` aufgerufen, um das entsprechende HTML-Element zu erzeugen.

Einfacher und auch leichter lesbar lässt sich dieser Schritt mit JSX umsetzen:

```
<a href="https://facebook.github.io/react/">Hello!</a>
```

Das Link-Element kann in diesem Fall direkt über den Einsatz des entsprechenden HTML-Codes erzeugt werden. Eine Methode zur Generierung des HTML-Codes muss in diesem Fall nicht aufgerufen werden. In den nachfolgenden Beispielen setzen wir daher durchgängig JSX-Code ein.

Vorbereitung der Entwicklungsumgebung

Bevor wir im Folgenden anhand von Beispielanwendungen in die Praxis einsteigen können, gilt es zunächst die Entwicklungsumgebung vorzubereiten und einzurichten.

Der erste Schritt besteht in der Installation des Meteor-Frameworks. Unter Mac OS X und Linux kann dieser Schritt über die Eingabe des folgenden Kommandos im Terminal ausgeführt werden:

```
curl https://install.meteor.com/ | sh
```

Für die Windows-Plattform steht ein Installer zum Download auf <https://www.meteor.com/install> zur Verfügung.

Haben Sie Meteor bereits auf Ihrem System installiert, sollen Sie prüfen, ob das Frameworks in der Version 1.2 oder höher vorhanden ist. Als View-Engine wird React von Meteor erst ab dieser Version unterstützt. Die aktuell installierte Meteor-Version lässt sich über die Kommandozeile ermitteln:

```
meteor --version
```

Falls eine Aktualisierung auf die neueste Version notwendig ist, kann dies ebenfalls schnell und einfach über den Aufruf eines Befehls erledigt werden:

```
meteor update
```

Ist die Installation von Meteor abgeschlossen, können wir im nächsten Schritt dazu übergehen, ein erstes Projekt für eine Meteor-React-Anwendung zu erstellen.

Erstellen eines Meteor-React-Projekts

Im ersten Schritt legen wir ein neues Meteor-Projekt an. Rufen Sie hierzu das folgende Kommando in einem Verzeichnis Ihrer Wahl auf:

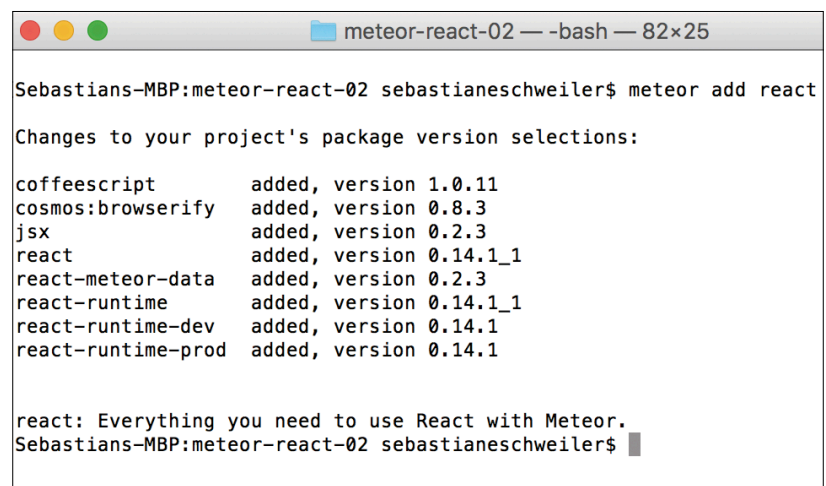
```
meteor create meteor-react-01
```

Im aktuellen Verzeichnis wird daraufhin das Unterverzeichnis *meteor-react-01* angelegt. Dieses Verzeichnis beinhaltet standardmäßig die drei Dateien *meteor-react-01.css*, *meteor-react-01.html* und *meteor-react-01.js*.

In diesen Dateien finden Sie eine einfache Meteor-Beispielanwendung, die Sie ohne weitere Anpassungen starten können. Wechseln Sie zu diesem Zweck in das neue Verzeichnis *meteor-react-01* und rufen Sie in diesem Verzeichnis den Befehl *meteor* auf. Daraufhin wird der Webserver gestartet und die Anwendung kann unter dem URL <http://localhost:3000> im Browser angezeigt werden (Bild 1).

Installation zusätzlicher Pakete

Meteor besitzt einen eingebauten Paketmanager, der es erlaubt, über das *meteor*-Kommando zusätzliche Pakete zum Projekt hinzuzufügen. Damit wir im nächsten Schritt in der Lage sind, die Meteor-Anwendung mit React-Komponenten zu erweitern, müssen nun zunächst einige Pakete hinzugefügt werden. Öffnen Sie daher erneut ein Terminal-Fenster,



```
Sebastians-MBP:meteor-react-02 sebastianeschweiler$ meteor add react

Changes to your project's package version selections:

coffeescript      added, version 1.0.11
cosmos:browserify added, version 0.8.3
jsx               added, version 0.2.3
react             added, version 0.14.1_1
react-meteor-data added, version 0.2.3
react-runtime     added, version 0.14.1_1
react-runtime-dev added, version 0.14.1
react-runtime-prod added, version 0.14.1

react: Everything you need to use React with Meteor.
Sebastians-MBP:meteor-react-02 sebastianeschweiler$
```

Erfolgreiche Installation des React-Pakets im Meteor-Projekt (Bild 2)

wechseln Sie in das zuvor erzeugte Projektverzeichnis und führen Sie im ersten Schritt das folgende Kommando aus:

```
meteor add react
```

Sie erhalten die Rückmeldung, dass neben dem *react*-Paket weitere Pakete, zu denen Abhängigkeiten bestehen, ebenfalls dem Projekt hinzugefügt wurden (Bild 2). Hierzu gehört beispielsweise auch das Paket mit dem Namen *jsx*, das es uns ermöglicht, in den nachfolgenden Beispielen die vorgestellte Syntax einzusetzen. Nach diesem Schritt sind die Voraussetzungen geschaffen, um React innerhalb des Meteor-Projekts verwenden zu können. Bevor wir mit der Implementierung der ersten React-Komponente beginnen, ergänzen wir jedoch noch drei weitere Pakete:

```
meteor add twbs:bootstrap
meteor add kadira:flow-router
meteor add kadira:react-layout
```

Das Paket *twbs:bootstrap* fügt dem Projekt das Bootstrap-Framework hinzu, sodass wir für den Aufbau der Pro- ►

grammoberflächen auf Bootstrap-Komponenten und -Klassen zurückgreifen können.

Mit *kadira:flow-router* wird das FlowRouter-Paket installiert, das es uns später erlauben wird, clientseitiges Routing innerhalb der Meteor-Anwendung umzusetzen. Damit wir FlowRouter außerdem in Verbindung mit React einsetzen können, wird auch noch das *kadira:react-layout* benötigt. Wird eine bestimmte clientseitige Route angesprochen, kann FlowRouter mit dieser Erweiterung die Ausgabe einer React-Komponente einbinden.

Implementierung einer React-Komponente

Sind die Vorbereitungen bis zu diesem Punkt abgeschlossen, ist für den Einsatz von React innerhalb der Meteor-Anwendung alles vorbereitet. Die Meteor-Standardanwendung, die beim Anlegen des Projekts automatisch generiert wurde, soll daher im ersten Schritt in eine einfache Meteor-React-Anwendung umgebaut werden.

Zunächst löschen wir den Inhalt der Datei *meteor-react-01.html* und fügen den HTML-Code aus [Listing 1](#) ein.

Der Body-Bereich der Seite besteht lediglich aus einem `<div>`-Element. Über die zugewiesene ID lässt sich dieses Element eindeutig referenzieren. Im nächsten Schritt nutzen wir dieses `<div>`-Element, um die Ausgabe der ersten React-Komponente in die HTML-Seite einzufügen. Das Element mit der ID *render-target* dient an dieser Stelle also lediglich als Platzhalter und markiert die Stelle, an der der Inhalt der React-Anwendung eingefügt wird.

Im nächsten Schritt löschen wir die Datei *meteor-react-01.js* und erstellen stattdessen die Datei *meteor-react-01.jsx*. Durch die Dateierweiterung *jsx* legen wir fest, dass in dieser Datei die JSX-Syntax zum Einsatz kommen kann. Fügen Sie im Anschluss den Quellcode aus [Listing 2](#) ein.

Der Code sorgt dafür, dass clientseitig beim Start der Anwendung die Methode *React.render* aufgerufen wird. Der Methodenaufruf bindet die Ausgabe der App-Komponente (die wir im nächsten Schritt erstellen werden) in die HTML-Seite der Webanwendung ein. Damit das zuvor aufgenommene `<div>`-Element mit der ID *render-target* durch die Ausgabe der React-Komponente ersetzt wird, werden an den Aufruf zwei Parameter übergeben. Im ersten Parameter wird die React-Komponente, die gerendert werden soll, übergeben. Da wir den JSX-Syntax verwenden, können wir hierzu direkt die HTML-Schreibweise nutzen und das `<App />`-Element übergeben. Im zweiten Parameter wird das HTML-Element referenziert, dass durch die Ausgabe der React-Komponente

ersetzt werden soll. Die Referenz auf das zuvor angelegte `<div>`-Element erhalten wir über die Funktion *document.getElementById*.

Damit die Anwendung funktionsfähig ist, fehlt schließlich noch die Implementierung der React-Komponenten. Die Hauptkomponente der Anwendung ist die bereits eingebundene App-Komponente. Zur Implementierung erstellen wir im Projektverzeichnis eine neue Datei mit dem Namen *App.jsx*. Aufgabe der Anwendung ist die Ausgabe einer Liste von Kommentaren. Im ersten Schritt ist hierzu der Quellcode aus [Listing 3](#) in die Datei *App.jsx* zu übernehmen:

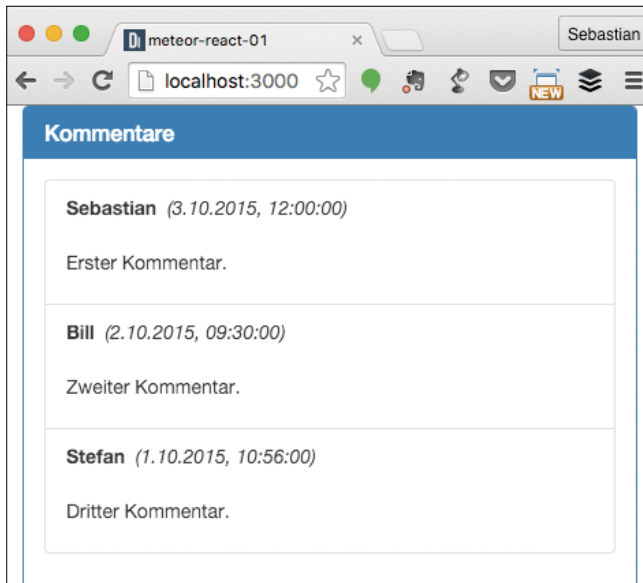
Die App-Komponente wird über den Aufruf der Methode *React.createClass* erstellt. An diesen Methodenaufruf wird ein Objekt übergeben, das eine Auflistung von Methoden enthält. Innerhalb dieser Methoden werden sowohl die Logik der Komponente als auch der Template-Code hinterlegt.

Listing 3: Die App-Hauptkomponente implementieren

```
App = React.createClass({
  getComments() {
    return [
      { _id: 1, user: "Sebastian", comment: "Erster
        Kommentar.", createdAt: new Date("2015-10-
        03T12:00:00") },
      { _id: 2, user: "Bill", comment: "Zweiter
        Kommentar.", createdAt: new Date("2015-10-
        02T09:30:00") },
      { _id: 3, user: "Stefan", comment: "Dritter
        Kommentar.", createdAt: new Date("2015-10-
        01T10:56:00") }
    ];
  },
  renderComments() {
    return this.getComments().map((comment) => {
      return <Comment key={comment._id}
        comment={comment} />;
    });
  },
  render() {
    return (
      <div className="container">
        <div className="panel panel-primary">
          <div className="panel-heading">
            <h3 className="panel-title">
              Kommentare</h3>
            </div>
            <div className="panel-body">
              <ul className="list-group">
                {this.renderComments()}
              </ul>
            </div>
          </div>
        </div>
      </div>
    );
  }
});
```

Listing 2: meteor-react-01.jsx

```
if (Meteor.isClient){
  Meteor.startup(function(){
    React.render(<App />,
      document.getElementById("render-target"));
  });
}
```

Beispielanwendung zur Ausgaben von Kommentaren (Bild 3)

Im vorliegenden Beispiel enthält die App-Komponente die drei Methoden *getComments*, *renderComments* und *render*.

Die *render*-Methode ist Bestandteil einer jeden React-Komponente und liefert den Template-Code für die HTML-Ausgabe der Komponente zurück. Durch die JSX-Syntax lassen sich hierzu HTML-Elemente direkt in den Quellcode aufnehmen. Zur Gestaltung des Layouts nutzen wir im aktuellen Beispiel einige Klassen des Bootstrap-Frameworks.

React und CSS-Klassen

Zu beachten ist hierbei, dass das HTML-Attribut *class* gegen den Attributnamen *className* ausgetauscht werden muss, damit React die CSS-Klassen anwendet. Zur Ausgabe einzelner Kommentare greifen wir auf die Komponenten-Methode *renderComments* zurück:

```
{this.renderComments()}
```

Mit diesem Ausdrucks-Block wird die Methode aufgerufen und der Rückgabewert an dieser Stelle in die HTML-Ausgabe eingefügt.

Die *renderComments*-Methode greift wiederum auf die *getComments*-Methode der App-Komponente zurück, um die Datensätze zur Ausgabe zu erhalten. Die *getComments*-Methode liefert hierzu ein Array bestehend aus drei Beispielsätzen. Für den ersten Schritt nutzen wir diese einfache Variante und hinterlegen die Beispielsätze direkt im Quellcode. Im nächsten Schritt werden wir die in Meteor standardmäßig vorhandene MongoDB-Datenbank nutzen, um die Datenhaltung auszulagern.

Über den Aufruf der *map*-Methode sorgen wir innerhalb von *renderComments* schließlich dafür, dass die Elemente des Kommentar-Arrays der Reihe nach durchlaufen werden und für jedes Element die Ausgabe erzeugt wird. Für die Ausgabe eines einzelnen Kommentars übergeben wir den Datensatz an den Aufruf der zusätzlichen React-Komponente *Com-*

ment. Auch bei diesem Schritt kommt die JSX-Schreibweise zum Einsatz:

```
<Comment key={comment._id} comment={comment} />
```

Beim Aufruf der Komponente werden die beiden Parameter *key* und *comment* übergeben. Dem Parameter *key* wird die ID des aktuellen Datensatzes zugewiesen. Hierzu kommt erneut die Schreibweise zum Einbetten eines Ausdrucks zur Anwendung:

```
{{Ausdruck}}
```

Der Parameter *comment* nimmt das aktuelle Kommentarobjekt auf, sodass innerhalb der *Comment*-Komponente sämtliche Eigenschaften des Datensatzes für die Ausgabe zur Verfügung stehen.

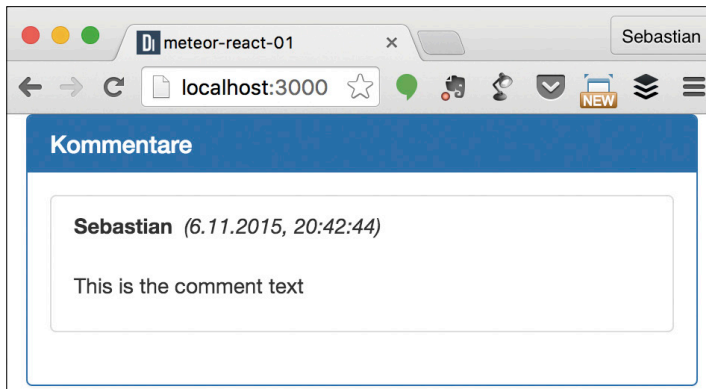
Bevor wir das Ergebnis der Applikation im Browser betrachten können, muss natürlich noch die fehlende Implementierung der *Comment*-Komponente ergänzt werden. Hierzu legen wir im Projektverzeichnis die neue Datei *Comment.jsx* an und fügen den Quellcode aus Listing 4 ein.

Der Quellcode zeigt, dass die React-Komponente in diesem Fall aus dem *propTypes*-Objekt und der *render*-Methode besteht. Das *propTypes*-Objekt enthält die Eigenschaft *comment* und legt über das Zuweisen von *React.PropTypes.object.isRequired* fest, dass dieses Attribut beim Aufruf der Komponente einen Wert enthalten muss.

Über die *render*-Methode wird die HTML-Ausgabe eines einzelnen Kommentardatensatzes erzeugt. Die Ausgabe ►

Listing 4: Comment-Komponente in App.jsx

```
Comment = React.createClass({
  propTypes: {
    comment: React.PropTypes.object.isRequired
  },
  render() {
    return (
      <li className="list-group-item">
        <strong>
          {this.props.comment.user} &nbsp;
        </strong>
        <i>
          ({this.props.comment.createdAt.
            toLocaleString()})
        </i>
        <br/><br/>
        <p>
          {this.props.comment.comment}
        </p>
      </li>
    )
  }
})
```

Ausgabe nach dem Hinzufügen des Datensatzes zur MongoDB-Datenbank (Bild 4)

der Kommentare anzulegen. Meteor macht das Anlegen einer Collection besonders einfach:

```
Comments = new Mongo.Collection("comments");
```

Diese Anweisung ergänzen wir zu Beginn der Datei *meteor-react-01.jsx* und stellen damit sicher, dass die Collection angelegt wird und sowohl client- als auch serverseitig zur Verfügung steht.

Damit der Zugriff auf die Collection auch aus der React-Komponente funktioniert, sind im Folgenden weitere vorbereitende Schritte notwendig. Innerhalb der App-Komponente nehmen wir das React-Mixin *ReactMeteorData* auf:

```
mixins: [ReactMeteorData]
```

Das Mixin sorgt dafür, dass der Zugriff auf Meteor-Collections innerhalb der Methode *getMeteorData* erfolgen kann. Der zweite notwendige Schritt ist daher die Aufnahme der Methode *getMeteorData*:

```
getMeteorData() {
  return {
    comments: Comments.find({}).fetch()
  }
}
```

Die *getMeteorData*-Methode liefert ein Rückgabeobjekt, dessen Eigenschaften mit Collection-Inhalten gefüllt werden. In unserem Beispiel wird die Eigenschaft *comments* mit sämtlichen Datensätzen der Comments-Collection gefüllt. Durch diesen Schritt stehen die Kommentardatensätze der Datenbank innerhalb der App-Komponente zur Verfügung. Der Zugriff auf die *comments* Eigenschaft erfolgt anschließend über das *data*-Objekt:

```
this.data.comments
```

Entsprechend ist auch die bereits vorhandene Methode *renderComments* anzupassen. Den gesamten aktualisierten Quellcode der App-Komponente sehen Sie in Listing 5.

Wird die Anwendung jetzt im Browser aufgerufen, fällt als Erstes auf, dass die Kommentarbox leer bleibt und keine Datensätze ausgegeben werden. Dieses Verhalten lässt sich leicht erklären: Wir haben zwar eine Collection für die Aufnahme von Kommentardatensätzen aufgenommen und damit eine Anbindung an die MongoDB-Datenbank realisiert, jedoch keine Datensätze erstellt. Über die Konsole lässt sich dies aber schnell nachholen. Öffnen Sie ein zweites Terminalfenster (über das erste Terminalfenster wurde der Meteor-Server gestartet) und starten Sie die MongoDB-Konsole über folgenden Aufruf im Projektverzeichnis:

```
meteor mongo
```

Sie werden durch diesen Schritt direkt mit der Datenbankinstanz verbunden und können anschließend Befehle an die Datenbank senden. Zum Hinzufügen eines neuen Datensatzes führen Sie beispielsweise den folgenden Befehl auf der MongoDB-Konsole aus:

```
db.comments.insert({user: "Sebastian", comment: "This is the comment text", createdAt: new Date() });
```

Ein Blick auf die geöffnete Applikationsseite im Browser zeigt sofort das Ergebnis (Bild 4). Der neu hinzugefügte Datensatz wird unmittelbar ausgegeben, ohne dass ein erneutes Laden der Webseite notwendig wird.

Eingabeformular mit React

Das Hinzufügen eines neuen Kommentardatensatzes haben wir im vorangegangenen Abschnitt direkt über die MongoDB-Konsole vorgenommen. Hiermit konnten wir sicherstellen, dass die Datenbank gefüllt wird und der Zugriff auf ►

Listing 7: Einsatz der CreateComment-Komponente

```
render() {
  return (
    <div className="container">
      <div className="panel panel-primary">
        <div className="panel-heading">
          <h3 className="panel-title">Kommentare</h3>
        </div>
        <div className="panel-body">
          <ul className="list-group">
            {this.renderComments()}
          </ul>
          <hr />
          <CreateComment />
        </div>
      </div>
    </div>
  );
}
```

Datensätze über die Comments-Collections vollständig funktioniert. Natürlich soll auch der Benutzer die Möglichkeit haben, neue Kommentare zu verfassen. Eine entsprechende Eingabemöglichkeit werden wir im Folgenden in der Applikation ergänzen. Wie in React-Anwendungen üblich, wird auch diese funktionale Einheit der Anwendung als eigenständige Komponente umgesetzt.

Die neu zu erstellende Komponente *CreateComment* wird in der neuen Datei *CreateComment.jsx* implementiert und innerhalb der App-Komponente aufgerufen. Beginnen wir mit der Implementierung von *CreateComment*.

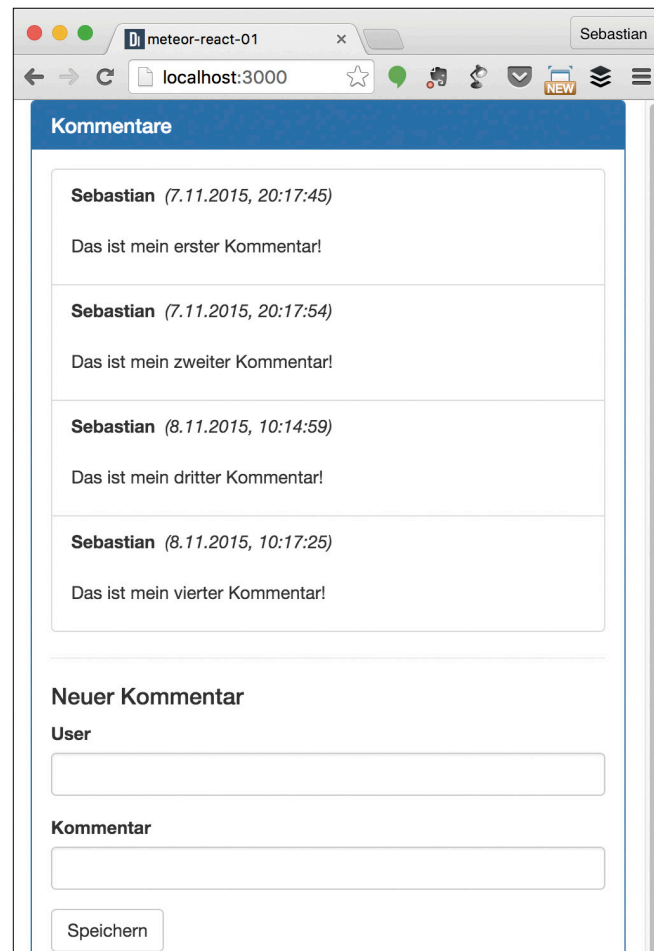
Der Quellcode zeigt, dass die neue React-Komponente aus zwei Methoden besteht: *handleSubmit* und *render* (Listing 6). Die *render*-Methode liefert den Template-Code zurück, mit dessen Hilfe die HTML-Ausgabe der Komponente erzeugt wird. Im Wesentlichen besteht die *CreateComment*-Komponente aus einem HTML-Formular mit zwei Texteingabeelementen und einem Submit-Button.

Anlegen eines neuen Kommentars

Zur Anlegen eines neuen Kommentars werden über die beiden `<input>`-Elemente der Benutzername und der Kommentartext eingegeben. Beide Steuerelemente erhalten über das Attribut `ref` einen eindeutigen Identifier, der anschließend zum Zugriff auf die Benutzereingaben in den Eingabefeldern genutzt wird. Über das Attribut `onSubmit` des `<form>`-Elements legen wir fest, dass die Ereignisbehandlung bei der Übermittlung des Formulars durch die Komponentenmethode *handleSubmit* erfolgen soll. Innerhalb dieser Methode sorgen wir mittels *event.preventDefault()*; zunächst dafür, dass die Standard-HTML-Formularverarbeitung unterbunden wird. Anschließend werden die Inhalte der `<input>`-Elemente eingelesen und in den beiden Variablen *userInputValue* und *commentInputValue* abgelegt. Der Zugriff auf die Eingabesteuerelemente erfolgt über den Aufruf der Methode *React.findDOMNode*. Als Parameter wird an den Methodenaufruf der `ref`-Identifier in der folgenden Form übergeben: *this.refs.[Identifier]*. Der Aufruf liefert eine Referenz auf das gewünschte HTML-Steuerelement zurück, sodass über die *value*-Eigenschaft auf den Wert zugegriffen werden kann.

Listing 8: Blaze-Template einbinden

```
AccountsWrapper = React.createClass({
  componentDidMount() {
    this.view = Blaze.render(Template.loginButtons,
      React.findDOMNode(this.refs.container));
  },
  componentWillUnmount() {
    Blaze.remove(this.view);
  },
  render() {
    return <span ref="container" />
  }
});
```



Eingabeformular zum Hinzufügen von Kommentaren (Bild 5)

Sind die Benutzereingaben ausgelesen, wird anschließend ein neuer Kommentardatensatz innerhalb der Comments-Collection erzeugt. Hierzu wird der Collection-Methode *insert* ein Kommentarobjekt übergeben. Meteor sorgt dann au-

Listing 9: AccountsWrapper-Komponente einbinden

```
render() {
  return (
    <div className="container">
      <div className="panel panel-primary">
        <div className="panel-heading">
          <h3 className="panel-title">Kommentare</h3>
        </div>
        <div className="panel-body">
          <ul className="list-group">
            {this.renderComments()}
          </ul>
          <hr />
          <AccountsWrapper />
          [...]
        </div>
      </div>
    </div>
  );
}
```


tomatisch dafür, dass nach dem Einfügen des Datensatzes in die Collection eine Synchronisation zwischen Client und Server stattfindet und der neue Datensatz auf dem Server anschließend automatisch in die Datenbank geschrieben wird.

Nach dem Hinzufügen des Kommentars werden zum Abschluss der Verarbeitung beide Eingabefelder des Formulars geleert.

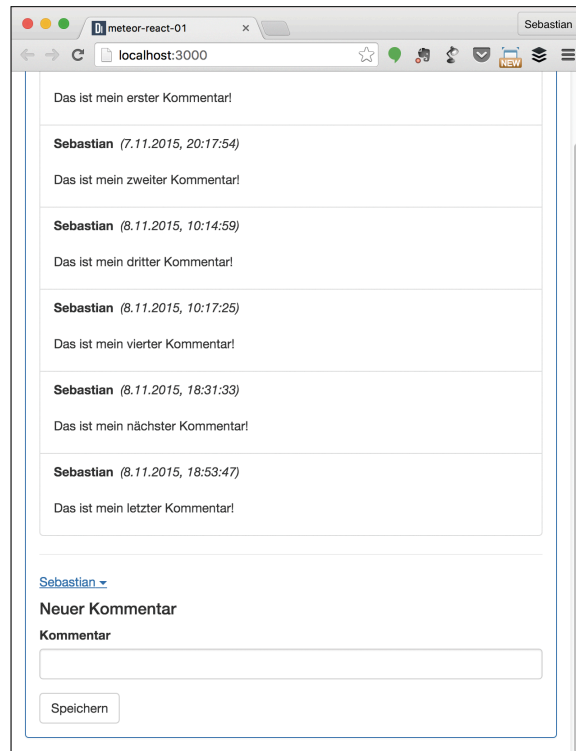
Damit die *CreateComment*-Komponente mit dem HTML-Eingabeformular für den Benutzer sichtbar wird, muss die Komponente abschließend dem Template-Code der App-Komponente hinzugefügt werden. Wechseln Sie hierzu in die Datei *App.jsx* und ändern Sie den Template-Code innerhalb der Methode *render* so ab, wie in Listing 7 dargestellt.

Die Erweiterung der Hauptkomponente führt schließlich dazu, dass das Eingabeformular für den Benutzer sichtbar wird und Kommentare direkt über die Oberfläche eingegeben werden können (Bild 5).

Benutzerauthentifizierung

Im vorigen Abschnitt haben wir die Kommentar-Anwendung mit einer Eingabemöglichkeit für den Benutzer versehen. Damit sichergestellt ist, dass die Eingabemöglichkeit für neue Kommentare nur angemeldeten Benutzern zur Verfügung steht, soll die Anwendung in diesem Abschnitt um die Funktion der Benutzerauthentifizierung erweitert werden.

Meteor macht die Erweiterung besonders einfach, da auf fertige Pakete zurückgegriffen werden kann. Wechseln Sie auf der Kommandozeile ins Projektverzeichnis und fügen Sie



Die Eingabe eines neuen Kommentars ist für einen angemeldeten Benutzer möglich (Bild 6)

die beiden Pakete *accounts-password* und *accounts-ui* ein:

```
meteor add accounts-password
accounts-ui
```

Das Paket *accounts-password* stellt alle notwendigen Funktionen bereit, die zur Benutzerauthentifizierung benötigt werden. Zusätzlich stellt das Paket *accounts-ui* das Blaze-Template *loginButtons* zur Verfügung.

Über dieses Template können UI-Elemente zur Benutzerauthentifizierung einfach in die bestehende Bedienoberfläche eingebunden werden. Damit das Blaze-Template aber auch innerhalb einer Meteor-React-Anwendung nutzbar ist, müssen wir zunächst einige Vorkehrungen treffen und das Blaze-Template in eine neue React-Komponente einbetten.

Für diese Wrapper-Komponente legen wir im Projektverzeichnis die neue Datei *AccountsWrapper.jsx* an und fügen den in Listing 8 gezeigten Quellcode ein. ▶

Listing 10: Konfiguration für einen Benutzernamen

```
Comments = new Mongo.Collection("comments");

if (Meteor.isClient){
  Accounts.ui.config({
    passwordSignupFields: "USERNAME_ONLY"
  });
  Meteor.startup(function(){
    React.render(<App />,
      document.getElementById("render-target"));
  });
}
```

Listing 11: getMeteorData()

```
getMeteorData() {
  return {
    comments: Comments.find({}).fetch(),
    currentUser: Meteor.user()
  }
},
```

Listing 12: Anzeige des Eingabeformulars

```
[...]
<hr />
<AccountsWrapper />
{this.data.currentUser ?
  <CreateComment /> :
  <div className="alert alert-info" role="alert">
    Bitte melden Sie sich an, um Kommentare
    einzugeben.
  </div>
}
[...]
```

Über den Aufruf von *React.createClass* wird eine React-Komponente erstellt, die aus den drei Methoden *componentDidMount*, *componentWillUnmount* und *render* besteht. Bei den genannten Methoden handelt es sich um sogenannte Lifecycle-Methoden der Komponente.

Zugriff auf DOM-Elemente

Die Methode *componentDidMount* wird zu Beginn des Lebenszyklus der Komponente aufgerufen, und zwar nach dem Ausführen der *render*-Methode. Hierdurch ist gewährleistet, dass innerhalb von *componentDidMount* bereits auf die DOM-Elemente der Komponente zugegriffen werden kann.

Für die *AccountsWrapper*-Komponente nutzen wir diese Tatsache und rufen die *Blaze.render*-Methode auf, um das *loginButtons*-Template auszugeben. Zusätzlich speichern wir eine Referenz des Rückgabewerts in der Eigenschaft *this.view*.

Mit dieser Referenz kann innerhalb der *componentWillUnmount*-Methode dafür gesorgt werden, dass die Ausgabe des *loginButtons*-Templates aus dem DOM entfernt wird, bevor die Komponente *AccountsWrapper* selbst aus dem DOM entfernt wird. *this.view* wird hierzu an den Aufruf von *Blaze.remove* übergeben.

Ist die Wrapper-Komponente erstellt, können wir das *loginButton*-Template über die Einbindung der Komponente in die Weboberfläche unserer Kommentaranwendung einbetten.



Ohne Benutzeranmeldung ist das Kommentar-Eingabeformular ausgeblendet (Bild 7)

Hierzu editieren wir das Template innerhalb der *render*-Methode der App-Komponente und nehmen das Element `<AccountsWrapper />` auf (Listing 9).

Als Ergebnis erhalten wir ein vollständiges Anmeldesystem, das es dem Benutzer erlaubt, ein Konto zu erstellen und sich anschließend mit den Benutzerdaten, bestehend aus E-Mail-Adresse und Passwort, an- und abzumelden.

Funktionen für angemeldete Benutzer

Die im vorangegangenen Schritt integrierte Funktion zur Benutzerauthentifizierung ermöglicht uns jetzt, Funktionen der Applikation nur für angemeldete Benutzer zugänglich zu machen. Die Eingabemöglichkeit für Kommentare soll nur dann angezeigt werden, wenn sich der Benutzer zuvor authentifiziert hat. Zusätzlich soll die Eingabe des Benutzernamens bei der Verfassung eines Kommentars entfallen. Stattdessen soll automatisch der Benutzername der Anmeldung mit dem Formulardatensatz gespeichert werden.

Zur Umsetzung der Anforderungen beginnen wir zunächst damit, die Anmeldung per E-Mail-Adresse gegen eine Anmeldung mit einem selbst gewählten Benutzernamen zu tauschen. Dazu rufen wir clientseitig die Methode *Accounts.ui.config* auf und übergeben ein Objekt mit der Eigenschaft *passwordSignupFields*. Als Wert wird der String *USERNAME_ONLY* zugewiesen. In Listing 10 sehen Sie den angepassten Quellcode der Datei *meteor-react-01.jsx*.

Listing 13: MainLayout.jsx

```

MainLayout = React.createClass({
  render() {
    return (
      <div>
        <nav className="navbar navbar-default navbar-fixed-top">
          <div className="container-fluid">
            <div className="navbar-header">
              <a className="navbar-brand" href="#">
                Meteor-React-01</a>
            </div>
            <ul className="nav navbar-nav">
              <li><a href="/">Kommentare</a></li>
              <li><a href="/moreinfos">Weitere
                Informationen</a></li>
            </ul>
          </div>
          </nav>
          <div className="container">
            {this.props.yield}
          </div>
          <nav className="navbar navbar-default navbar-fixed-bottom">
            <div className="container">
              <p className="navbar-text navbar-right">
                Powered by Meteor and React!</p>
            </div>
          </nav>
        </div>
      </div>
    );
  }
});

```

Ist die Authentifizierung des Benutzers über Username und Passwort erfolgt, liefert Meteor das aktuelle Benutzerkonto über `Meteor.user()` zurück. Damit auf die Kontoinformationen innerhalb der React-Komponenten zugegriffen werden kann, fügen wir diesen Wert dem Rückgabeobjekt der `getMeteorData`-Funktion innerhalb der Datei `App.jsx` hinzu (Listing 11). Anschließend lässt sich das Template der Hauptkomponente so ändern, dass das Eingabeformular nur dann eingeblendet wird, wenn ein Benutzer angemeldet ist und das `currentUser`-Objekt einen entsprechenden Wert liefert (Listing 12).

Über einen Anweisungsblock wird die Eigenschaft `this.data.currentUser` ausgewertet. Ist die Eigenschaft mit einem Wert belegt, wird die `CreateComment`-Komponente eingebunden. Im anderen Fall wird eine Meldung ausgegeben, dass zur Eingabe von Kommentaren eine Anmeldung erfolgen muss. Die beiden Zustände der Anwendung sind in Bild 6 und Bild 7 zu finden.

Routing

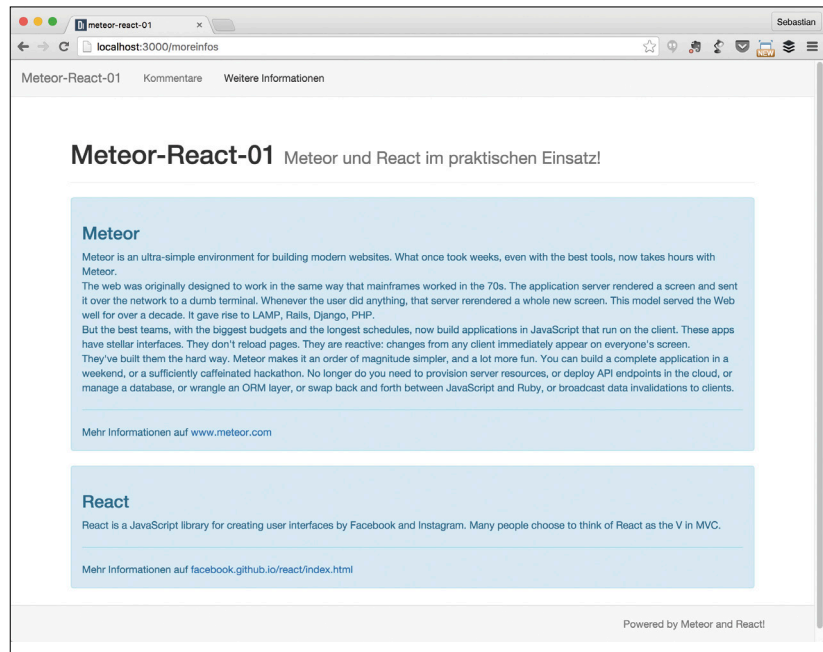
Bisher besteht unsere Anwendung lediglich aus einer Seite, sodass beim Aufruf des URL `http://localhost:3000` die einzige Seite der Anwendung aufgerufen und gerendert wird.

Oftmals wird jedoch der Bedarf bestehen, die Funktionalität der Anwendung auf mehrere Ansichten zu verteilen. Im Sinne einer Single-Page-Webanwendung wird die Anwendung zwar als ein HTML-Dokument geladen, je nach aufgerufenem URL werden jedoch verschiedene Ansichten der Seite generiert. Benötigte Daten werden in diesen Fällen dynamisch nachgeladen. Für die Generierung der verschiedenen Ansichten in Abhängigkeit vom aufgerufenen URL ist ein entsprechendes Routing erforderlich.

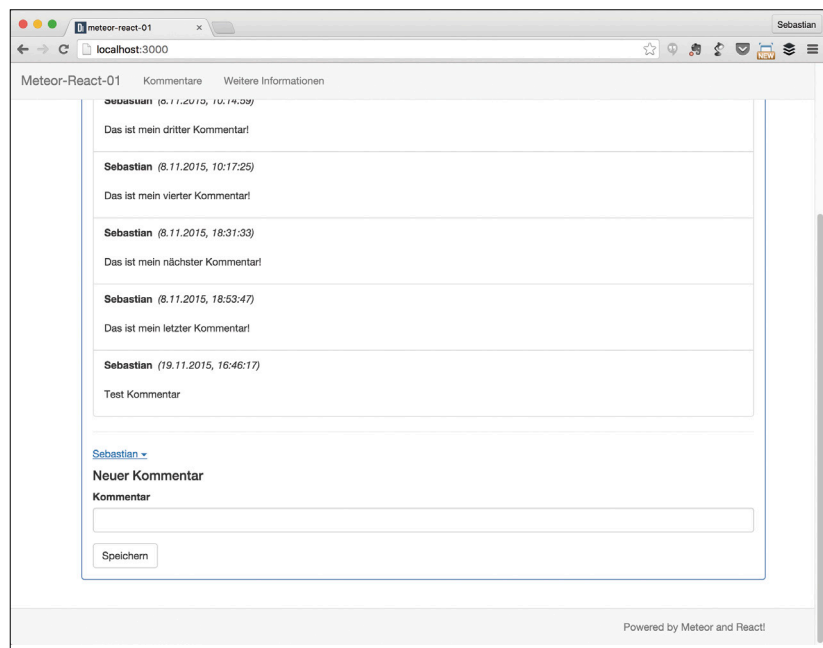
Im Standardumfang des Meteor-Frameworks ist keine Routing-Komponente erhalten. Es muss zu diesem Zweck daher auf ein zusätzliches Paket zurückgegriffen werden. Wir haben hierzu bei der Einrichtung des Projekts bereits die beiden Pakete `kadira:flow-router` und `kadira:react-layout` hinzugefügt und hierdurch FlowRouter installiert. FlowRouter zeichnet sich besonders dadurch aus, dass der Router modular aufgebaut ist und durch entsprechende Erweiterungen neben Blaze auch React- und AngularJS-Views rendern kann.

Layout-Komponente

Bevor wir FlowRouter für unsere Applikation konfigurieren und verschiedene URL-Routen mit React-Komponenten verbinden, gilt es zunächst die Hauptkomponente der Anwen-



Ausgabe der Anwendung nach Aufruf der Route `/moreinfos` (Bild 8)



Ausgabe der Kommentarseite nach dem Aufruf der Standardroute (Bild 9)

derung zu einer Layout-Komponente umzubauen. Eine Layout-Komponente enthält das generelle Layout der Applikation (zum Beispiel Header, Menüs und Fußzeile) und markiert den Bereich, in dem die Ausgabe der zur Route gehörenden Komponente ausgegeben wird. Legen Sie hierzu die neue Datei `MainLayout.jsx` im Projektverzeichnis an und fügen Sie den Code aus Listing 13 ein.

Die `MainLayout`-Komponente besteht lediglich aus der `render`-Methode, die den entsprechenden Template-Code zurückliefert. Das Layout setzt sich aus drei Bereichen zusammen. ►

Im ersten Bereich wird über den Einsatz des `<nav>`-Elements ein Menü eingebunden. Das Menü enthält zwei Menüelemente mit den Beschriftungen *Kommentare* und *Weitere Informationen*. Über den *Kommentare*-Link wird die Standardroute der Anwendung aufgerufen. In diesem Fall soll die zuvor erstellte Kommentarlite ausgegeben werden. Der Link *Weitere Informationen* führt zur Route `/moreinfos`. Diese Route werden wir in Kürze mit dem Aufruf und der Ausgabe einer weiteren React-Komponente verbinden.

Der zweite Bereich des Templates besteht aus einem `<div>`-Elemente, das den folgenden Ausdruck enthält:

```
{this.props.yield}
```

Hiermit kennzeichnen wir den Bereich, der für die Ausgabe der von der Route abhängigen React-Komponente genutzt werden soll.

Im dritten Bereich des Templates wird ein weiteres `<nav>`-Element eingebunden und über die Bootstrap-CCS-Klasse `navbar-fixed-bottom` zusätzlich festgelegt, dass dieses Element als Fußzeile im unteren Bereich der Anwendungsoberfläche angezeigt werden soll.

FlowRouter-Konfiguration

Die FlowRouter-Konfiguration erfolgt clientseitig und wird daher in den `if(Meteor.isClient) { }`-Block innerhalb der Datei `meteor-react-01.jsx` eingebunden (Listing 14).

Die Konfiguration einer URL-Route erfolgt über den Aufruf von `FlowRouter.route`. Die Methode erwartet beim Aufruf die Übergaben von zwei Parametern. An erster Stelle wird die Route als String übergeben. An zweiter Stelle erfolgt die Übergabe eines Konfigurationsobjekts mit der Eigenschaft `name` und der Methode `action`.

Der `name`-Eigenschaft weisen wir einen String zu, der als Identifier für die Route dient. Die Methode `action` wird von FlowRouter aufgerufen, sobald die Route im Browser angesprochen wird. Innerhalb dieser Methode fügen wir daher den Aufruf ein, der für das Rendern einer React-Komponente sorgt und das Ergebnis an die vorgemerkte Stelle im Template der `MainLayout`-Komponente einfügt:

```
ReactDOM.render(MainLayout, {yield: <App />});
```

Die Methode `ReactDOM.render` übernimmt diese Aufgabe und erhält im ersten Parameter die React-Layout-Komponente. In unserem Fall ist dies die zuvor implementierte Komponente `MainLayout`. Der zweite Parameter ist wiederum ein Objekt und beinhaltet die Eigenschaft `yield`. Der Eigenschaft wird hierbei die React-Komponente zugewiesen, die im Template der `MainLayout`-Komponente an die Stelle eingefügt werden soll, an der zuvor der `yield`-Rückgabewert als Platzhalter aufgenommen wurde.

Für die Zuweisung der Komponente an die `yield`-Eigenschaft des Übergabeobjekts nutzen wir ebenfalls die JSX-Schreibweise. Die Standard-Route der Anwendung verbinden wir hierdurch mit der App-Komponente. Die App-Komponente verliert daraufhin die Rolle als Hauptkomponente

Listing 14: FlowRouter-Konfiguration

```
FlowRouter.route('/', {
  name: 'main-app',
  action() {
    ReactDOM.render(MainLayout, {yield:
      <App />});
  }
});
FlowRouter.route('/moreinfos', {
  name: 'more-info',
  action() {
    ReactDOM.render(MainLayout, {yield:
      <MoreInfos />});
  }
});
```

der Anwendung. Diese Rolle nimmt jetzt die `MainLayout`-Komponente ein.

Der zweite Aufruf von `FlowRouter.route` verbindet die Route `/moreinfos` mit der `MoreInfos`-Komponente. Diese Komponente liefert die zweite Ansicht unserer Applikation und gibt einige Informationen zu den eingesetzten Frameworks Meteor und React aus (Bild 8).

Ein Klick auf den Menüpunkt *Kommentare* bringt Sie zurück zur Kommentar-Ansicht, die jetzt ebenfalls in das Layout der `MainLayout`-Komponente eingebettet ist (Bild 9).

Fazit

Das Beispiel dieses Artikels hat gezeigt, dass das Zusammenspiel von Meteor und React nahtlos funktioniert. React nimmt hierbei die Rolle der View-Engine ein und ersetzt damit optional die Meteor-eigene View-Engine Blaze. React bietet zusätzlich den Vorteil einer komponentenorientierten Entwicklung. Durch den Einsatz von JSX lassen sich React-Komponenten einfach und schnell implementieren. Der notwendige HTML-Template-Code kann direkt in den JavaScript-Code eingebettet werden.

Mit FlowRouter steht außerdem ein Meteor-Paket bereit, das das clientseitige Routing auch für React-Komponenten erlaubt.

Den vollständigen Quellcode der Beispielanwendung des Artikels finden Sie unter <https://github.com/seeschweiler/meteor-react-01> zum Download. ■



Sebastian Eschweiler

hat Wirtschaftsinformatik studiert und ist als IT-Projektleiter tätig. Sein Wissen im Bereich Webentwicklung und mobile Technologien gibt er als Autor von Fachbeiträgen und Büchern weiter.

<http://meanstack.de>

Jetzt Aussteller werden!



developer-week.de



DeveloperWeek

Aussteller & Sponsoren 2015/2016:



Veranstalter:



Neue
Mediengesellschaft
Ulm mbH

Präsentiert von:



web & mobile
DEVELOPER

RESPONSIVE HTML-MAILS

Schönere Newsletter

Bei der Codierung von HTML-Mails gelten andere Regeln als bei Webseiten.

Zurück in die Vergangenheit – so kommt es einem vor, wenn man sich den Code von HTML-Mails ansieht: tief verschachtelte HTML-Tabellen, Inline-Styles, veraltete HTML-Elemente wie *center* und Attribute wie *align*. All diese Techniken, die auf Webseiten schon längst nicht mehr eingesetzt werden, sind bei HTML-Mails üblich und erforderlich. Trotzdem lässt sich diese scheinbar vorsintflutliche Herangehensweise durchaus mit einem responsiven Webdesign kombinieren.

Bevor wir dazu kommen, wie sich ein responsives Design bei HTML-Mails umsetzen lässt, sehen wir uns zunächst die Besonderheiten von HTML-Mails an. Dieses Wissen ist die entscheidende Basis, um die Möglichkeiten und Grenzen von responsivem Design bei E-Mails auszuloten.

Mail-Clients

Bei Webseiten haben wir es mit einer Handvoll Browser zu tun, mit denen Webseiten betrachtet werden. Und berücksichtigt man zudem, dass ein Teil der Browser dieselbe Rendering-Engine hat – etwa Chrome und Opera – so reduziert sich die Anzahl der möglichen Ausgabe-Tools noch einmal. Bei E-Mails stellt sich die Situation völlig anders dar. Die Adressaten können die Mail beziehungsweise den Newsletter auf ganz verschiedenen Clients ansehen. Man unterscheidet Webinterfaces, Desktop-Clients und mobile Mail-Clients.

Zu den Webinterfaces gehören Outlook.com, Yahoo-Mail, Gmail, GMX oder Web.de. Das Besondere dabei ist, dass diese Webclients Informationen herausfiltern können – zum Beispiel aus Sicherheitsgründen. Außerdem kann es eine Rolle spielen, mit welchem Browser der Benutzer das Webinterface aufruft. Das heißt, mögliche Beschränkungen durch den Browser gelten zusätzlich, diese sind aber im Allgemeinen nicht so gravierend wie die Modifizierungen durch das Webinterface selbst.

Dann gibt es auch Desktop-Clients, die man für E-Mails nutzen kann. Am weitesten verbreitet – und leider auch am problematischsten – ist Outlook. Daneben kommen Apple Mail, Mozilla Thunderbird und weitere zum Einsatz. Bei den Desktop-Clients kann ebenfalls die Version eine Rolle spielen wie beispielsweise bei Outlook: Outlook vor 2007 verwendet den Internet Explorer 6 als Rendering Engine, danach



E-Mail-Client-Verwendung laut der Statistik bei <http://emailclientmarketshare.com> (Bild 1)

kommt Word zum Einsatz – und damit unterscheiden sich die Versionen grundlegend in der Interpretation und Darstellung von HTML/CSS.

Schließlich gibt es noch mobile Mail-Clients wie etwa Gmail Inbox, iOS Mail, Mailbox und weitere. Im Allgemeinen ist bei Smartphones nicht das Gerät beziehungsweise das Betriebssystem entscheidend, auf dem eine Mail betrachtet wird, sondern die App, mit der die Mail gelesen wird.

Auch wenn sie den Namen teilen, hat der Webclient Outlook.com wenig mit dem Desktop-Client Outlook oder auch mit der Outlook-App gemein, was die Darstellung von HTML-Mails betrifft.

Üblicherweise setzt ein Benutzer verschiedene Clients ein. Nicht ungewöhnlich ist etwa, dass ein Benutzer ein Desktop-Mail-Programm hat und daneben Apps fürs Smartphone und fürs Tablet. Und zwischendurch verwendet er auch mal Webclients.

Um festzustellen, welche Clients Ihre Adressaten nutzen, ziehen Sie am besten eigene Statistiken heran, wie sie von Newsletter-Service-Anbietern zur Verfügung gestellt werden. Ansonsten gibt es auch öffentliche Statistiken, etwa die von Litmus (Bild 1). Auch wenn solche Statistiken nicht direkt auf den Nutzerkreis Ihrer E-Mail übertragbar sind, zeigen sie doch deutlich, dass beispielsweise Gmail und Outlook, die beide bei der Darstellung von HTML-Mails problematisch sind, nicht ignoriert werden können.

Besonderheiten von Outlook

Die verschiedenen Versionen von Outlook unterscheiden sich wie erwähnt darin, womit die E-Mails gerendert werden. Bei Outlook vor Version 2007 war es der Internet Explorer 6, danach ist es Word.

Natürlich wird kein Webdesigner Lobeshymnen auf den Internet Explorer 6 anstimmen, trotzdem eignet er sich für die Darstellung von HTML/CSS wesentlich besser als Word. Dass im Jahr 2015 für HTML-Mails noch Tabellen verwendet werden müssen, liegt daran, dass Outlook damit am besten umgehen kann.

Konditionale Kommentare, die bei Webseiten allmählich aus der Mode kommen, sind beim Umgang mit Outlook-Besonderheiten sehr wichtig. Outlook sprechen Sie über *mso*

an. Im folgenden Beispiel werden besondere CSS-Angaben für Outlook ≥ 9 bereitgestellt:

```
<!--[if gte mso 9]>
  <style type="text/css">
    /* für Outlook größer als Version 9*/
  </style>
<![endif]-->
```

Häufiger als für speziellen CSS-Code werden diese konditionalen Kommentare jedoch genutzt, um im HTML-Teil der Mail zusätzliche Tabellen zu definieren. Das ist beispielsweise sinnvoll, um die fehlende Unterstützung für die Eigenschaft *max-width* auszugleichen, wie im folgenden Beispiel:

```
<!--[if gte mso 9]>
<table align="center" border="0" cellspacing="0"
cellpadding="0" width="600" style="width:600px;">
<tr>
<td align="center" valign="top" width="600"
style="width:600px;">
<![endif]-->
```

Diese Tabelle muss dann später auch wieder geschlossen werden.

In den konditionalen Kommentaren geben Sie die Outlook-Version als Nummer an. Dabei gelten folgende Zuordnungen:

- Outlook 2000 – Version 9
- Outlook 2002 – Version 10
- Outlook 2003 – Version 11
- Outlook 2007 – Version 12
- Outlook 2010 – Version 14
- Outlook 2013 – Version 15

Kommen wir zum nächsten wichtigen und zugleich problematischem Mail-Client: Gmail.

Gmail

Die Besonderheit von Gmail ist, dass es vor der Darstellung kategorisch Code aus den HTML-Mails herausfiltert. Dazu gehören erst einmal JavaScript und Flash. Für die Entwicklung von Mails ist aber entscheidend, dass auch das *head*-Element (mit allem Inhalt) und *<body>* sowie *</body>* ent-

Listing 1: Inlining CSS

```
<html>
<head>
  <style type="text/css">
    table {
      border-spacing: 0;
      border-collapse: collapse;
    }
    table.body {
      height: 100%;
      width: 100%;
    }
    table, tr, td {
      padding: 0;
      vertical-align: top;
      text-align: left;
    }
  </style>
</head>
<body>
  <table class="body">
    <!-- restliche Struktur gekürzt -->
  </table>
</body>
</html>
```

fernt wird. Außerdem werden alle *style*-Elemente gelöscht. Angaben, die innerhalb von *<style>* und *</style>* geschrieben werden, haben damit keinerlei Auswirkung auf die Darstellung. Deswegen können Formatierungen für Gmail nur durch direkt bei den HTML-Elementen stehende Inline-Styles angegeben werden.

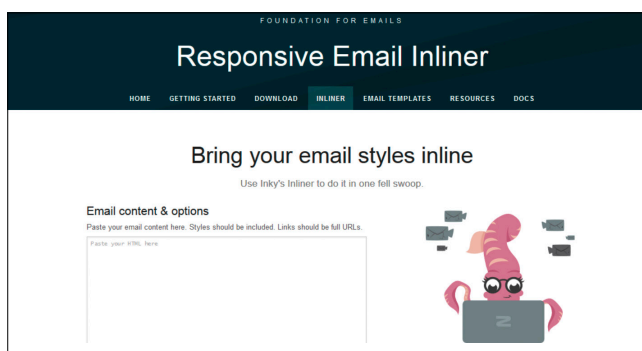
Inlining CSS

Externe Stylesheets über das *link*-Element funktionieren in den meisten Mail-Clients nicht. Und da es Clients wie Gmail gibt, die auch *style*-Elemente oder *head*-Elemente herausfiltern, müssen Sie Formatierungen immer direkt in den Elementen unterbringen, das heißt inlinen. Dieses Grundprinzip zeigt **Listing 1**. Durch das Inlinen werden alle Angaben direkt bei *style=* in *table* geschrieben:

```
<table class="body" style="border-spacing: 0;
border-collapse: collapse; height: 100%; width: 100%;
vertical-align: top; text-align: left;
padding: 0;">...</table>
```

In echten Beispielen ist durch das Inlinen wesentlich mehr Code betroffen. Das heißt: Der Vorgang des Inlinens ist äußerst aufwendig und das Ergebnis sehr unübersichtlich.

Deswegen empfiehlt es sich, für das Inlinen Tools zu verwenden, von denen es mehrere gibt (**Bild 2**). Neben webbasierten Tools wie denen von MailChimp oder Foundation existieren Inliner, die Sie mit PHP, Node.js oder ähnlichen ►



Inliner von Foundation (**Bild 2**)

Tools verwenden können. Beachten Sie dabei, dass sich die Inliner mitunter im Detail unterscheiden können.

Wichtig ist, dass Sie die Ursprungsversion Ihrer HTML-Mail, bei der die CSS-Angaben im *style*-Element innerhalb des Kopfes stehen, weiterhin erhalten. Denn Anpassungen an einer Datei mit Inline-CSS durchzuführen ist mehr als mühsam und wäre ein unnötiges Erschwernis.

Üblicherweise sind bei vorgefertigten E-Mail-Templates die CSS-Angaben auch noch im Kopf untergebracht, das heißt dort, wo Sie sie gut verändern und anpassen können. Das Inlining lassen Sie danach durchführen.

Grenzen des Inlinings

Es gibt allerdings Fälle, in denen der CSS-Code nicht den einzelnen Elementen zugeordnet werden kann, weil das prinzipiell unmöglich ist. Beispielsweise bei den verschiedenen Linkzuständen *a:visited*, *a:focus* und *a:hover*. Diese können Sie nicht bei den Links selbst angeben und sie müssen deshalb weiterhin innerhalb von *<style>* und *</style>* bleiben. Das gilt auch für alle anderen Pseudoklassen/-elemente.

Der Inliner von Foundation fügt für diese Pseudoklassen einen *<style>*-Bereich nach dem *body* ein, in dem die Definitionen der Linkzustände stehen:

```
</head>
<body>
  <style type="text/css">
    a:hover {
      color: #2795b6 !important;
```

Listing 2: HTML-Mail-Grundstruktur

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible"
    content="IE=edge">
  <meta name="viewport" content="width=device-width,
    initial-scale=1">
  <title>Titel</title>
</head>
<body>
  <center>
    <table align="center" border="0" cellpadding="0"
      cellspacing="0" height="100%" width="100%" >
      <tr>
        <td align="center" valign="top">
          <!--Inhalte -->
        </td>
      </tr>
    </table>
  </center>
</body>
</html>
```

```
}
a:active {
  color: #2795b6 !important;
}
a:visited {
  color: #2ba6cb !important;
}
</style>
<table><!-- hier die restliche Seite -->
```

Der Inliner von MailChimp hingegen belässt das *style*-Element mit allen Inhalten zusätzlich im *head*.

Der andere Fall, in dem das Inlining der CSS-Angaben nicht funktioniert, sind *@media*-Regeln. Denn die dort angegebenen Formatierungen gelten nur in bestimmten Kontexten, beispielsweise bei bestimmten Maximalbreiten des Viewports. Deswegen müssen auch die *@media*-Regeln innerhalb von *<style>* und *</style>* stehen bleiben. Das bedeutet eine deutliche Einschränkung für das responsive Designen: MailClients, die das *style*-Element herausfiltern, können *@media*-Regeln nicht interpretieren.

Die auffälligste Besonderheit beim Umgang mit CSS in Mails ist das hier beschriebene Inlining. Aber auch bei der Wahl der CSS-Eigenschaften gibt es einiges zu beachten: Prinzipiell funktioniert einfaches CSS 2.1 besser als fortgeschrittene Angaben. Außerdem empfiehlt es sich, auf verkürzte Eigenschaften wie *font* zu verzichten und stattdessen die Einzeleigenschaften wie *font-family*, *font-style* et cetera zu schreiben. Bei Farbangaben sollten Sie nur sechsstellige Angaben wie *#FF00CC* verwenden und auf mögliche Verkürzungen, also im Beispiel *#F0C*, verzichten. Für die CSS-Unterstützung einzelner Mail-Clients und ihre Besonderheiten gibt es verschiedene Kompatibilitätstabellen, die im Kasten aufgeführt sind. Denn es gibt leider Ungereimtheiten – beispielsweise, dass bei Outlook.com die Eigenschaft *margin* nicht unterstützt wird.

HTML-Grundgerüst

Noch eine weitere Einschränkung betrifft CSS: Wo es nämlich möglich ist, eine Formatierung statt mit CSS über HTML durchzuführen, ist diese Lösung vorzuziehen. Deswegen sind längst totgesagte Attribute wie *align* in HTML-Mails nicht verpönt, sondern empfohlen. Und am solidesten ist es, auch heute noch für das grundlegende Layout eine HTML-Tabelle einzusetzen. Dabei übernimmt diese HTML-Tabelle im Zweifelsfall zudem die Rolle des *body*-Elements – da das *body*-Element ja teilweise herausgefiltert wird. Das *center*-Element sorgt für eine zusätzliche Zentrierung. Den grundlegenden Aufbau zeigt Listing 2. 100 Prozent Höhe und 100 Prozent Breite bei *table* sorgen dafür, dass die Tabelle den gesamten Platz einnimmt. Innerhalb der inneren Zelle werden dann weitere Tabellen für die einzelnen Bereiche/Spalten angesetzt. Auch für kleinere Komponenten wie Buttons lässt sich das *table*-Element einsetzen. Auf die klassischen *div*-Elemente oder neuere Elemente wie *article/section* et cetera sollten Sie hingegen verzichten. Das bedeutet, dass Layouts für HTML-Mails aus tief verschachtelten Tabellen bestehen.

Feature	Outlook 2007/10/13	Apple Mail	iOS	Android 4	Gmail App	Windows Phone 8.1	Yahoo! Mail	Gmail	Outlook.com	Office 365/OWA	AOL Mail	YMail App	Outlook iOS
Kinetic support ?													
<style>													
@media													

Unterstützung für @media gemäß Freshinbox (Bild 3)

Bei Webseiten gilt das responsive Design als State of the Art, aber auch bei HTML-Mails hat es große Vorteile. Da mehr als die Hälfte aller E-Mails auf mobilen Geräten geöffnet werden, sollte sichergestellt sein, dass die Mails gerade auch auf den mobilen Geräten gut funktionieren.

Eine Untersuchung von Litmus zeigt 130 Prozent höhere Klickraten bei einer responsiven Mail im Gegensatz zur nicht-responsiven Variante. Auch wenn die Zahlen natürlich nicht universell übertragbar sind und je nach konkretem Newsletter recht unterschiedlich ausfallen können, zeigen sie doch, dass es sich lohnt, E-Mails responsiv zu gestalten.

Media Queries in E-Mail-Clients

Die entscheidende Komponente des responsiven Webdesigns sind bei Webseiten die @media-Regeln. Browser, die diese nicht unterstützen, wie der Internet Explorer 8, können Sie im Web getrost ignorieren oder, sofern notwendig, über JavaScript – beispielsweise mit *respond.js* – nachbessern.

Bei HTML-Mails haben wir jedoch eine vollkommen andere Situation und können die Unterstützung für @media-Regeln nicht voraussetzen. Dafür sind zwei Gründe verantwortlich: Zum einen gibt es Clients, die diese einfach nicht unterstützen – zum Beispiel Outlook, das wie erwähnt bei der Darstellung von HTML-Mails auf Word zurückgreift. Und zum anderen funktionieren @media-Regeln nur, wenn sie in einem eigenen *style*-Abschnitt eingefügt werden, und diese werden bei Gmail automatisch herausgefiltert. Das heißt, dass @media-Regeln bei Outlook und Gmail wirkungslos bleiben. Ein Nachbessern per JavaScript scheidet ebenfalls aus.

Aber natürlich gibt es auch Mail-Clients, die @media-Regeln unterstützen (Bild 3), wie mobile Webclients oder Desktop-Varianten wie Apple Mail oder Thunderbird.

Es sind also Lösungen gefragt, die im Prinzip auch auf Clients funktionieren, die @media-Regeln nicht interpretieren, das heißt, dass eine grundlegende Anpassung auch ohne @media-Regeln stattfindet. Für fortgeschrittene Clients können Sie dann detailliertere Anpassungen mit Hilfe von @media-Queries durchführen.

Neben all den im Vergleich zu Webseiten größeren Schwierigkeiten für eine responsive Gestaltung gibt es aber auch einen Vorteil von Mails: Normalerweise sind HTML-Mails inhaltlich weniger komplex als Webseiten, da sie sich auf ein Thema beziehungsweise ein Anliegen konzentrieren. Das

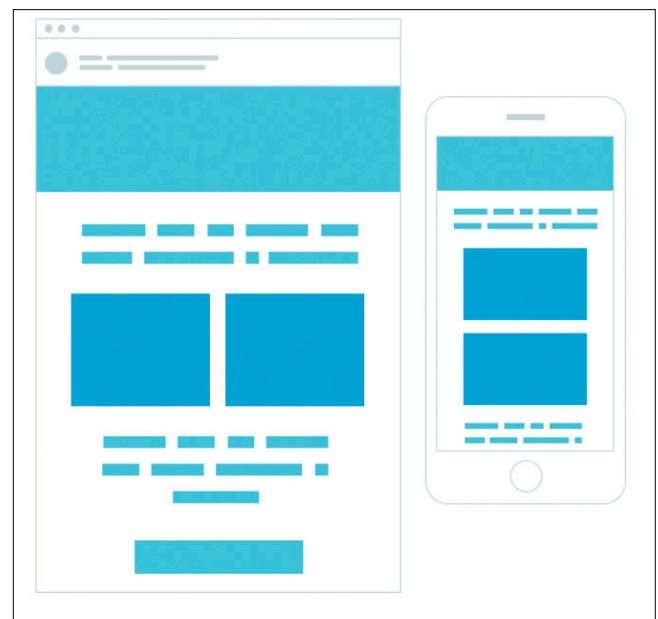
Problem der zu vielen Inhalte, die sich nur schlecht auf kleinen Screens unterbringen lassen, tritt hier seltener auf.

Anpassungen ohne @media

Einfache Layoutanpassungen sind auch ohne Media Queries möglich: Sinnvoll ist es etwa, im Fall von ausreichend vorhandenem Platz die Inhalte mehrspaltig darzustellen, ansonsten einspaltig. Bei einer Webseite wäre dafür *float* das Mittel der Wahl: Wenn Sie mehrere Elemente nebeneinander floaten, werden so viele nebeneinander dargestellt, wie Platz haben, die übrigen kommen in die nächste Zeile.

Leider funktioniert jedoch die Eigenschaft *float* bei HTML-Mails nicht clientübergreifend. Aber es gibt verschiedene andere Ansätze. Gehen wir von einem zweispaltigen Layout aus, das bei Bedarf einspaltig dargestellt werden soll.

Der von Fabio Carneiro von MailChimp beschriebene Ansatz sieht folgendermaßen aus: In der *body*-Tabelle gibt es eine weitere Tabelle mit flüssigen Ausmaßen, das heißt 100 Prozent Breite. In diese Tabelle werden zwei weitere Tabellen mit 100 Prozent Breite gesetzt. Diese werden dann mit *align="left"* und *align="right"* nebeneinander angeordnet. ►



Einfache Layoutanpassungen gelingen auch ohne @media – Beispiel von Campaignmonitor (Bild 4)

align ermöglicht dabei eine Änderung der Quellcodereihenfolge: Die Tabelle mit *align="right"* wird als zweite Spalte rechts angezeigt, auch wenn sie im Quellcode zuerst dasteht. Mit CSS werden vernünftige Maximalbreiten definiert. Damit befinden sich die inneren Tabellen bei wenig verfügbarem Platz untereinander. Ist dagegen mehr Platz vorhanden, erhalten wir eine zweispaltige Darstellung. Zuerst kommt also die äußere Tabelle:

```
<table align="center" border="0" cellpadding="0"
cellspacing="0" width="100%" class="flexContainer">
  <tr>
    <td valign="top">
      <!-- Weitere Inhalte -->
    </td>
  </tr>
</table>
```

In dieser Tabelle werden zwei weitere Tabellen ergänzt und über *align="left"* und *align="right"* nebeneinander platziert.

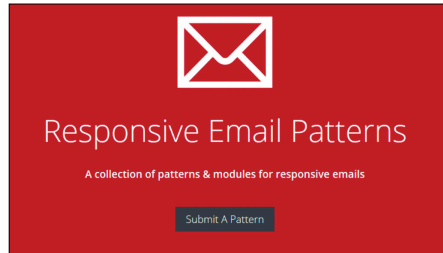
```
<table align="right" border="0" cellpadding="0"
cellspacing="0" width="100%" class="rightColumn">
  <tr>
    <td align="left" valign="top">
      Rechte Spalte
    </td>
  </tr>
</table>
<table align="left" border="0" cellpadding="0"
cellspacing="0" width="100%" class="rightColumn">
  <tr>
    <td align="left" valign="top">
      Linke Spalte
    </td>
  </tr>
</table>
```

Die maximalen Breiten werden per CSS festgelegt:

```
.flexContainer{max-width:640px;}
.leftColumn, .rightColumn{max-width:320px;}
```

Da Outlook *max-width* nicht interpretiert, wird für diesen Client die Breite über eine zusätzlich verschachtelte Tabelle simuliert – glücklicherweise gibt es für solche Zwecke die konditionalen Kommentare.

Einen anderen Ansatz stellt Ros Hodgekiss bei Campaign Monitor vor: Problematisch an der Anordnung nebeneinander über *align: left* und *align: right* ist, dass dadurch Spalten mit weniger als 100 Prozent Breite bei der einspaltigen Darstellung leicht versetzt angezeigt werden (die eine rechts, die andere links). Die bei Campaign Monitor vorgestellte Alter-



Patterns für Layouts und kleinere Komponenten von HTML-Mails (Bild 5)

native besteht im Einsatz von *display:inline-block* in Kombination mit *text-align:center*. (Bild 4)

Mit einer der vorgestellten Methoden können Sie also, ohne auf Media Queries zurückzugreifen, grundlegende Layoutanpassungen durchführen. Es ist allerdings kein Feintuning möglich; Sie können nicht Anpassungen bei der Schriftgröße durchführen oder beispielsweise speziell bei bestimmten Viewportgrößen einzelne Angaben

ein- oder ausblenden. Solche Anpassungen lassen sich aber über Media Queries für die fortgeschritteneren Mail-Clients vornehmen.

RWD mit @media

Die *@media*-Regeln können Sie ganz normal in einen *style*-Bereich im Kopf oder an den Anfang von *body* schreiben. Eine Besonderheit ergibt sich jedoch durch die Spezifität. Da die Standardangaben als Inline-Stile in den Elementen stehen, würden sie sich immer gegenüber anderen Angaben durchsetzen.

Links zum Thema

- Besonderheiten einzelner Mail-Clients
<http://templates.mailchimp.com/development/css/outlook-conditional-css>
- Inliner
<http://foundation.zurb.com/emails/inliner.html>
<http://templates.mailchimp.com/resources/inline-css>
- Verbesserte Klickraten bei responsiven Mails
<https://litmus.com/blog/responsive-design-ab-testing-leads-to-a-130-increase-in-clicks>
- Sich anpassende Mails ohne *@media*-Angaben
<https://litmus.com/community/code/499-responsive-email-possible-with-no-body-head-tags>
<https://www.campaignmonitor.com/blog/email-marketing/2014/07/creating-a-centred-responsive-design-without-media-queries>
- Vorlagen, Frameworks, Assistenten
<http://responsiveemailpatterns.com>
<http://foundation.zurb.com/emails.html>
<http://mailchimp.com>
<https://github.com/mailchimp/Email-Blueprints>
- Cross-Client-Tests
<https://litmus.com/email-testing>
- CSS-Unterstützung in Mail-Clients
<https://www.campaignmonitor.com/css>
<http://freshinbox.com/resources/css.php>
<http://templates.mailchimp.com/resources/email-client-css-support>

Gehen wir davon aus, dass im *table*-Element Folgendes angegeben ist:

```
<table style="width: 320px"
class="linkespalte">
```

Innerhalb von `<style>` und `</style>` gibt es weitere Angaben für die Tabelle:

```
@media screen and (max-width:480px){
  .linkespalte {
    width: 100%;
  }
}
```

Allerdings wirkt sich die Angabe innerhalb der *@media*-Regel nie aus, da Inline-Angaben prinzipiell eine höhere Spezifität haben. Um dieses Problem zu lösen, benötigen die Regeln innerhalb von *@media* den Zusatz *!important*. Nun setzen sie sich durch, wenn die Bedingung *Maximalbreite von 480px* zutrifft.

```
@media screen and (max-width:480px){
  .linkespalte {
    width: 100% !important;
  }
}
```

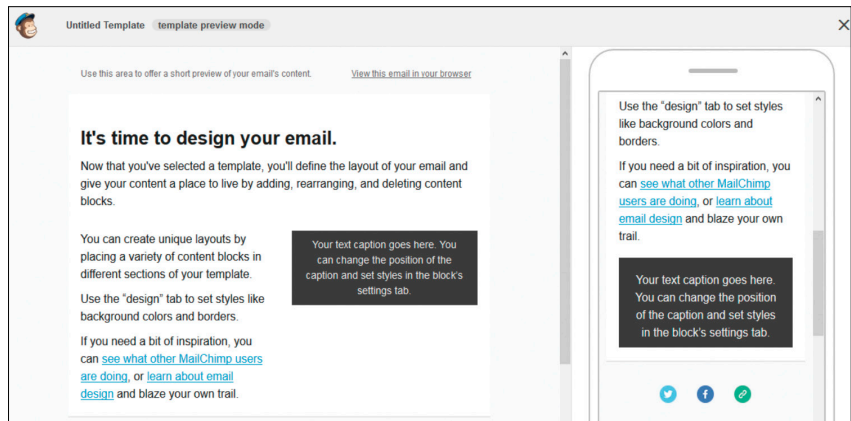
Angeichts der für bestimmte Clients benötigten Sonderangaben und der Komplexität des Codes ist es sinnvoll, sich bei der Erstellung von responsiven Mails helfen zu lassen.

Layoutvorlagen

Hier gibt es verschiedene Möglichkeiten: Eine Quelle sind die Responsive Email Patterns (Bild 5) mit Layoutvorlagen sowie Vorlagen für Navigationen und Listen.

Ein Framework für responsive HTML-Mails stammt von der Firma Zurb, die auch weitere bekannte Frameworks für Webseiten und Apps erstellt hat. Das Framework hieß ursprünglich Ink, nennt sich nun jedoch nur noch schlicht Foundation for Emails (Bild 6). Es bietet unter anderem ein Raster zur Gestaltung, mehrere Templates und Komponenten wie Buttons und mehr.

Viele Newsletter-Dienstleister bieten bereits direkt Templates oder sogar Assistenten an, die Sie durch die Erstellung



Vorschau eines anpassungsfähigen Layouts bei MailChimp (Bild 7)

der Mail führen – beispielsweise MailChimp (Bild 7). Das ist natürlich die komfortabelste Lösung.

Bei allen Templates sollten Sie sich die Informationen zu den unterstützten Mail-Clients genau ansehen und mit Ihren Statistiken vergleichen. Sinnvoll ist es außerdem, auf verschiedenen Systemen zu testen. Für Webseiten greift man dafür auf Dienste wie Browserstack zu – Ähnliches liefert Litmus für HTML-Mails.

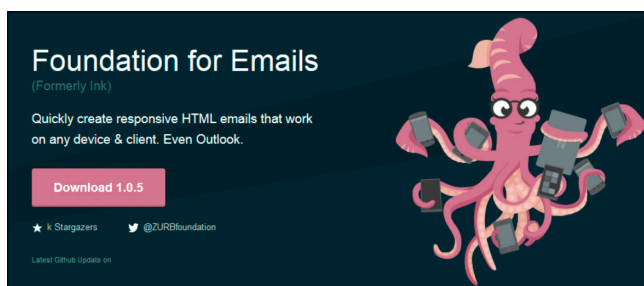
Fazit

Responsiv designte HTML-Mails sind machbar und werden auch empfohlen. Trotzdem ist die Herangehensweise ganz anders als bei Webseiten. Wichtig ist zuerst einmal eine anpassungsfähige Basis ohne Media Queries. Einfache Layoutveränderungen wie ein Wechsel zwischen einspaltiger und mehrspaltiger Darstellung lassen sich auf verschiedene Arten auch ohne *@media*-Regeln definieren.

Für fortgeschrittene Mail-Clients können Sie dann noch weitere Verbesserungen über *@media*-Regeln vornehmen. Damit diese Regeln sich gegenüber den Inline-Angaben durchsetzen, müssen sie mit *!important* markiert werden.

Bei Webseiten hat man die Wahl, ob man einem Mobile-First- oder einem Desktop-First-Ansatz folgt. Bei HTML-Mails hingegen ist der Desktop-First-Ansatz der richtige, da die Mail-Clients, die die *@media*-Regeln nicht unterstützen, im Wesentlichen auf dem Desktop zum Einsatz kommen.

Und noch mehr als bei Webseiten gilt für HTML-Mails: Eine pixelgenaue identische Darstellung über Clients hinweg kann nicht das Ziel sein. Vielmehr sollte eine funktionierende, attraktive Darstellung erreicht werden. ■



E-Mail-Framework von Zurb (Bild 6)



Dr. Florence Maurice

ist Autorin, Trainerin und Programmiererin in München. Ihr aktuelles Buch ist dem Thema »CSS3« gewidmet und bei dpunkt erschienen (ISBN 978-3-86490-118-8). Außerdem bloggt sie regelmäßig zu CSS und verwandten Themen unter: <http://maurice-web.de/blog>

ÜBERBLICK

CD-Highlights

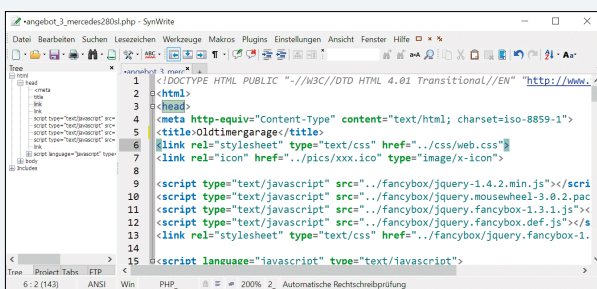
Auf der Heft-CD finden Sie Werkzeuge für Webentwickler sowie nützliche Tools.

Apple Swift 2.2



Swift wurde 2014 von Apple eingeführt und gilt derzeit als die am schnellsten wachsende Programmiersprache der Geschichte. Sie will die Leistung und Effizienz der kompilierten Sprachen mit der Einfachheit und Interaktivität gängiger Skriptsprachen kombinieren. Swift ist leicht zu erlernen und zu verwenden, selbst wenn man vorher noch nie programmiert hat – laut Apple ist es die erste System-Programmier Sprache, die so expressiv wie eine Skriptsprache ist. Verfügbare Komponenten enthalten Swift Compiler, Debugger, Standard Library, Foundation Libraries, Package Manager und REPL. Swift für Apple und Ubuntu Linux ist Open Source und benötigt Xcode ab Version 7.2.

SynWrite 6.19.215



SynWrite ist ein Quelltext-Editor für über 80 Programmiersprachen von ADA bis XML. Ein Highlight ist die automatische Syntaxhervorhebung während der Eingabe. SynWrite richtet sich vor allem an Software-Entwickler und Webdesigner. Der Übersichts dient neben der Baumstruktur eine Minimap, für wiederkehrende Aufgaben gibt es einen Makrorekorder. In großen Dateien helfen Lesezeichen sowie Such- und Textmarken. Außerdem gibt es einen FTP-Client und eine Rechtschreibprüfung.



Weitere Highlights:

FileZilla Server 0.9.4

Das Programm startet auf Ihrem Windows-Rechner einen eigenen FTP-Server für Zugriffe übers Internet. Legen Sie FTP-Nutzer mit Passwörtern an und wählen Sie Verzeichnisse aus, auf die diese zugreifen dürfen. In den Einstellungen geben Sie vor, ob die Nutzer Dateien und Verzeichnisse schreiben, lesen oder löschen dürfen.

PureSync 4.0.0

Das Tool synchronisiert Verzeichnisse, kopiert Daten auf externe Speichermedien und erstellt Backups. Für Privatanutzer ist die Software kostenlos. In PureSync richten Sie Abgleichvorgänge für Dateien und Ordner nur einmal ein und führen sie dann beliebig oft mit oder ohne Anpassung der Einstellungen aus.

ShareX 10.5.0

Per Knopfdruck erstellen Sie Bildschirmfotos mit zahlreichen Zusatzfunktionen. Neben der Bereichsauswahl haben Sie die Möglichkeit, Ihre Screenshots automatisch mit Rahmen, Wasserzeichen oder Logos zu versehen. Fertige Bilder lädt das Tool auf Wunsch gleich zu Dropbox oder zu anderen Datei-Hostern hoch.

TeamViewer 11.0.52465

Für den privaten Einsatz in einem nicht-kommerziellen Umfeld ist TeamViewer kostenlos. Das Tool baut über das Internet sichere Verbindungen zu anderen PCs auf, die sich dann fernsteuern lassen. Dabei bahnt sich TeamViewer einen Weg durch Firewalls, ohne dass eine Änderung der Regeln nötig ist.

Zenmap GUI for Nmap 7.01

Eine grafische Bedienoberfläche für den Portscanner Nmap, der nur auf der Kommandozeile läuft. Der Network Mapper ist vor allem dazu geeignet, alle aktiven Hosts in der Netzwerk-Umgebung sowie deren Betriebssystem und die Versions-Nummern verschiedener dort installierter Dienste herauszufinden.

Jetzt kostenlos testen!



2x gratis!



Praxiswissen für Entwickler!

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

webundmobile.de/probelesen

REGRESSIONSTESTS UNTER NODE.JS MIT WEBDRIVERIO UND SELENIUM 2.0

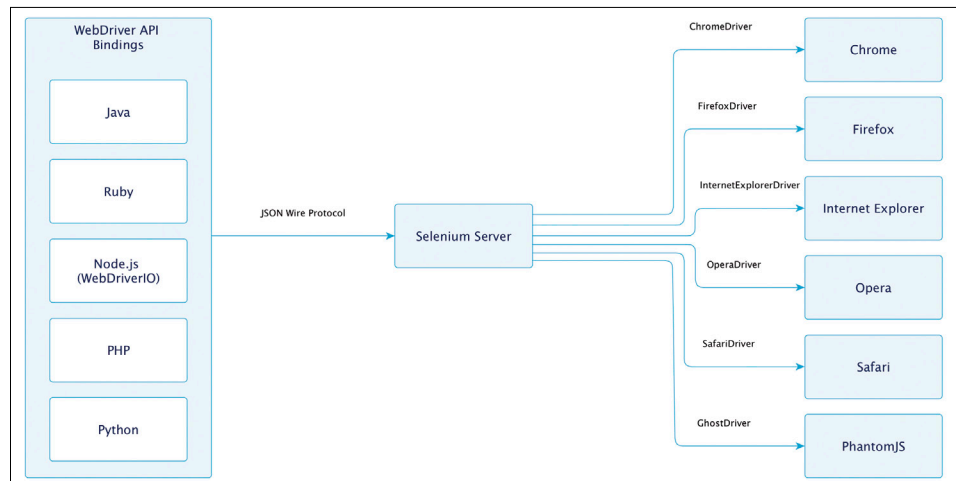
Visuelle Regressionstests

Regressionstests sollen neue Fehler bei Änderungen an einer Software verhindern.

Eine Software- oder Webanwendung erfährt in der Regel auch nach ihrer Veröffentlichung mehrere Änderungen, sei es durch Hinzufügen neuer Features oder das Beheben von Fehlern. Hierbei kann es leicht passieren, dass diese Änderungen ungewollte Auswirkungen auf andere bereits bestehende Komponenten der Anwendung haben beziehungsweise dort zu neuen Fehlern führen.

Das ist der Punkt, an dem Regressionstests ansetzen: Sie dienen dazu, genau diese Art von Fehlern aufzuspüren, bevor die entsprechenden Änderungen in die jeweilige Software integriert werden.

Im Rahmen der Webentwicklung sind diesbezüglich insbesondere sogenannte visuelle Regressionstests interessant. Das Ziel dabei ist es, die Anwendung speziell aus visuellen Gesichtspunkten gegen ungewollte Änderungen abzusichern, sprich zu verhindern, dass zum Beispiel durch neue CSS-Regeln bestehende Komponenten der Oberfläche in ihrem Aussehen oder ihrer Anordnung beeinflusst werden.



Die Architektur von Selenium und des Web Driver API (Bild 1)

Da solche Tests mit zunehmender Komplexität einer Webanwendung recht zeitaufwendig sein können, wenn man sie manuell durchführen würde (besonders wenn mehrere Auflösungen und verschiedene Browser in unterschiedlichen Versionen getestet werden sollen), liegt es nahe, diese Tests automatisiert durchzuführen.

Selenium Framework

Ein Framework, welches generell bezüglich automatisierter Browsertests sehr beliebt ist, ist das Selenium Framework, genauer gesagt Selenium 2.0 (www.seleniumhq.org), das im Unterschied zur ersten Version des Frameworks auch das sogenannte WebDriver API (www.w3.org/TR/webdriver) implementiert.

Dabei handelt es sich um eine Schnittstellenbeschreibung des W3C, deren Ziel es ist, den programmatischen Zugriff auf Funktionalitäten von Browsern zu vereinheitlichen. Ideal also, um automatisierte Cross-Browser-Tests im Allgemeinen oder auch die genannten visuellen Regressionstests im Besonderen durchzuführen.

Zentrale Komponente von Selenium 2.0 ist der Selenium-Server, an den sich über das JSON-Wire-Protokoll Befehle schicken lassen, die dieser dann über entsprechende Treiber an den jeweiligen Browser weiterleitet (Bild 1).

Auf diese Weise lassen sich sämtliche namhaften Browser (Chrome, Internet Explorer, Firefox, Safari und Opera) sowie der Headless Browser PhantomJS (<http://phantomjs.org>) fernsteuern.

Für den vereinfachten Zugriff gibt es mittlerweile WebDriver-API-Clients beziehungsweise Bindings für verschiedene

Listing 1: Verwendung von webdriverio

```
'use strict';
let webdriverio = require('webdriverio');
let options = {
  desiredCapabilities: {
    browserName: 'firefox'
  }
};

let client = webdriverio.remote(options);
client.init()
  .url('http://www.google.de')
  .title((error, response) => {
    console.log(
      response.value
    );
  })
  .end();
```

Programmiersprachen, die die Details des JSON-Wire-Protokolls vor dem Entwickler verbergen und entsprechende Funktionen bereitstellen, die sich dann bequem in der jeweiligen Anwendung verwenden lassen. Eines dieser Bindings ist das Node.js-Modul WebDriverIO (<http://webdriver.io>).

Installiertes Node.js ist Voraussetzung

Um WebDriverIO verwenden zu können, wird logischerweise ein installierter Selenium-Server vorausgesetzt. Musste man dazu in der Vergangenheit noch extra die Java-Laufzeitumgebung installieren, ist das mittlerweile dankenswerterweise nicht mehr so.

Stattdessen lässt sich – ein installiertes Node.js vorausgesetzt – das Node.js Modul selenium-standalone (<https://github.com/vvo/selenium-standalone>) verwenden. Dieses lässt sich wie für Node.js-Module gewohnt über den Node.js Package Manager NPM mit folgendem Befehl installieren:

```
npm install -g selenium-standalone@latest
```

Anschließend steht der Befehl *selenium-standalone* zur Verfügung, mit Hilfe dessen man über *selenium-standalone install* die Installation des Selenium-Servers durchführen kann. Zusätzlich werden Treiber für Chrome, Internet Explorer, Firefox und Chrome installiert. Nach erfolgreicher Installation lässt sich der Server dann über den Befehl *selenium-standalone start* starten.

Die Installation von WebDriverIO folgt ähnlich einfach mit Hilfe von NPM über den Befehl *npm install webdriverio*.

Ein einfaches Beispiel für die Verwendung von WebDriverIO zeigt **Listing 1**. Einbinden lässt sich das Modul wie für Node.js-Module gewohnt über *require()*. Anpassen kann man WebDriverIO jeweils über ein entsprechendes Konfigurationsobjekt (**Tabelle 1**).

Beispielsweise lassen sich hier Host und Port für den Selenium-Server einstellen, über die Eigenschaft *desiredCapabilities* wiederum verschiedene Selenium-spezifische Konfigurationen. Wer sich mit Selenium bereits auskennt, dürfte sich dementsprechend schnell zurechtfinden.

Verwendung von WebDriverIO

Zu den Einstellungen, die vorgenommen werden können, zählen beispielsweise der zu verwendende Browser, dessen Version oder das zu verwendende Betriebssystem (vorausgesetzt, eine entsprechende Selenium-Testumgebung mit virtuellen Maschinen steht bereit). Darüber hinaus zeigt **Tabelle 2** einige weitere ausgewählte Konfigurationen, die vorgenommen werden können.

Im Beispiel wird auf diese Weise Firefox als Browser konfiguriert. Andere mögliche Werte wären analog zu den entsprechenden Browsern *chrome*, *internet explorer*, *opera*, *safari* (wobei die Treiber für Opera und Safari manuell installiert werden müssen, da diese von selenium-standalone nicht installiert werden) oder auch *phantomjs*, um den Headless Browser PhantomJS zu verwenden.

Letzteres bietet sich beispielsweise an, um WebDriverIO-Skripts auf einem Continuous-Integration-Server wie Jenkins (<https://jenkins-ci.org>) auszuführen, ohne dass man ►

Tabelle 1: Konfiguration von WebDriverIO

Eigenschaft	Beschreibung
<i>desiredCapabilities</i>	Enthält ein Objekt zur Konfiguration von Selenium
<i>logLevel</i>	Angabe des Loglevels
<i>logOutput</i>	Angabe einer Datei oder eines Datenstreams, in die/den Logging-Meldungen geleitet werden sollen
<i>host</i>	Der Host des WebDriver-Servers (beziehungsweise des Selenium-Servers)
<i>port</i>	Der Port des WebDriver-Servers (beziehungsweise des Selenium-Servers)
<i>path</i>	Der Pfad zum WebDriver-Server (beziehungsweise zum Selenium-Server)
<i>baseUrl</i>	Angabe eines Basis-URL
<i>coloredLogs</i>	Angabe, ob Logging-Meldungen farblich codiert werden sollen
<i>screenshotPath</i>	Angabe eines Pfades zur Speicherung von Screenshots, falls bei der Verwendung des WebDriver-Servers (beziehungsweise des Selenium-Servers) ein Fehler auftritt
<i>waitforTimeout</i>	Angabe eines Timeouts für alle möglichen WebDriverIO-Funktionen, die auf bestimmte Ereignisse warten

Tabelle 2: Konfiguration von Selenium (Ausschnitt)

Eigenschaft	Beschreibung
<i>browserName</i>	Name des zu verwendenden Browsers
<i>version</i>	Version des zu verwendenden Browsers
<i>platform</i>	Name des zu verwendenden Betriebssystems
<i>javascriptEnabled</i>	Boolesche Angabe, ob JavaScript aktiviert sein soll
<i>databaseEnabled</i>	Boolesche Angabe, ob auf die Browserdatenbanken zugegriffen werden darf
<i>locationContextEnabled</i>	Boolesche Angabe, ob auf den Location Context des Browsers zugegriffen werden darf
<i>applicationCacheEnabled</i>	Boolesche Angabe, ob auf den Application Cache zugegriffen werden darf
<i>browserConnectionEnabled</i>	Boolesche Angabe, ob auf Verbindungsinformationen des Browsers zugegriffen werden darf
<i>webStorageEnabled</i>	Boolesche Angabe, ob auf Storage-Objekte (localStorage, sessionStorage) zugegriffen werden darf
<i>waitforTimeout</i>	Angabe eines Timeouts für alle WebDriverIO-Funktionen, die auf bestimmte Ereignisse warten

(wie im Fall der anderen Browser) dafür Selenium-seitig jeweils eine virtuelle Maschine konfigurieren muss.

Ausgehend von diesem Konfigurationsobjekt lässt sich über den Aufruf von *remote()* ein WebDriver-Client-Objekt erzeugen, über das im Anschluss die Kommunikation mit dem WebDriver-Server beziehungsweise dem Selenium-Server möglich ist. Somit können entsprechende Befehle an den jeweiligen Browser formuliert werden.

Im Beispiel wird die Webseite www.google.de aufgerufen und deren Titel ausgegeben. Neben diesem zugegeben recht einfachen Beispiel lassen sich aber auch komplexere Abläufe und Interaktionen mit einer Webseite formulieren. Dazu zählt das Ausfüllen und Absenden von Formularen, das Simulieren von Mausklicks, das Auslösen von Drag-and-Drop-Ereignissen und vieles mehr. Eine detaillierte Beschreibung des API findet sich auf der Webseite des Projekts unter <http://webdriver.io/api.html>.

Übrigens lässt sich das API nicht nur mit Callback-Funktionen verwenden, sondern auch wie in Listing 2 zu sehen mit ES6 Promises. Alternativ dazu ist es sogar möglich, ES6 Generatoren zu verwenden (ein Beispiel dazu findet man auf der Homepage von WebDriverIO).

Visuelle Regressionstests mit WebDriverCSS

WebDriverIO verfügt über ganze eine Reihe von interessanten Plug-ins, beispielsweise für das vereinfachte Testen von AngularJS-Anwendungen (<http://webdriver.io/guide/plugins/webdriverjs-angular.html>) oder für die Integration in die Build-Tools Gulp (<http://webdriver.io/guide/plugins/gulp-webdriver.html>) und Grunt (<http://webdriver.io/guide/plugins/grunt-webdriver.html>).

Das Plug-in **WebDriverCSS** (<http://webdriver.io/guide/plugins/webdrivercss.html>) dagegen bietet die Möglichkeit, visuelle Regressionstests durchzuführen. Der Name des Plug-ins ist dabei etwas irreführend, da Änderungen an der Ober-

Listing 2: Verwendung von webdriverio mit Promises

```
'use strict';
let webdriverio = require('webdriverio');
let options = {
  desiredCapabilities: {
    browserName: 'firefox'
  }
};

let client = webdriverio.remote(options);
client.init()
  .url('http://www.google.com')
  .title()
  .then((response) => {
    console.log(
      response.state,
      response.sessionId,
      response.value
    );
  })
  .catch((error) => {
    console.error(error)
  })
  .end();
```

fläche natürlich nicht zwangsläufig durch CSS hervorgerufen werden müssen.

Die Idee bei dem Plug-in ist es, das aktuelle Aussehen einer Webanwendung mit Referenz-Screenshots zu vergleichen und entsprechend Fehler zu melden, wenn hierbei zu starke Abweichungen zwischen den Screenshots auftreten. Installieren lässt sich das Plug-in ebenfalls über NPM mit Hilfe des Befehls:

```
npm install webdrivercss
```

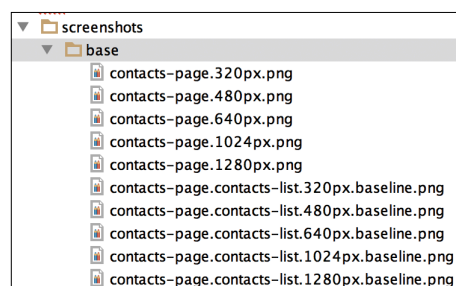
Da das Plug-in momentan allerdings nicht mit der Version 3 von WebDriverIO funktioniert, muss Letzteres in Version 2.4.5 installiert werden (*npm install webdriverio@2.4.5*).

Verwendung von WebDriverCSS

Den Ausgangspunkt für die nachfolgenden Beispiele bildet der HTML-Code aus Listing 3. Hierbei handelt es sich um ei-

Tabelle 3: Konfiguration von WebDriverCSS

Eigenschaft	Beschreibung
<i>screenshotRoot</i>	Angabe des Pfades, unter dem die Screenshots gespeichert werden sollen
<i>failedComparisonsRoot</i>	Angabe des Pfades, unter dem im Fall von Abweichungen entsprechende Screenshots gespeichert werden sollen
<i>misMatchTolerance</i>	Zahl zwischen 0 und 100, die beschreibt, inwieweit zwei Screenshots voneinander abweichen dürfen, bevor ein Fehler gemeldet wird
<i>screenWidth</i>	Angabe eines Arrays von Zahlenwerten, um Screenshots für verschiedene Fensterbreiten zu testen
<i>updateBaseline</i>	Boolesche Angabe, ob die Referenz-Screenshots aktualisiert werden sollen, wenn der Vergleich mit diesen dauerhaft fehlschlägt (in so einem Fall wird dann davon ausgegangen, dass die aktuellen Screenshots als neue Referenzscreenshots verwendet werden sollen)



Das Modul **webdrivercss** generiert Screenshots für verschiedene Bildschirmgrößen (Bild 2)

Listing 3: Ausgangs-HTML für die Regressionstests

```

<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
  <link rel="stylesheet" href="https://
maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/
bootstrap.min.css">
  <link rel="stylesheet" href="styles/styles.css">
</head>
<body>
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle
collapsed" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1"
aria-expanded="false">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
    </div>
    <div class="collapse navbar-collapse"
id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li><a href="#">Home</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Separated link</a></li>
      </ul>
    </div>
  </div>
</nav>
<div id="contacts" class="container">
  <div class="col-xs-12 col-sm-6">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h3 class="panel-title">Kontakte</h3>
      </div>
      <div class="panel-body">
        <ul class="list-group">
          <li class="list-group-item">Max Mustermann</li>
          <li class="active list-group-item">
            Peter Meier</li>
          <li class="list-group-item">
            Moritz Mustermann</li>
          <li class="list-group-item">Petra Meier</li>
          <li class="list-group-item">Paul Müller</li>
        </ul>
      </div>
    </div>
  </div>
  <div class="col-xs-12 col-sm-6">
    <div class="panel panel-default">
      <div class="panel-heading">
        <h3 class="panel-title">Kontaktdetails</h3>
      </div>
      <div class="panel-body">
        <label for="firstName">Vorname</label>
        <input type="text" class="form-control"
id="firstName" value="Peter">
        <label for="lastName">Nachname</label>
        <input type="text" class="form-control"
id="lastName" value="Mustermann">
        <label for="age">Alter</label>
        <input type="text" class="form-control"
id="age" value="33">
        <label for="phone">Telefon</label>
        <input type="text" class="form-control"
id="phone" value="0228/234567">
      </div>
    </div>
  </div>
</div>
<script src="https://code.jquery.com/
jquery-2.1.4.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.5/js/bootstrap.min.js"></script>
</body>
</html>

```

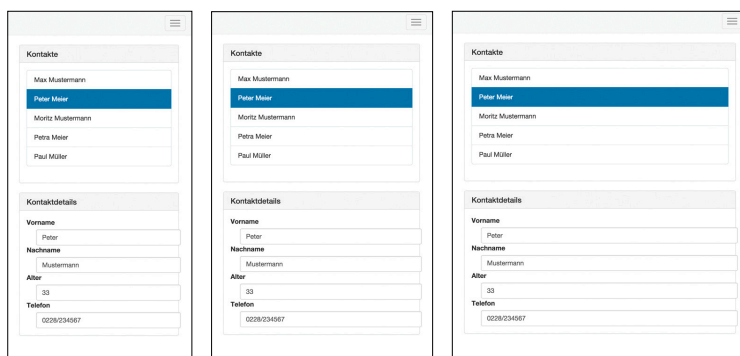
ne einfach gestrickte Webseite (**Bild 2**), die lediglich aus einer Navigationsleiste und zwei Bereichen besteht, nämlich einer Kontaktliste und den Kontaktdetails. Als CSS-Framework und zur Definition des responsiven Verhaltens wurde das weit verbreitete Framework Bootstrap 3 (<http://getbootstrap.com>) verwendet.

Die Verwendung von WebDriverCSS zeigt exemplarisch **Listing 4**. Um WebDriverCSS zu initialisieren, ruft man die Methode *init()* auf und übergibt ihr den WebDriver-Client sowie ein Konfigurationsobjekt (**Tabelle 3**). Über dieses Konfigurationsobjekt lassen sich unter anderem die Fensterbreite des

Browsers sowie Zielordner für das Speichern der Screenshots angeben.

Der eigentliche Test wird anschließend über die Methode *webdrivercss()* gestartet. Durch Parameter lassen sich wie im Beispiel gezeigt die Regressionstests auch gezielt für einzelne Elemente der Oberfläche durchführen. Im Beispiel wird auf diese Weise das Element mit der ID *contacts* ausgewählt, sprich, lediglich dieser Teil der Webseite wird in die Tests mit einbezogen.

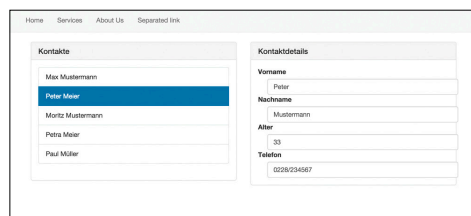
Anschließend generiert WebDriverCSS beim ersten Lauf für jede der angegebenen Fensterbreiten einen Screen- ►



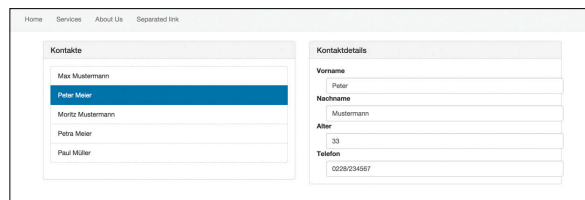
Screenshot für
Fensterbreite von
320 Pixeln (Bild 3)

Screenshot für
Fensterbreite von
480 Pixeln (Bild 4)

Screenshot für
Fensterbreite von
640 Pixeln (Bild 5)



Screenshot für Fensterbreite von 1024 Pixeln (Bild 6)



Screenshot für Fensterbreite von 1280 Pixeln (Bild 7)

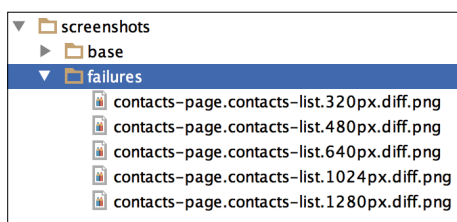
shot im angegebenen Ordner (`./test/screenshots/base`). Die Namen der Dateien setzen sich dabei aus der Fensterbreite und den vorher beim Aufruf von `webdrivercss()` definierten Kennzeichnern zusammen.

Diese generierten Screenshots (Bild 3 bis Bild 7) werden von WebDriverCSS für nachfolgende Testläufe als Referenzpunkt verwendet. Weicht ein Screenshot zu stark von dem jeweili-

gen Referenz-Screenshot ab, speichert WebDriverCSS die Abweichung in Form eines weiteren Screenshots, der jedoch in einem separaten Ordner steht (im Beispiel `./test/screenshots/failures`) (Bild 8).

Um nun eine Änderung zu simulieren, die Auswirkungen auf das Layout der Anwendung hat, wird im Folgenden der eingebundenen CSS-Datei die Regel `input {margin-left: 20px}` hinzugefügt, wodurch der linke Rand von Eingabefeldern auf 20 Pixel gesetzt wird. Lässt man nun das Testskript erneut laufen, erkennt WebDriverCSS die Abweichungen und generiert entsprechende Screenshots (Bild 9 bis Bild 13). Die Abweichungen sind jeweils farblich hervorgehoben.

Regressionstests sind natürlich nur sinnvoll, wenn man sie auch automatisiert ausführen kann. Listing 5 zeigt, wie sich WebDriverIO beziehungsweise WebDriverCSS in einen Mocha-Test integrieren lassen.



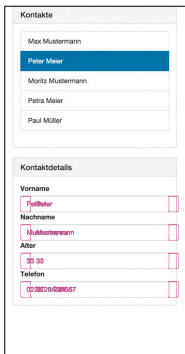
Für jede Auflösung, für die eine Regression festgestellt wurde, wird ein Screenshot generiert (Bild 8)

Listing 4: Verwendung von webdrivercss

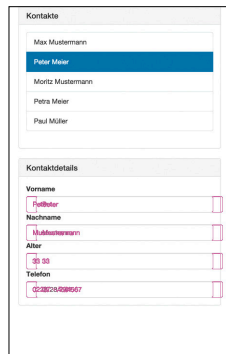
```
'use strict';
let webdriverio = require('webdriverio');
let webdrivercss = require('webdrivercss');
let options = {
  desiredCapabilities: {
    browserName: 'firefox'
  }
};
let client = webdriverio.remote(options);

webdrivercss.init(
  client,
  {
    screenshotRoot: './test/screenshots/base',
    failedComparisonsRoot:
      './test/screenshots/failures',
    mismatchTolerance: 0.5,
    screenWidth: [
      320,
      480,
      640,
      1024,
      1280
    ]
  }
);

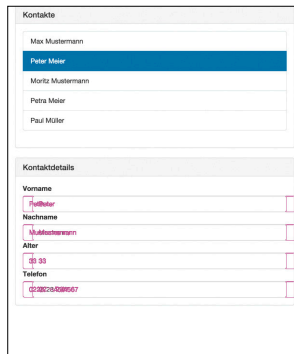
client
  .init()
  .url('http://localhost:8080/')
  .webdrivercss(
    'contacts-page',
    [
      {
        name: 'contacts-list',
        elem: '#contacts'
      }
    ]
  )
  .end();
```



Differenz für
Fensterbreite von
320 Pixeln (Bild 9)



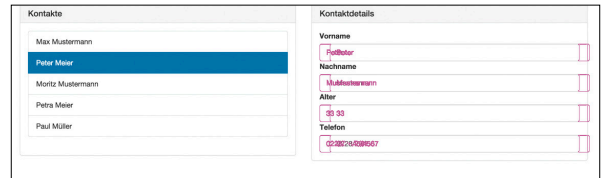
Differenz für
Fensterbreite von
480 Pixeln (Bild 10)



Differenz für
Fensterbreite von
640 Pixeln (Bild 11)



Differenz für Fensterbreite von 1024 Pixeln (Bild 12)



Differenz für Fensterbreite von 1280 Pixeln (Bild 13)

Initialisierung und Aufruf sind dabei gegenüber den vorigen Listings unverändert, allerdings wird zusätzlich über die Eigenschaft *isWithinMisMatchTolerance* innerhalb eines *assert.ok()* überprüft, ob sich die Abweichung zwischen zwei Versionen der Weboberfläche innerhalb des tolerierten Bereichs befindet. Ist dies der Fall, besteht der Test, andernfalls schlägt er fehl.

Fazit

Je komplexer eine Webanwendung und je komplexer deren Oberfläche, desto aufwendiger ist es für einen Entwickler, manuell zu überprüfen, ob neue CSS-Regeln ungewollte Auswirkungen auf bestehende Bereiche der Webanwendung haben. Mit Hilfe der Node.js-Module *webdriverio* und *web-*

drivercss lassen sich relativ einfach visuelle Regressionstests definieren, die automatisiert ausgeführt werden können und einem als Entwickler eine Menge Arbeit abnehmen. ■



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>

Listing 5: Verwendung von *webdrivercss* in einem Mocha-Test

```
'use strict';
let assert = require('assert');
let webdriverio = require('webdriverio');
let webdrivercss = require('webdrivercss');
let options = {
  desiredCapabilities: {
    browserName: 'firefox'
  }
};
let client = webdriverio.remote(options);

webdrivercss.init(client, {
  screenshotRoot: './test/screenshots/base',
  failedComparisonsRoot: './test/screenshots/failures',
  misMatchTolerance: 0.5,
  screenWidth: [320,480,640,1024]
});

describe('my website should always look the same', () =>
{
  it('header should look the same', (done) =>
  {
    client
      .init()
      .url('http://localhost:8080/')
      .webdrivercss('contacts-page',
        {
          name: 'contacts-list',
          elem: '#contacts'
        }, (error, response) =>
        {
          assert.ifError(error);
          assert.ok(
            response['contacts-list'][0].
              isWithinMisMatchTolerance
          );
        }
      )
      .call(done);
  });
});
```

WEB ANIMATIONS API

JavaScript-Animationen

Über das Web Animations API lassen sich Animationen definieren und steuern.

Das Web Animations API (www.w3.org/TR/web-animations) bietet Entwicklern die Möglichkeit, innerhalb einer Webseite per JavaScript Animationen zu definieren und zu steuern.

Das API definiert dazu eine Reihe neuer Interfaces, die Stand heute von Firefox, Chrome und Opera implementiert werden. Internet Explorer, Microsoft Edge und Safari unterstützen das API derzeit noch nicht. Für mehr Details empfiehlt sich wie immer ein Blick auf die Webseite <http://caniuse.com>. Dort gibt man den Suchbegriff *Web Animations API* ein, um im Detail zu erfahren, welche Browser das Web Animations API unterstützen und welche nicht (Bild 1).

In Browsern, die bislang noch keine Unterstützung anbieten, kann man gegebenenfalls auf Polyfills wie das unter <https://github.com/web-animations/web-animations-js> zu Findende zurückgreifen.

Verwendung des API

Kern des Web Animations API bildet die Methode *animate()*, die durch das API für DOM-Elemente neu bereitgestellt wird. Über diese Methode lassen sich einzelne Animationen erstellen, die jeweils durch ein Objekt vom (ebenfalls neuen) Typ *Animation* repräsentiert werden (Tabelle 1).

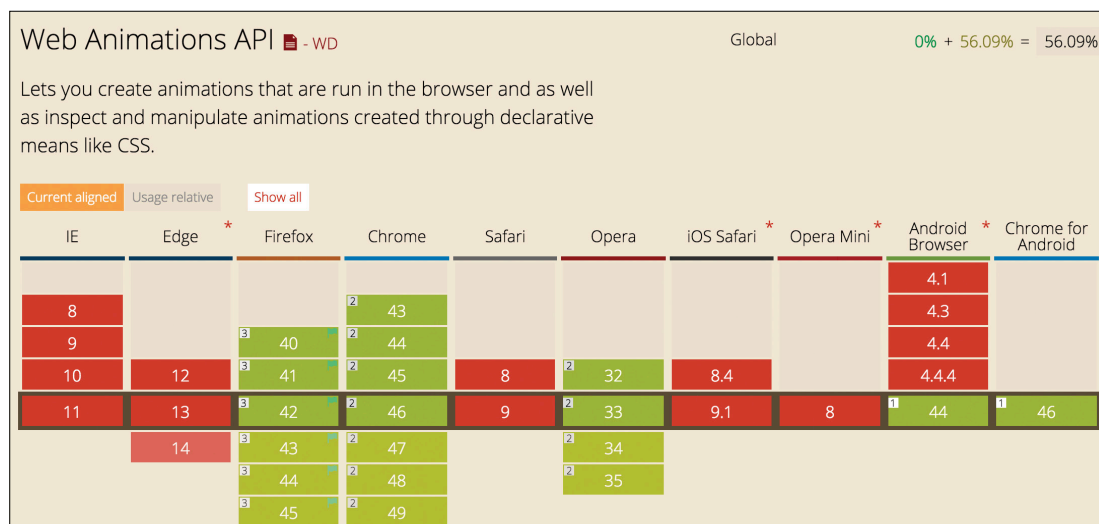
Listing 1 zeigt ein einfaches Beispiel für den Einsatz des API: Der Methode *animate()* übergibt man dabei als ersten Para-

Tabelle 1: Die wichtigsten Interfaces des Web Animations API

Interface	Beschreibung
<i>Animation</i>	Repräsentiert eine Animation
<i>Animatable</i>	Repräsentiert ein animierbares Element auf einer Webseite. Wird von dem Element-Interface und dem PseudoElement-Interface implementiert, wodurch alle DOM-Elemente einer Webseite die Methode <i>animate()</i> erhalten
<i>AnimationTimeline</i>	Fasst die Animationen auf einer Zeitachse zusammen
<i>AnimationEffectTiming</i>	Repräsentiert die Effekte einer Animation (der zweite Parameter, welcher der Methode <i>animate()</i> übergeben wird)
<i>DocumentTimeline</i>	Fasst die Animationen einer Webseite in einer Zeitachse zusammen. Leitet von <i>AnimationTimeline</i> ab
<i>KeyframeEffect</i>	Repräsentiert die Effekte eines einzelnen Keyframes einer Animation. Typ der Objekte, die als Array-Parameter der Funktion <i>animate()</i> übergeben werden

meter ein Array von Keyframe-Beschreibungen und als zweiten Parameter ein Konfigurations-Objekt, über das sich fallweise weitere Eigenschaften der Animation definieren lassen (Tabelle 2).

In etwa entsprechen die Einstellungen, die man über diese beiden Parameter vornehmen kann, den Einstellungen, die man auch von der Arbeit mit CSS-Animationen kennt (Listing 2). Die Keyframe-Beschreibungen lassen sich somit auf die *@keyframes*-Regel, das Konfigurationsobjekt auf die (CSS-)Shorthand-Eigenschaft *animation* (beziehungsweise die einzelnen *animation-**-Eigenschaften) abbilden.



Browser: Derzeit wird das Web Animations API noch nicht von allen größeren Browsern unterstützt (Bild 1)

Listing 1: Verwenden des Web Animations API

```
'use strict';
function init() {
  let animation =
    document.getElementById('circle').animate([
      {
        transform: 'scale(2)',
        opacity: 1,
        offset: 0
      },
      {
        transform: 'scale(3)',
        opacity: 0.8,
        offset: 0.3
      },
      {
        transform: 'scale(4)',
        opacity: 0.6,
        offset: 0.6
      },
      {
        transform: 'scale(5)',
        opacity: 0.7,
        offset: 1
      }
    ], {
      duration: 500,          // Dauer in Millisekunden
      easing: 'ease-in-out',  // Animationsverhalten
      delay: 10,             // Verzögerung in Millisekunden
      iterations: Infinity,   // Anzahl Wiederholungen
      direction: 'alternate', // Animationsrichtung
      fill: 'both'           // Füllverhalten der Animation
    });

  document.addEventListener('DOMContentLoaded', init);
}
```

Über die (Objekt-)Eigenschaft *offset* jedes Keyframe-Objekts lässt sich im Detail definieren, zu welchem Zeitpunkt der jeweilige Keyframe innerhalb der Animation gezeigt werden soll. Lässt man diese Eigenschaft weg, so werden die einzelnen Keyframes gleichmäßig auf die definierte Zeitdauer verteilt.

Steuern einer Animation

Darüber hinaus lassen sich für jedes Keyframe (animierbare) CSS-Eigenschaften wie beispielweise *transform* oder *opacity* und ihre Werte angeben, die das animierte Element zum jeweiligen Zeitpunkt der Animation annehmen soll. Die Werte zwischen zwei Keyframes werden in diesem Fall automatisch interpoliert.

Kurz gesagt: Wenn man in Projekten bereits mit CSS-Animationen zu tun hatte, wird man auch schnell die Grundlagen des Web Animations API verstehen. Vom Prinzip her

Listing 2: Äquivalente CSS-Animation

```
@keyframes example {
  0% {
    transform: scale(2);
    opacity: 1;
  }
  30% {
    transform: scale(3);
    opacity: 0.8;
  }
  60% {
    transform: scale(4);
    opacity: 0.6;
  }
  100% {
    transform: scale(5);
    opacity: 0.7;
  }
}

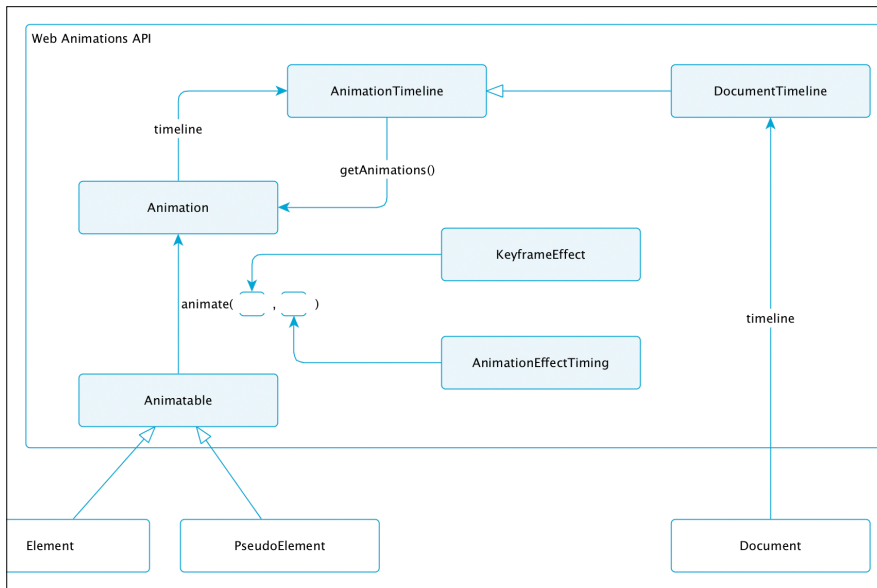
#circle {
  animation: example 500ms ease-in-out 10ms infinite
  alternate both;
}
```

Tabelle 2: Konfigurationsoptionen einer Animation

Eigenschaft	Beschreibung
<i>delay</i>	Verzögerung (in Millisekunden), mit der die Animation gestartet werden soll
<i>endDelay</i>	Verzögerung (in Millisekunden), mit der die Animation beendet werden soll
<i>fill</i>	Füllverhalten der Animation (<i>none</i> , <i>forwards</i> , <i>backwards</i> , <i>both</i> , <i>auto</i>)
<i>iterationStart</i>	Gibt die Iteration an, in der ein bestimmter Effekt für die Animation ausgelöst werden soll
<i>iterations</i>	Anzahl an Wiederholungen (Infinity oder konkrete Anzahl)
<i>duration</i>	Dauer der Animation in Millisekunden
<i>direction</i>	Richtung der Animation (<i>alternate</i> , <i>normal</i> , <i>reverse</i> , <i>alternate-reverse</i>)
<i>easing</i>	Verhalten der Animation (<i>ease</i> , <i>ease-in</i> , <i>ease-out</i> , <i>ease-in-out</i> , <i>cubic-bezier()</i>)

funktioniert das alles ähnlich, nur eben nicht deklarativ, sondern programmatisch.

Über das Animation-Objekt, das man von der Methode *animate()* erhält, lässt sich das Abspielverhalten der Animation steuern. Über die Methode *pause()* lässt sich die Animation pausieren, über *play()* wieder starten. Die Methoden *cancel()* und *finish()* erlauben es zudem, eine Animation abzubrechen oder zu beenden. ►



Zusammenhang der verschiedenen Interfaces, die durch das Web Animations API definiert und erweitert werden (Bild 2)

Listing 3 zeigt dazu ein entsprechendes Beispiel. Dem ersten Beispiel werden hier verschiedene Schaltflächen hinzugefügt, die über `click`-Listener die entsprechenden Funktionen am Animations-Objekt aufrufen. Ebenfalls zu sehen: über die Eigenschaft `playbackRate` lässt sich die Geschwindigkeit der Animation beeinflussen.

Im Detail ist das Web Animations API schon recht umfangreich, enthält aber auch viele Informationen zum Thema Animationen im Allgemeinen. Heruntergebrochen auf die wesentlichen Interfaces kann man sich relativ schnell zurechtfinden. Bild 2 zeigt als Einstieg hierzu die wichtigsten Interfaces und veranschaulicht den Zusammenhang zwischen den einzelnen Komponenten. Bezüglich der Unterstützung sind wie erwähnt die Browser Chrome, Firefox und Opera vorne

mit dabei. Aber auch hier werden bisher nicht alle Aspekte des API unterstützt. Chrome beispielsweise stellt für das `document`-Objekt noch keine Eigenschaft `timeline` zur Verfügung, wie es eigentlich gefordert ist und über die man unter anderem Zugang zu allen Animationen auf einer Webseite erhält.

Fazit

Das Web Animations API befindet sich momentan im Status Working Draft. Ein Blick auf die Änderungen (www.w3.org/TR/web-animations/#changes-since-last-publication), die es gegenüber der vorigen Version vom Juni 2014 gegeben hat (die aktuelle Version stammt vom Juli 2015), macht deutlich, dass weiterhin aktiv an dem API gearbeitet wird.

Ein Einsatz in Produktsystemen ist momentan (wie für Working Drafts oft üblich) also noch nicht anzuraten. Experimentieren kann man aber in den genannten Browsern oder eben mit einem Polyfill schon heute. ■



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>

Listing 3: Steuern einer Animation

```

let buttonPause = document.getElementById
('button-pause');
buttonPause.addEventListener('click', () => {
  animation.pause();
});

let buttonPlay = document.getElementById
('button-play');
buttonPlay.addEventListener('click', () => {
  animation.play();
});
let buttonCancel =
document.getElementById('button-cancel');
buttonCancel.addEventListener('click', () => {
  animation.cancel();
});

```

```

let buttonFinish = document.getElementById
('button-finish');
buttonFinish.addEventListener('click', () => {
  animation.finish();
});

let buttonFaster = document.getElementById
('button-faster');
buttonFaster.addEventListener('click', () => {
  animation.playbackRate *= 2;
});
let buttonSlower = document.getElementById
('button-slower');
buttonSlower.addEventListener('click', () => {
  animation.playbackRate /= 2;
});

```



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 Architektur sowie Smart Development.



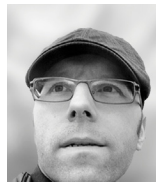
Das Advisory Board der SMART DATA Developer Conference:



Sascha Dittmann,
Microsoft
Deutschland GmbH



Robert Eichenseer,
Microsoft Corp.



Oliver B. Fischer,
E-Post
Development
GmbH



Constantin Klein,
Freudenberg
IT SE & Co. KG



Michael Nolting,
Sevenval
Technologies
GmbH



Stefan Papp,
Teradata
Operations Inc.



**Agnieszka
Walorska,**
Creative
Construction
Heroes GmbH

smart-data-developer.de #smartddc



SMARTDATADeveloperConference

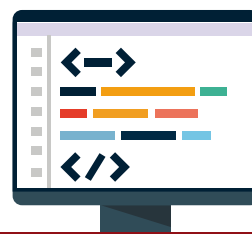
Präsentiert von: dotnetpro

web & mobile
DEVELOPER

Veranstalter:



Neue
Mediengesellschaft
Ulm mbH



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

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

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

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

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 algorithms 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 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.



101101011100110110
101101011100110110101



Workshops: 19. April 2016

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.

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



Jan Fellien,
devCrowd GmbH



Daniel Greitens,
MAXIMAGO GmbH



Werner Keil,
Creative Arts &
Technologies



Constantin Klein,
Freudenberg
IT SE & Co. KG



Olivia Klose,
Microsoft Deutschland
GmbH



André Krämer,
Software, Training &
Consulting



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



Stefan Lieser,
CCD School



Michael Nolting,
Sevenval Technologies
GmbH



Diego Poza,
Auth0 Inc.



Peter Rozek,
ecx.io germany GmbH



Alexander Schulze,
Innotrade GmbH



Rupert Steffner,
Otto (GmbH & Co KG)



Marcel Tilly,
Microsoft Research



Thomas Worm,
DATEV eG



Foto: Shutterstock / Bakhtiar Zein

DEPENDENCY INJECTION UND UNIT-TESTING MIT ANGULAR 2.0

Alleinstellungsmerkmal

DI und gute Testbarkeit waren schon immer Alleinstellungsmerkmale für AngularJS.

Hinweis: Das hier gezeigte Beispiel nutzt eine Vorschauversion von Angular 2.0. Der gezeigte Code muss für spätere Versionen wahrscheinlich angepasst werden.

In den beiden vorangegangenen Artikeln zu Angular 2.0 wurde zunächst ein modulares Setup auf Basis von SystemJS beschrieben. Im zweiten Artikel haben wir Ihnen anhand einer Beispielanwendung die neue Template-Syntax vorgestellt. Es ist nun an der Zeit, dieser Anwendung neue Funktionen zu geben, professionelle Entwurfsmuster anzuwenden und die Fehlerfreiheit des Codes mit Unit-Tests zu beweisen.

Inversion of Control

Das Dashboard ist die Demo-Anwendung aus dem vorangegangenen Artikel. Die Anwendung soll in diesem Artikel eine neue Funktionalität erhalten. Es soll per AJAX nach dem aktuell günstigsten Benzinpreis gesucht werden. Mit dem ermittelten Preis kann man dann die verwalteten Autos auftanken (Bild 1).

Wenn man an einer beliebigen Stelle im Programmcode eine andere Funktionalität benötigt, dann liegt es zunächst nahe, jene andere Funktionalität an Ort und Stelle zu initialisieren. Ein erster Ansatz könnte wie folgt aussehen:

```
var Dashboard = function() {
  this.gasService =
    new GasService();
  // gasService verwenden
  this.gasService.getBestPrice();
}
```

Dieses Vorgehen ist prinzipiell einwandfrei – nur stößt man mit steigender Menge an Code an eine Grenze. Der Code wird zunehmend unübersichtlicher, schwerer zu warten und verweigert sich einem einfachen Test-Setup. Dem Problem lässt sich begegnen, indem man die Verantwortung zum Erzeugen von Abhängigkeiten an eine übergeordnete Stelle abgibt.

Das ist die Idee hinter dem Prinzip des »Inversion of Control«. Bei die-

Cars dashboard	
Best Oil Price 1.049 €	
ID NG-CAR 1.0	Refill for 80 €
Damaged false	
Tank Capacity 1.33 liter	
Driver Johannes	
Stone impact	
ID NG-CAR 2.0	Refill for 100 €
Damaged false	
Tank Capacity 0.00 liter	
Driver Gregor	
Stone impact	

Benzinpreis: Die Autos zum günstigsten Benzinpreis betanken (Bild 1)

sem Prinzip kehrt man die Verantwortlichkeit einfach um. Das Prinzip findet sich in verschiedenen Entwurfsmustern bei allen Programmiersprachen wieder. AngularJS zum Beispiel verwendet das Entwurfsmuster Dependency Injection. Ein Framework im Kern von AngularJS sorgt dafür, dass die benötigte Abhängigkeit identifiziert wird und die Konstruktor-Funktion beim Aufruf bereitgestellt wird. In AngularJS 1.x kann man einen Service wie folgt anfordern:

```
// AngularJS 1.x
var Dashboard = ['GasService', function(GasService) {
  this.gasService = GasService;
}];
```

Wer das DI-Framework aus AngularJS 1.x kennt, wird mit Sicherheit auch an dessen Grenzen gestoßen sein. Besonders hinderlich sind fehlende Namespaces und die Notwendigkeit, stets alle Abhängigkeiten per Namen zu identifizieren. Dies ist doppelter Schreibaufwand. Im vorliegenden Beispiel muss man beispielsweise zweimal *GasService* schreiben.

Dependency Injection in Angular 2.0

Mit der Unterstützung von ECMAScript 6 beziehungsweise von TypeScript wird die Bedienung nun viel vertrauter (Bild 2). So lässt sich mittels des Decorators *@Inject* die Abhängigkeit in den Konstruktor injizieren:

```
class GasService {
}
class Dashboard {
  constructor(@Inject
    (GasService) gasService) {
    console.log('Dependency:',
      gasService)
  }
}
var injector = Injector.resolveAndCreate([Dashboard,
  GasService]);
var dashboard = injector.get(Dashboard);
```

Die Methode *resolveAndCreate()* erzeugt einen einsatzbereiten Injector. Die Methode akzeptiert ein Array aus Typen oder Providern. Übergibt man einen Typ, so wird ein entsprechender Provider für diesen Typ erzeugt. Mit dem Einsatz von TypeScript lässt sich die Schreibweise noch weiter vereinfachen. Durch die Verwendung von Typen kann man auf den Decorator *@Inject* verzichten:

```
@Injectable()
class Dashboard {
  constructor(gasService: GasService) {
    console.log('Dependency:', gasService)
  }
}
```

Damit dieses Beispiel funktioniert, muss TypeScript einen Hinweis dazu erhalten, dass der Konstruktor mit Decoratoren

versehen werden soll. Dies geschieht mit dem Decorator *@Injectable()*.

Auf die Verwendung von *@Injectable()* kann verzichtet werden, sobald ein anderer Decorator die Klasse verziert. Weitere Decoratoren sind etwa *@Component()*, *@View()* oder *@RouteConfig()*. Da Angular 2.0 stark auf einen deklarativen Stil mittels Decoratoren setzt, benötigt man *@Injectable()* eigentlich nur für eigene Service-Klassen.

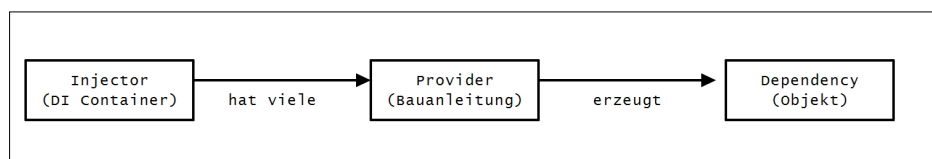
Der Injector versteht eine Reihe von Bauanleitungen. Hierzu verwendet man die Methode *provide()*. Das Beispiel kann auch in einer längeren Syntax ausgedrückt werden. Soll also ein Token als Klasse aufgelöst werden, so verwendet man *useClass*:

```
var injector = Injector.resolveAndCreate([Dashboard,
  GasService]);
```

// entspricht:

```
var injector = Injector.resolveAndCreate([
  provide(Dashboard, {useClass: Dashboard}),
  provide(GasService, {useClass: GasService}),
]);
```

Ebenso kann ein Token auch zu einem einfachen Wert aufgelöst werden (*useValue*):



Die wichtigsten Bausteine für Dependency Injection in Angular 2.0 (Bild 2)

```
var injector = Injector.resolveAndCreate([
  provide('TEST', {useValue: 'Hello Angular2'})
]);
var test = injector.get('TEST');
```

Wie Sie sehen, kann ein Token nicht nur ein Typ, sondern auch ein einfacher String sein. Ebenso findet man auch die aus AngularJS 1.x bekannten Factories wieder (*useFactory*):

```
// a factory can have own dependencies, too
var factory = (gasService: GasService) => {
  return new Dashboard(gasService); // !!
};
var injector = Injector.resolveAndCreate([
  provide(GasService, {useClass: GasService}),
  provide(Dashboard, { useFactory: factory,
    deps: [GasService]})
]);
```

Factories bieten sich immer dann an, wenn das Objekt eine speziellere Initialisierung benötigt. An dieser Stelle sei erwähnt, dass der Injector Klassen lazily instanziert. Die Ob- ►

Listing 1: Asynchrone Programmierung mit Observables

```
// gas-service.ts

import {Injectable} from 'angular2/core';
import {Http} from 'angular2/http';
import Station from './Station';

@Injectable()
export default class GasService {

  apiUrl: string = 'https://creativecommons.
  tankerkoenig.de/json/list.php?
  lat=52.03&lng=13.0&rad=4&sort=price&type=diesel';
  apiKey: string = '&apikey=XXX';

  apiUrlAndKey: string;
  constructor(private http: Http) {
    this.apiUrlAndKey = this.apiUrl + this.apiKey
  }

  getBestPrice() {
    return this.http.get(this.apiUrlAndKey)
      .map(result => (<any>result).json().stations)
      .map((stations: Array<Station>): number =>
        stations[0].price
      )
  }
}
```

jekte werden erst zu dem Zeitpunkt erzeugt, zu dem sie benötigt werden, und anschließend gecacht. Durch das Caching sind alle instanziierten Klassen Singletons. Dies gilt auch für die Rückgabewerte der Factory-Funktion.

Die Methode *get()* verwendet den Cache, die Methode *resolveAndInstantiate()* hingegen nicht – in dem Fall wird auch die Factory ein weiteres Mal aufgerufen. In der Dokumentation auf Angular.io wird daher auch darauf hingewiesen, welche Methoden den Cache nutzen und welche nicht [1].

Weitere Beispiele für die unterschiedlichen Verwendungen von *provide* finden Sie in den Codebeispielen zum Artikel (Datei: *Injector_tests.ts*).

Durchstarten

Die Methode *resolveAndCreate()* vom Injector kann man gut für ein schnelles Experiment oder in einem Unit-Test verwenden. Bei der Erstellung der eigentlichen Anwendung bedient man sich aber der bereits bekannten *bootstrap*-Methode. Zuvor haben wir bei dieser Methode nur den ersten Parameter verwendet. Über den ersten Parameter erwartet Angular die Einstiegs-Komponente der Anwendung – also im vorliegenden Fall die Dashboard-Komponente:

```
// app.ts
import {bootstrap} from 'angular2/angular2';
import Dashboard from './components/dashboard-
component';
```

```
bootstrap(Dashboard);
```

Als zweiten Parameter akzeptiert die Methode wiederum ein Array aus Typen oder Providern.

Sollte die Dashboard-Komponente oder eine andere Komponente den *GasService* benötigen, so lässt sich dieser wie folgt registrieren:

```
// app.ts
import {bootstrap} from 'angular2/angular2';
import Dashboard from
```

```
./components/dashboard-component';
import GasService from './models/gas-service';
```

```
bootstrap(Dashboard, [GasService]);
```

Die Demo-Anwendung muss natürlich noch den aktuell günstigsten Benzinpreis ermitteln können. Zu diesem Zweck verwenden wir die öffentliche Schnittstelle des Verbraucherinformationsdienstes Tankerkönig.de.

Daten per AJAX laden

Der Dienst bietet ein freies JSON-API für die Kraftstoffpreise des Bundeskartellamts an. Hierfür verwenden wir die injizierbare Klasse *http*. Der Rückgabewert der *get*-Methode ist ein Observable-Objekt. Hier kommen wir mit dem neuen Prinzip der reaktiven Programmierung in Berührung, die das Arbeiten mit asynchronen Daten sehr vereinfacht. Für die aktuelle Aufgabe reicht es aus, die eintreffenden Daten mittels *map()* zu transformieren (Listing 1). An anderer Stelle kann dann der ermittelte Preis per *subscribe()* abonniert werden (Listing 2).

Damit ist der Grundstein für das neue Feature gelegt. Der aktuell günstigste Preis liegt vor und kann in der Komponente verwendet werden. Es fehlen noch ein paar Anpassungen am Model und am Templating. Die vollständige Anwendung finden Sie im Codebeispiel zum Artikel [2].

Karma einrichten

Unit-Tests verbessern die Qualität von Software. Tests beweisen, dass die Software das tut, wofür sie konzipiert wurde. Ebenso dokumentieren Tests fachliches Wissen und den Erkenntnisstand eines Entwicklers, den er zum Zeitpunkt der Erstellung hatte. Wenn man als Entwickler das existierende Wissen nicht durch Tests ausdrückt, ist die Wahrscheinlichkeit sehr hoch, dass das Wissen über die Zeit für einen selbst, für das Team und für das Unternehmen verloren geht. Die Verwendung von Angular erweist sich hierbei als großer Vorteil, da das Framework speziell darauf ausgerichtet ist, gut testbare Module zu erstellen.

Listing 2: Mit subscribe das Ergebnis abonnieren

```
// dashboard-component.ts
export default class DashboardComponent {

  constructor(private gasService: GasService) { }

  refillTank(car: Car, amountOfMoneyToSpend: number)
  {

    this.gasService
      .getBestPrice()
      .subscribe((bestPrice: number) => {
        car.refillTank(amountOfMoneyToSpend /
          bestPrice);
      },
      err => console.error(err));
  }
}
```

Um Unit-Tests für JavaScript/TypeScript auszuführen, verwendet man am besten einen sogenannten Test-Runner. Prinzipiell würde auch nur ein Browser ausreichen. Doch dieses Setup lässt sich schlecht automatisieren. Empfehlenswert ist der Test-Runner Karma, der zusammen mit AngularJS von Google entwickelt wurde.

Auf allen gängigen Betriebssystemen

Das Tool basiert auf Node.js und läuft somit auf allen gängigen Betriebssystemen. Erwähnenswert ist die Tatsache, dass Karma einen eigenen Webserver startet und dann einen echten Browser (zum Beispiel den Internet Explorer, Firefox und Chrome) die JavaScript-Dateien ausführen lässt. Der eigene Webserver vermeidet technische Probleme, die man bei der Ausführung durch ein lokales Dateisystem hätte.

Die Installation von Karma nebst Plug-ins geschieht per NPM:

```
npm install karma karma-chrome-launcher karma-jasmine
--save-dev
npm install karma-cli -g
```

Die Datei *package.json* wird dabei um neue *devDependencies* ergänzt. So kann man später per *npm install* das Setup jederzeit wiederherstellen. Die globale Installation des Karma Command Line Interface (*karma-cli*) macht den Kommandozeilen-Befehl *karma* verfügbar.

Unit-Tests starten

Mit *karma start* lassen sich nun die Unit-Tests starten. Beim Überprüfen der Datei *package.json* bietet es sich an, trotz der globalen Installation das Start-Skript auf *karma* startfestzulegen. So kann man später den Testrunner per *npm test* starten. Die Verwendung der *scripts*-Property ist eine empfehlenswerte Konvention in der Node.js-Welt. Mit den Befehlen *npm*

install, *npm start* und *npm test* sollte jeder Node.js-Entwickler vertraut sein.

```
{
  [...]

  "devDependencies": {
    "karma": "^0.13.15",
    "karma-chrome-launcher": "^0.2.1",
    "karma-jasmine": "^0.3.6"
  },

  "scripts": {
    "start": "live-server",
    "test": "karma start"
  }
}
```

Anschließend benötigt das Projekt eine Konfigurationsdatei, die standardmäßig den Namen *karma.conf.js* trägt. Der Befehl *karma init* startet einen Kommandozeilen-Dialog, der bei der Erstellung der Datei hilft. Wie schon bei der Verwendung mit SystemJS/JSPM müssen anschließend noch Pfade gemappt werden.

An dieser Stelle ist das Setup zum aktuellen Stand (Alpha-48) noch etwas unkomfortabel. Wir empfehlen Ihnen aktuell den *ng2-test-seed* [3] von Julie Ralph. Julie Ralph ist eine sehr bekannte Google-Mitarbeiterin, die auch die Hauptentwicklerin des Oberflächen-Test-Tools Protractor ist. Kopieren Sie sich aus dem GitHub-Repository die beiden Dateien *karma.conf.js* und *karma-test-shim.js*.

Die Codebeispiele zum Artikel enthalten ebenso die beiden Dateien. Achten Sie auf die verwendete Ordnerstruktur, sonst funktioniert es nicht. Die Datei *karma-test-shim.js* lädt die Tests per SystemJS. Überprüfen Sie im Fehlerfall in der Datei den Befehl *System.config()*. SystemJS haben wir bereits im ersten Artikel (Ausgabe 12/2015) kennengelernt.

Unit-Tests mit Jasmine

Die Auswahl eines geeigneten Test-Frameworks fällt aktuell sehr leicht. Derzeit wird nur Jasmine [4] vollständig von Angular 2 unterstützt. Jasmine hat eine Syntax im BDD-Stil (Behavior Driven Development). Die Funktion *describe()* definiert eine Sammlung (test suite) zusammenhängender Tests.

Die Funktion erwartet zwei Parameter: Der erste Parameter ist ein String und beschreibt als Wort oder in wenigen kurzen Worten, was gerade getestet wird. Der zweite Parameter ist eine Funktion, die alle Spezifikationen (Specs) beinhaltet. Die *it()* Funktion stellt konkret eine Spezifikation dar. Auch eine Spezifikation benötigt beschreibende Worte.

Describe-Methoden können beliebig tief verschachtelt werden, um die Übersichtlichkeit zu erhöhen. Die eigentlichen Prüfungen erfolgen durch die Funktion *expect()*. Die Funktion *beforeEach* läuft, wie der Name vermuten lässt, stets vor jeder Spezifikation ab. Hier lässt sich doppelter Code beim Initialisieren vermeiden. Der BDD-Stil von Jasmine ermöglicht es, Tests in natürlicher Sprache zu definieren. ►

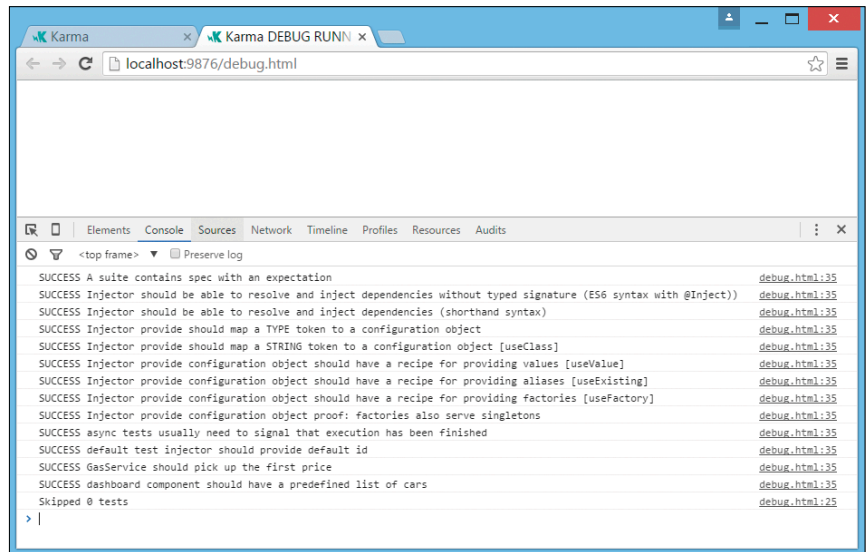
Das folgende Listing veranschaulicht die Syntax:

```
describe("A suite", () => {

  var number;

  beforeEach(() => {
    number = 1;
  });

  it("contains spec with an expectation", () => {
    expect(number).toBeGreaterThan(0);
  });
});
```



Die Methoden von Jasmine (*describe()*, *it*, *expect* und so weiter) sind nicht neu. In jedem Unit-Test stehen diese Methoden

seit jeher im globalen Gültigkeitsbereich zur Verfügung. Wir empfehlen aber, für einen Angular-2-Test die globalen Methoden nicht direkt zu verwenden. Angular bietet dieselben Methoden über einen Import an (*angular2/testing*).

Das Angular-Testing wrappt die Methoden und fügt neue Matcher hinzu. Durch diese Manipulation von Jasmine wird das Injizieren von Abhängigkeiten in Tests vereinfacht und das Testen von asynchronem Code ermöglicht. Zudem erhält man auch gleich die passenden TypeScript Type Definitions.

Komponenten testen

Das Modul Angular-Testing bietet eine neue Methode an, die das Setup eines Unit-Tests sehr komfortabel gestaltet. Zu der

Listing 3: Einsatz von *beforeEachProviders()* und *inject()*

```
import { it, describe, expect, inject,
  beforeEachProviders } from 'angular2/testing';
import { HTTP_PROVIDERS } from 'angular2/http';

import DashboardComponent from '../app/
components/dashboard-component';
import GasService from '../app/models/
gas-service';

describe('dashboard component', () => {
  beforeEachProviders(() => [DashboardComponent,
    GasService, HTTP_PROVIDERS]);

  it('should have a predefined list of cars',
    inject([DashboardComponent], (dashboard:
    DashboardComponent) => {
      expect(dashboard.cars.length).toBe(2);
    }));
});
```

bereits bekannten Methode *beforeEach* gesellt sich nun die Methode *beforeEachProviders*. Mit dieser Methode kann man vor der eigentlichen Ausführung des Tests den Injector mit Providern befüllen beziehungsweise bestehende Provider überschreiben. Es lassen sich hierbei auch Kern-Funktionalitäten von Angular überschreiben.

Wie bei den anderen Methoden zum DI-System akzeptiert *beforeEachProviders()* ein Array aus Typen oder Providern. Der Test aus Listing 3 beweist zum Beispiel, dass die Dashboard-Komponente stets mit einem gefüllten Array initialisiert wird.

Beachten Sie auch die Verwendung der Methode *inject()*. Sie ist dazu gedacht, in einer *beforeEach()* oder *it()* eine Abhängigkeit anzufordern. In der Quelltext-Dokumentation von Angular findet sich der Hinweis, dass es gegebenenfalls in Zukunft noch eine Syntax mit Decoratoren geben wird (Bild 3):

```
// aktuell
inject([DashboardComponent], (dashboard:
DashboardComponent) => { /* [...] */ })

// mögliche zukünftige Syntax
@Inject(dashboard: DashboardComponent) => { ... }
```

Oft muss man in der JavaScript-Welt auf etwas warten. Dies kann unter anderem die Antwort auf einen AJAX-Call sein, wie es auch unsere Anwendung bei den Preisdaten tut. Um das Problem abzubilden, verwendet man üblicherweise Callbacks, Promises oder Observables. Allen Herangehensweisen ist gemein, dass der ausgeführte Code asynchron abläuft. Der Ansatz in Angular 1.x war es, asynchronen Code in ein synchrones Format zu pressen.

Man musste dazu in Unit-Tests speziell Rücksicht nehmen und zum Beispiel mit *\$rootScope.\$digest()*, *\$httpBackend.flush()* oder *\$timeout.flush()* manuell die Promises erfüllen, um anschließend das Ergebnis überprüfen zu können. Dieser Ansatz ermöglicht zunächst elegante und leicht verständliche

Listing 4: Ein asynchroner Unit-Test, der nicht mehr manuell abgeschlossen werden muss

```

import { beforeEachProviders, describe, expect,
inject, it } from 'angular2/testing';
import { provide } from 'angular2/angular2';
import { MockBackend, BaseRequestOptions, Http,
Response, ResponseOptions } from 'angular2/http';
import GasService from '../app/models/gas-service';

describe('GasService', () => {

  beforeEachProviders(() => [
    BaseRequestOptions, MockBackend,
    provide(Http, {
      useFactory: (backend, defaultOptions) =>
        new Http(backend, defaultOptions),
      deps: [MockBackend, BaseRequestOptions]
    }),
    GasService
  ]);

  it('should pick up the first price',
    inject([GasService, MockBackend], (gasService,
    backend) => {

      backend.connections.subscribe(c => {
        c.mockRespond(new Response(
          new ResponseOptions({body: '{ "stations": [{
            "price": 42 }, { "price": 4 }] }'))));
      });

      gasService.getBestPrice().subscribe((price) => {
        expect(price).toBe(42);
      });
    });
  });
});

```

Tests. Doch gerade dieser Ansatz kann bei komplexeren Aufgabenstellungen eine Lösung erschweren, da das eigentliche Problem hinter der synchronen Fassade versteckt wird.

Angular 2 birgt das Potenzial, an dieser Stelle um einiges einfacher zu werden. Eine vergleichbare synchrone Fassade existiert nicht mehr. Daher müsste ein Test für asynchrone Methoden eigentlich stets wie folgt aussehen:

```

describe('async tests', () => {
  it('usually need to signal that execution has been
  finished', (done) => {
    setTimeout(() => {
      expect(true).toBe(true);
      done(); // !!!
    }, 500)
  });
});

```

Seit Dezember 2015 (Alpha-47) ist jedoch ein sehr interessantes Feature in Angular vorhanden. Angular-Testing verwendet *zones.js* und Microtasks, um selbstständig festzustellen, wann der Unit-Test abgeschlossen ist. Ein Beispiel für einen Unit-Test für asynchronen Code ist das **Listing 4**.

Zunächst muss der HTTP-Service ausgemockt werden. Dies geschieht mit der bekannten Methode *provide()*. Mittels *subscribe()* können wir den Output des *GasService* empfangen. Der *GasService* wird dabei nicht gegen einen echten HTTP-Endpunkt gehen, sondern die gemockte Verbindung verwenden. Es fällt auf, dass ein Aufruf des *Callbacks done()* nicht notwendig ist – dies erledigt Angular für uns.

Fazit

Mit Version 2 hat das Angular-Team auf viele technische Schwächen von AngularJS reagiert. Das neue Konzept wirkt wie aus einem Guss: Typen und Decoratoren durch Type-

Script, modularer Code durch SystemJS und auf Basis dessen ein generalüberholtes DI-System. Das neue Prinzip leuchtet schnell ein und die Testbarkeit profitiert von dieser neuen Umsetzung. Die neuen Provider begegnen uns an vielen Stellen bei der täglichen Arbeit mit Angular 2. Ein Grundverständnis der dahinterliegenden Mechanismen erleichtert den Einstieg entsprechend signifikant. ■

- [1] Angular 2 for TypeScript: Injector,
<https://angular.io/docs/ts/latest/api/core/Injector-class.html>
- [2] Codebeispiel zum Artikel, <https://github.com/Angular2Buch/angular2-testing>
- [3] ng2-test-seed von Julie Ralph,
<https://github.com/juliemr/ng2-test-seed>
- [4] Jasmine (Modultest-Bibliothek für JavaScript),
<http://jasmine.github.io>



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>



Gregor Woiwode

ist als Software-Entwickler im Bereich des Competitive Intelligence beziehungsweise Enterprise Knowledge Managements für ein Software-Unternehmen in Leipzig tätig. Er veranstaltet Trainings zu AngularJS.

www.woiwode.info/blog

SPEICHERVERWALTUNG MIT ARC IN SWIFT

Automatic Reference Counting

Trotz Automatismen wie ARC müssen Swift-Programmierer auf einige Dinge achten.

Speicherverwaltung ist bisweilen ein leidiges Thema unter Entwicklern, ganz gleich um welche Programmiersprache es sich handelt. Kein Wunder: Speicherverwaltung in ihrer Reinform ist durchaus ein komplexes Thema und bisweilen alles andere als leicht zu verstehen. Doch umso größer sind die Konsequenzen, wendet man Speicherverwaltung falsch beziehungsweise gar nicht in den eigenen Anwendungen an.

Swift stellt dabei in Sachen Speicherverwaltung keine Ausnahme unter den Programmiersprachen dar. Die Technik, die zu diesem Zweck in Swift zum Einsatz kommt, ist wie auch bei Objective-C das sogenannte Reference Counting. Dabei werden Verweise auf Objekte im Speicher gezählt. Gibt es keine Verweise mehr auf ein Objekt, wird es entsprechend freigegeben und so aus dem Speicher entfernt.

Mit der Einführung des sogenannten Automatic Reference Counting (kurz ARC) hat Apple diesen Prozess deutlich vereinfacht. Damit ist es nicht mehr an uns, den Referenzzähler auf ein Objekt selbst zu erhöhen beziehungsweise zu verringern. Das geschieht automatisch im Hintergrund, ohne dass wir dafür auch nur eine Zeile Code schreiben müssen.

Ein einfaches ARC-Beispiel

Um das Verhalten von ARC im Speziellen und der Speicherverwaltung in Swift im Allgemeinen zu demonstrieren, soll einmal die in [Listing 1](#) deklarierte Klasse *Vehicle* herhalten.

In [Listing 2](#) werden sodann drei Variablen erzeugt, die vom Typ *Vehicle?* sind und denen kein Initialwert zugewiesen wird. Sie zeigen somit auf kein Objekt und entsprechen stattdessen *nil*.

Nun erzeugen wir eine neue *Vehicle*-Instanz und weisen diese der eben deklarierten Variable *vehicleReference1* zu:

```
vehicleReference1 = Vehicle(manufacturer: "Opel")  
// Vehicle-Objekt ist initialisiert.
```

Durch das Erzeugen der neuen *Vehicle*-Instanz wird auch die *print*-Anweisung ausgeführt, die Teil des Initializers der *Vehicle*-Klasse ist und die Info ausgibt, dass ein neues *Vehicle*-Objekt erzeugt wurde.

Die Variable *vehicleReference1* hält nun eine sogenannte starke Referenz auf das erzeugte *Vehicle*-Objekt. Das bedeutet, dass das *Vehicle*-Objekt so lange nicht freigegeben wird, wie *vehicleReference1* darauf verweist. So lange bleibt das Objekt im Speicher erhalten. Nun wird den beiden anderen

Listing 1: Deklaration der Klasse Vehicle

```
class Vehicle {  
    let manufacturer: String  
    init(manufacturer: String) {  
        self.manufacturer = manufacturer  
        print("Vehicle-Objekt ist initialisiert.")  
    }  
    deinit {  
        print("Vehicle-Objekt ist deinitialisiert.")  
    }  
}
```

Listing 2: Deklaration von drei Vehicle-Variablen

```
var vehicleReference1: Vehicle?  
var vehicleReference2: Vehicle?  
var vehicleReference3: Vehicle?
```

Variablen *vehicleReference2* und *vehicleReference3* ebenfalls dasjenige *Vehicle*-Objekt zugewiesen, auf das *vehicleReference1* verweist:

```
vehicleReference2 = vehicleReference1  
vehicleReference3 = vehicleReference1
```

Damit besitzt das zu Beginn erzeugte *Vehicle*-Objekt nun insgesamt drei starke Referenzen, die darauf verweisen. Selbst wenn nun also zwei dieser starken Referenzen verloren gehen, bleibt das *Vehicle*-Objekt noch immer im Speicher erhalten, da wenigstens eine starke Referenz darauf existiert. Das gilt selbst für die Variable *vehicleReference1*, der das *Vehicle*-Objekt initial zugewiesen wurde. Im Folgenden werden zwei dieser drei Referenzen aufgelöst:

```
vehicleReference1 = nil  
vehicleReference2 = nil
```

Da das zu Beginn erzeugte *Vehicle*-Objekt nun noch immer von *vehicleReference3* stark referenziert wird, bleibt es im

Listing 3: Deklaration einer Vehicle- und Driver-Klasse

```

class Vehicle {
    let manufacturer: String
    var driver: Driver?
    init(manufacturer: String) {
        self.manufacturer = manufacturer
    }
    deinit {
        print("Vehicle-Objekt ist deinitialisiert.")
    }
}

class Driver {
    let name: String
    var vehicle: Vehicle?
    init(name: String) {
        self.name = name
    }
    deinit {
        print("Driver-Objekt ist deinitialisiert.")
    }
}

```

Speicher erhalten. Erst wenn auch dieser Verweis verschwindet, gibt es keine Referenz mehr auf das *Vehicle*-Objekt und es wird aus dem Speicher entfernt:

```

vehicleReference3 = nil
// Vehicle-Objekt ist deinitialisiert.

```

Zur Bestätigung wird nach Verschwinden des letzten Verweises auf das *Vehicle*-Objekt auch die passende *print*-Meldung ausgegeben, die innerhalb des Deinitializers der *Vehicle*-Klasse implementiert ist.

Strong References

Im vorangegangenen Abschnitt war sehr häufig die Rede von sogenannten starken Referenzen (Strong References). Starke Referenzen bezeichnen einen Verweis auf ein Objekt im Speicher, der den Referenzzähler des betreffenden Objekts erhöht, das Objekt also somit stark im Speicher hält. Solange mindestens eine solche Referenz auf einem Objekt im Speicher existiert, wird dieses auch nicht aus dem Speicher entfernt. Das ist die Grundlage des Reference Counting.

Diese Form von Referenzen gibt es in Swift ausschließlich bei Objekten, die von Klassen abgeleitet sind, denn nur Klassen sind in Swift sogenannte Reference Types. Demgegenüber stehen die sogenannten Value Types, wie sie in Swift mittels Enumerations oder Structures umgesetzt werden können.

Der Unterschied zwischen diesen beiden besteht darin, dass Objekte von Value Types bei der Zuweisung zu einer neuen Variable, Konstante oder einem Parameter immer kopiert werden und somit also ein neues Objekt erzeugt wird.

Bei Reference Types hingegen wird lediglich eine Referenz auf das im Speicher liegende Objekt weitergegeben, wodurch Variablen, Konstanten oder Parameter, denen ein Objekt eines Reference Type zugewiesen wird, auf exakt dasselbe Objekt verweisen und eine Änderung des Objekts an der einen Stelle auch direkte Auswirkungen auf alle Referenzen des entsprechenden Objekts an allen anderen Stellen nach sich zieht.

Strong Reference Cycles

Möglicherweise fragen Sie sich nun an dieser Stelle, wo eigentlich genau das Problem liegt. Wie wir gesehen haben, müssen wir in Swift keinerlei Code schreiben, um den Referenzzähler auf ein Objekt zu erhöhen oder zu verringern; das geschieht vollautomatisch und Swift gibt selbsttätig Objekte im Speicher frei, sobald diese nicht mehr benötigt werden.

Tatsächlich gibt es aber ein Problem, bei dem ARC nicht weiterhelfen kann und bei dem wir Entwickler verantwortlich sind, es korrekt zu lösen. Die Rede ist dabei von den sogenannten Strong Reference Cycles. Bei einem Strong Reference Cycle referenzieren sich zwei Objekte gegenseitig mit einer starken Referenz und werden somit niemals automatisch freigegeben und aus dem Speicher entfernt. Zur Verdeutlichung dieser Problematik soll ein weiteres Beispiel herhalten. Listing 3 zeigt die dazugehörige Basis, bestehend aus einer *Vehicle*- und einer *Driver*-Klasse.

Beide Klassen verfügen über einen eigenen Initializer sowie Deinitializer und besitzen jeweils eine optionale Property, die auf ein Objekt der jeweils anderen Klasse verweist. So verfügt eine *Vehicle*-Instanz über einen optionalen Fahrer und jedem Fahrer kann optional ein Fahrzeug zugewiesen werden.

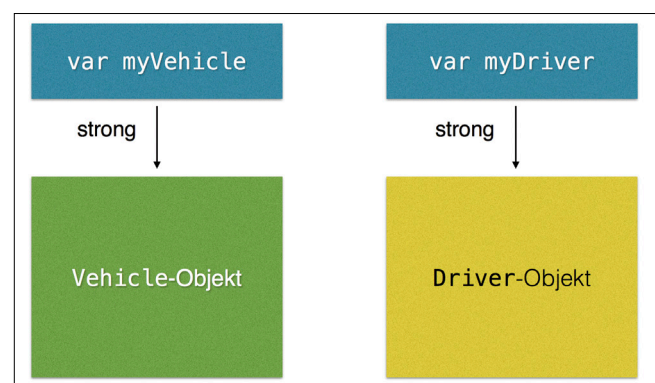
Nun werden im nächsten Schritt zwei Variablen erzeugt; die eine ist vom Typ *Vehicle?*, die andere vom Typ *Driver?*. Beiden wird initial bereits ein passendes Objekt der jeweiligen Klasse zugewiesen:

```

var myVehicle: Vehicle? = Vehicle(manufacturer: "Opel")
var myDriver: Driver? = Driver(name: "Thomas Sillmann")

```

Somit haben wir an dieser Stelle zwei starke Referenzen erzeugt: Die eine verweist auf das erzeugte *Vehicle*-Objekt ►



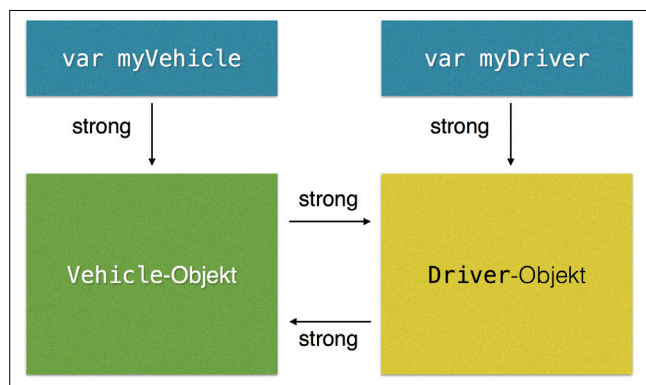
Zwei Variablen verweisen auf je ein Objekt im Speicher (Bild 1)

und geht von der Variablen *myVehicle* aus, die andere verweist auf das erzeugte *Driver*-Objekt und geht von der Variablen *myDriver* aus. **Bild 1** verdeutlicht dieses Verhältnis der Objekte und ihrer Referenzen.

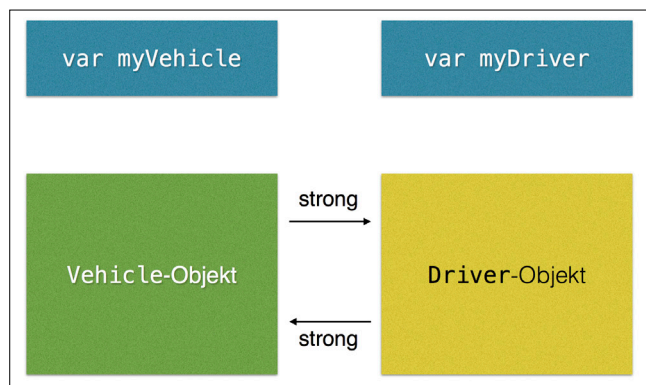
Im nächsten Schritt weisen wir nun dem erzeugten *Vehicle*-Objekt einen passenden Fahrer über dessen Property *driver* zu. Dazu verwenden wir das ebenfalls erzeugte *Driver*-Objekt. Umgekehrt weisen wir aber auch dem *Driver*-Objekt das Fahrzeug über die zugehörige Property *vehicle* zu:

```
myVehicle!.driver = myDriver
myDriver!.vehicle = myVehicle
```

Dieser Code an sich ist vollkommen in Ordnung und legitim, schließlich weisen wir einfach zwei Properties passende Referenzen auf ein Objekt zu. Das hat aber in diesem Fall zur



Zwei Objekte referenzieren sich selbst gegenseitig mit einer starken Referenz (**Bild 2**)



Referenzen: Selbst wenn die starken Referenzen der Variablen verschwinden, bleiben noch immer die gegenseitigen starken Referenzen und halten die Objekte im Speicher (**Bild 3**)

Folge, das sowohl das *Vehicle*- als auch das *Driver*-Objekt sich gegenseitig stark referenzieren. **Bild 2** illustriert diesen Zwischenstand.

Und nun haben wir nämlich den Salat: Ein Strong Reference Cycle ist geschaffen, zwei Objekte verweisen gegenseitig mit einer starken Referenz auf sich. Warum das ein Pro-

Listing 4: Deklaration der Strong und Weak Reference

```
var anotherVehicle: Vehicle? = Vehicle()
var strongReference: Vehicle? = anotherVehicle
weak var weakReference: Vehicle? = anotherVehicle
```

blem ist, zeigt der folgende Code, der unsere Variablen *myVehicle* und *myDriver* auf *nil* setzt:

```
myVehicle = nil
myDriver = nil
```

Auch dieser Code ist an sich vollkommen korrekt und legitim. Durch die Zuweisung von *nil* verschwinden die starken Referenzen auf das erzeugte *Vehicle*- und *Driver*-Objekt vonseiten der Variablen *myVehicle* und *myDriver*. Doch zwei Referenzen bleiben übrig: Die beiden Objekte referenzieren sich zu diesem Zeitpunkt noch immer gegenseitig selbst. **Bild 3** verdeutlicht diese Situation.

Das Problem ist nun, dass die erzeugten *Vehicle*- und *Driver*-Objekte weiterhin im Speicher leben. Warum? Weil sie sich gegenseitig stark referenzieren und damit den Referenzzähler auf eins halten. Und wir wissen ja, dass Objekte erst aus dem Speicher freigegeben werden, wenn keine Referenz mehr darauf existiert. Das wird aber bei unseren beiden Objekten nun niemals der Fall sein. Und genau das ist der Fallstrick, den wir Entwickler bei der Speicherverwaltung mit Swift berücksichtigen müssen.

Weak References

Die Lösung für unser Problem lautet in diesem Fall Weak References, also schwache Referenzen auf ein Objekt. Dabei sind Weak References zunächst einmal den Strong References sehr ähnlich: Auch diese verweisen auf ein Objekt eines Reference Type.

Der große Unterschied besteht aber darin, dass eine Weak Reference nicht den Referenzzähler des Objekts, auf das sie zeigt, erhöht. Diese Referenz spielt letzten Endes für das Referenzzählen mittels ARC keine Rolle.

Diese Tatsache bedingt aber auch, dass Weak References immer als Variablen und als Optionals deklariert werden müssen. Warum? Nun, da eine Weak Reference das Objekt, auf das sie zeigt, nicht selbst fest im Speicher hält, kann es folglich passieren, dass das betroffene Objekt irgendwann nicht mehr existiert und die entsprechende Referenz somit stattdessen auf *nil* zeigt.

Und *nil* können in Swift bekanntermaßen nur Optionals annehmen. Und aufgrund dessen, dass Weak References immer auf andere Objekte im Speicher verweisen, ohne sie selbst zu halten, kann es natürlich auch passieren, dass sich dieses Objekt im Lauf der Zeit verändert, daher die notwendige Deklaration einer Weak Reference als Variable.

Um eine Weak Reference zu erstellen braucht es das Schlüsselwort *weak*. Wird dieses einer Property vorangestellt, so bedeutet das, dass der Wert, der dieser Property zugewie-

Listing 5: Weak Reference für Vehicle und Driver

```
class Vehicle {
    let manufacturer: String
    weak var driver: Driver?
    init(manufacturer: String) {
        self.manufacturer = manufacturer
    }
    deinit {
        print("Vehicle-Objekt ist deinitialisiert.")
    }
}

class Driver {
    let name: String
    var vehicle: Vehicle?
    init(name: String) {
        self.name = name
    }
    deinit {
        print("Driver-Objekt ist deinitialisiert.")
    }
}
```

sen wird, von dieser Property nicht fest im Speicher gehalten wird. Der Referenzzähler wird also bei Zuweisung eines Objekts zu dieser Property nicht erhöht. Listing 4 zeigt einmal beispielhaft die Deklaration einer Strong- und einer Weak-Reference im Vergleich.

Obwohl in Listing 4 insgesamt drei Properties auf die neu erzeugte *Vehicle*-Instanz verweisen (nämlich die Variablen *anotherVehicle*, *strongReference* und *weakReference*), so beträgt der Referenzzähler in diesem Fall lediglich zwei. *weakReference* verweist zwar ebenfalls auf das Objekt, hält es aber nicht im Speicher. Folglich wird das *Vehicle*-Objekt bereits dann aus dem Speicher entfernt, sobald die Referenz der beiden Strong References von *anotherVehicle* und *strongReference* nicht mehr existiert:

```
anotherVehicle = nil
strongReference = nil
// Vehicle-Objekt ist deinitialisiert.
```

In diesem Moment verweist auch *weakReference* umgehend auf *nil*; das Objekt, auf das es verweist, existiert nun nämlich nicht mehr im Speicher.

Weak References gegen Strong Reference Cycles

Mit diesem Wissen im Gepäck ist es uns nun möglich, das Beispiel des Strong Reference Cycle zu durchbrechen. Dabei gilt es, bei zwei Typen, die sich mit Properties gegenseitig referenzieren, zu überlegen, welche Referenz optional ist und keinen starken Verweis auf das jeweils andere Objekt benötigt. Nehmen wir als Beispiel wiederum unsere *Vehicle*- und *Driver*-Klasse.

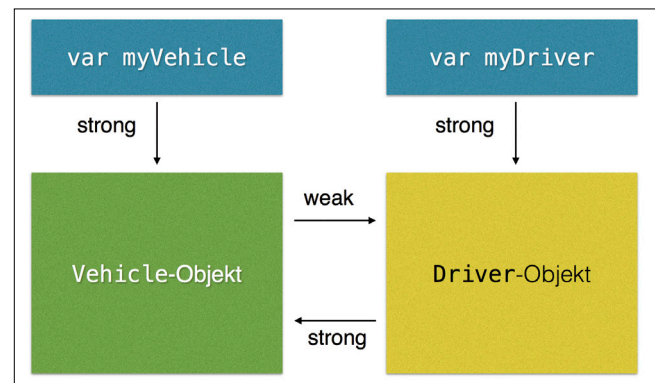
Hier können wir davon ausgehen, dass jeder Fahrer auch immer über ein passendes Fahrzeug verfügt, aber nicht jedes Fahrzeug zu jedem Zeitpunkt einen Fahrer besitzt. Entsprechend wäre die Property *driver* der *Vehicle*-Klasse genau die Stelle, an der wir den Strong Reference Cycle mit Hilfe einer Weak Reference durchbrechen können. Listing 5 zeigt dabei noch einmal die Deklaration und Implementierung der beiden Klassen, dieses Mal ist aber die *driver*-Property der *Vehicle*-Klasse mittels *weak* als Weak Reference deklariert.

Mit dieser kleinen Änderung in der *Vehicle*-Klasse bleibt unser übriger Code wie gehabt; es wird je eine Instanz der beiden Klassen erstellt, die sich gegenseitig referenzieren:

```
var myVehicle: Vehicle? = Vehicle(manufacturer: "Opel")
var myDriver: Driver? = Driver(name: "Thomas Sillmann")

myVehicle!.driver = myDriver
myDriver!.vehicle = myVehicle
```

Während der Code äußerlich an dieser Stelle genau gleich geblieben ist, hat sich intern etwas getan. Bild 4 zeigt, wie die Objekte und Referenzen nun zueinander stehen. So verweisen die beiden Variablen *myVehicle* und *myDriver* noch im-



Das Vehicle-Objekt besitzt nun nur noch eine schwache Referenz auf die *Driver*-Instanz (Bild 4)

mer – wie gewünscht – mit starken Referenzen auf die neu erzeugten Objekte, ebenso zeigt auch das *Driver*-Objekt selbst mit einer starken Referenz auf das zugewiesene *Vehicle*.

Aber die Referenz des *Vehicle*-Objekts in Bezug auf das *Driver*-Objekt hat sich nun geändert, diese ist nun *weak* und nicht mehr *strong*. Das bedeutet, dass die Instanz der Klasse *Vehicle* zwar auf das *Driver*-Objekt verweist, dieses aber nicht im Speicher hält.

Was bedeutet das aber nun konkret? Spielen wir dieses Beispiel einmal weiter durch und lösen die starke Referenz der Variablen *myDriver* zum *Driver*-Objekt auf:

```
myDriver = nil
```

Bild 5 zeigt, wie die Referenzen zwischen den Objekten nach dieser Aktion aussehen. Das *Driver*-Objekt zeigt zwar ►

noch mittels Strong Reference auf die *Vehicle*-Instanz, aber es gibt umgekehrt keine einzige Strong Reference mehr auf das *Driver*-Objekt.

Und sobald es keine Strong References mehr auf ein Objekt gibt, entspricht das einem Referenzzähler von null. Folglich wird nun das *Driver*-Objekt aus dem Speicher entfernt. Zur Bestätigung wird auch die *print*-Meldung des Deinitializers der *Driver*-Klasse ausgegeben:

```
// Driver-Objekt ist deinitialisiert.
```

Und was geschieht nun, wenn die Strong Reference zum *Vehicle*-Objekt unserer Variablen *myVehicle* wie im Folgenden gezeigt entfernt wird?

```
myVehicle = nil
```

Bild 6 zeigt den entsprechenden Zustand der Referenzen zwischen den Objekten.

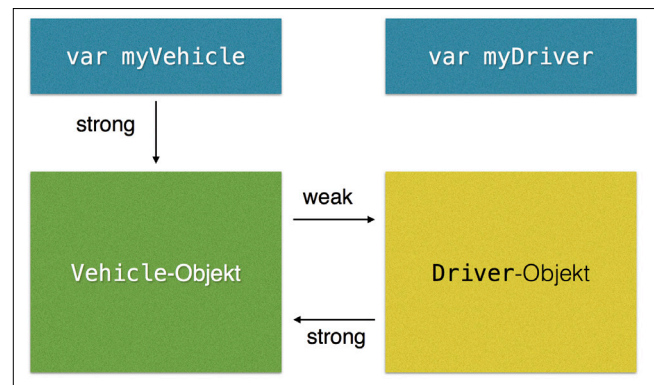
Nun existieren auch keinerlei Strong References mehr auf das *Vehicle*-Objekt, weshalb auch dieses nun erfolgreich aus dem Speicher freigegeben wird:

```
// Vehicle-Objekt ist deinitialisiert.
```

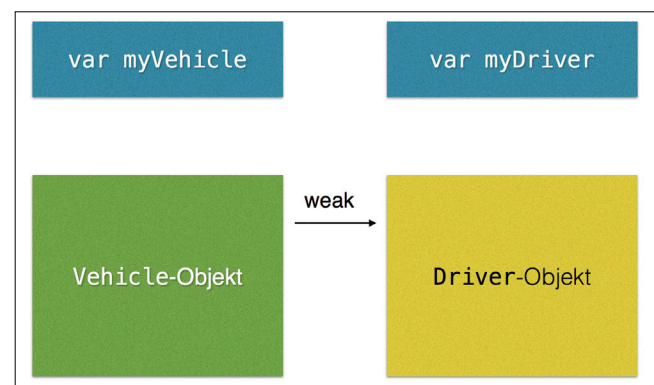
Somit haben wir durch die kleine Änderung einer Strong in eine Weak Reference dafür gesorgt, dass alle Objekte korrekt und zum passenden Zeitpunkt freigegeben werden.

Unowned References

Neben den bisher gezeigten Strong und Weak References gibt es in Swift noch eine dritte Form, die sogenannten Unowned References. Diese sind vom Prinzip her mit den Weak References identisch; sie erhöhen nicht den Referenzzähler des zugewiesenen Objekts. Allerdings sind Unowned References für Properties gedacht, die immer über einen validen Wert



Driver-Objekt: Es existiert keine Strong Reference mehr auf das Driver-Objekt (**Bild 5**)



Alle Strong References wurden erfolgreich aufgelöst (**Bild 6**)

verfügen sollen und somit nicht optional sind, und genau dahingehend unterscheiden sie sich von den Weak References.

Diese werden immer als Optionals deklariert und können somit auch *nil* entsprechen. Unowned References aber sind keine Optionals und müssen folgerichtig immer auf ein valides Objekt im Speicher zeigen, auch wenn sie dieses selbst nicht stark im Speicher halten.

Typisches Beispiel

Ein typisches Beispiel für die Verwendung einer Unowned Reference anstelle einer Weak Reference zeigt **Listing 6**.

Dort werden zwei Klassen deklariert: *Person* und *Company*. Eine *Person* setzt sich aus einem Namen sowie einer optionalen *Company* zusammen, also einer Firma, in der die betreffende *Person* arbeitet. Die *Company* wiederum besitzt ebenfalls einen Firmennamen sowie einen Manager. Der Manager ist dabei nicht optional; es muss immer einen Manager geben, und dieser ist vom Typ *Person*. Hier kann es also bei dieser Konstellation erneut zu einem Strong Reference Cycle zwischen den Klassen *Person* und *Company* kommen, falls diese sich gegenseitig stark referenzieren.

Um das Problem zu lösen, ist die *manager*-Property der *Company*-Klasse als Unowned Reference deklariert. Das bedeutet, dass sie genau wie eine Weak Reference keine starke Referenz auf das zugewiesene Objekt hält, stellt umgekehrt aber sicher, dass *manager* immer über einen validen Wert

Listing 6: Verwendung einer Unowned Reference

```
class Person {
    let name: String
    var company: Company?
    init(name: String) {
        self.name = name
    }
}

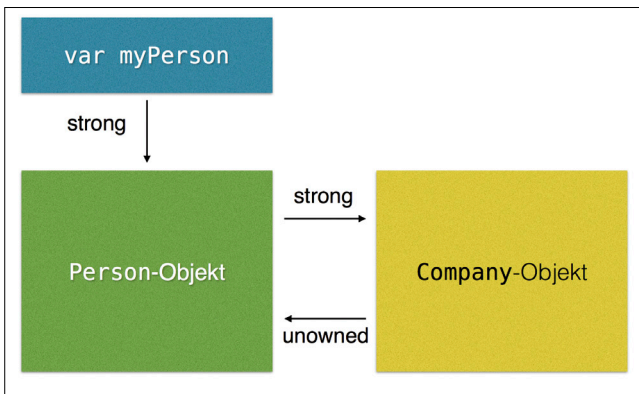
class Company {
    let companyName: String
    unowned let manager: Person
    init(companyName: String, manager: Person) {
        self.companyName = companyName
        self.manager = manager
    }
}
```


verfügen muss und nicht *nil* sein darf. Würde die *manager*-Property zu einem bestimmten Zeitpunkt dennoch fälschlicherweise auf *nil* verweisen und würde man versuchen, auf diese Property zuzugreifen, so hätte das umgehend einen Absturz unserer Anwendung zur Folge.

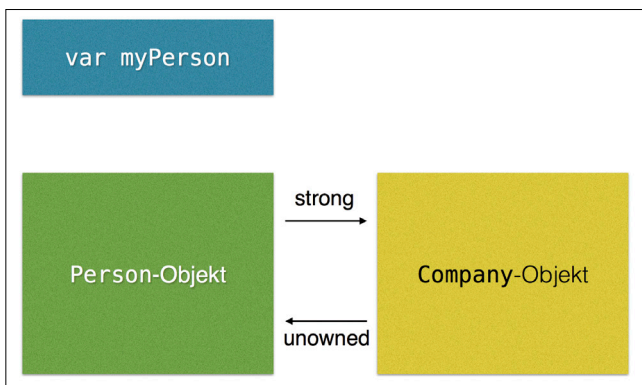
Um diese Konstellation der beiden Klassen *Person* und *Company* nun noch einmal zu verdeutlichen, wird im Folgenden eine Instanz der Klasse *Person* erstellt und ihr eine Firma zugewiesen, in der eben diese Person als Manager agiert:

```
var myPerson: Person? = Person(name: "Thomas Sillmann")
myPerson!.company = Company(companyName: "Max Mustermann GmbH", manager: myPerson!)
```

Bild 7 illustriert die durch diesen Code erzeugten Verbindungen und Referenzen zwischen dem *Person*- und *Company*-



Die Unowned Reference stellt sicher, das *Company* immer auf ein valides *Person*-Objekt verweisen wird (**Bild 7**)



Strong Reference Cycle: Dank der Unowned Reference kommt es an dieser Stelle nicht zu einem Strong Reference Cycle (**Bild 8**)

Objekt. So hält die Variable *myPerson* eine starke Referenz auf die *Person*-Instanz und dieses Objekt wiederum hält eine starke Referenz auf die erzeugte *Company*-Instanz. Lediglich das *Company*-Objekt verweist mittels einer schwachen Unowned Reference wiederum auf die erzeugte *Person*.

Sollte nun die starke Referenz auf die *Person*-Instanz ausgehend von der *myPerson*-Variable entfernt werden, ergibt

Links zum Thema

- Apple Developer Portal
<https://developer.apple.com>
- Swift-Seite von Apple
<https://developer.apple.com/swift>
- Swift Community Site
<https://swift.org>
- Swift – ARC Overview
www.tutorialspoint.com/swift/swift_automatic_reference_counting.htm
- Advanced Memory Management Programming Guide
<https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/MemoryMgmt/Articles/MemoryMgmt.html>

sich eine Situation wie in **Bild 8** gezeigt. Würden die beiden Objekte im Speicher sich gegenseitig stark referenzieren, würde es hier erneut zu einem Strong Reference Cycle kommen. So aber wird zunächst das *Person*-Objekt freigegeben, da dessen Referenzzähler nun 0 beträgt und es keine Strong References mehr auf das Objekt gibt. Dadurch verschwindet auch die einzig verbliebene Strong Reference auf die *Company*-Instanz, wodurch diese im Anschluss ebenfalls aus dem Speicher entfernt und dieser wieder freigegeben wird.

Fazit

Speicherverwaltung ist heutzutage ein Thema, das im täglichen Entwickleralltag immer weniger präsent ist. Da moderne Programmiersprachen, Compiler und Tools Entwicklern viel grundlegende Arbeit in Bezug auf die Speicherverwaltung abnehmen, gibt es kaum noch Dinge, um die man sich als Entwickler selbst kümmern muss.

Bei der Programmierung mit Swift sind die sogenannten Strong Reference Cycles aber unbedingt zu beachten. Diese verursachen trotz ARC Speicherlecks und führen im schlimmsten Fall zum Absturz der eigenen App. Wann immer zwei Objekte unterschiedlicher Typen sich gegenseitig referenzieren, sollten bei jedem Entwickler die Alarmglocken schrillen. Dann sind in der Regel immer Weak- oder Unowned-References notwendig, um den Strong Reference Cycle zu durchbrechen und die Stabilität der eigenen App sicherzustellen. ■



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.



Foto: Shutterstock/Maxx-Grafik

QR-CODES MIT SWIFT

Quick Response

Mit den Bordmitteln von iOS QR-Codes lesen und erzeugen.

An Plakatwänden sieht man sie, in Zeitschriften und auch im Internet sind sie omnipräsent: QR-Codes (Quick Response) haben sich in unserer Gesellschaft ihren Platz erarbeitet. Ein Smartphone mit Kamera hat fast jeder in seiner Tasche, und um gezielt auf Informationen (zum Beispiel im Internet) zu verweisen, sind QR-Codes ein überaus praktisches Mittel. Auch in der eigenen App kann das Erzeugen eines QR-Codes oder auch das Einlesen desselben äußerst nützlich sein. Der Artikel zeigt, wie QR-Codes mit iOS sowohl erzeugt als auch gelesen werden können.

Vor iOS 7 war die Verarbeitung von QR-Codes ausschließlich mit Hilfe von externen Bibliotheken möglich. Seit Version 7 gehören Klassen, die das Lesen und Erzeugen von QR-Codes ermöglichen, zu den Bordmitteln von iOS.

Zunächst sollen die benötigten Schritte vorgestellt werden, um mit Swift QR-Codes selbst zu erzeugen. Dazu wird ein neues Projekt vom Typ *Single View Application* angelegt. Auf der View werden neben einem *TextField*-Control noch ein Button sowie ein *ImageView*-Control eingefügt.

Im *TextField*-Control soll die als QR-Code zu verschlüsselnde Zeichenkette eingegeben werden. Der Button dient als Auslöser, um den benötigten Code zur Umwandlung aufzurufen, während das *ImageView*-Control verwendet wird, um das generierte QR-Code-Image dem Anwender anzeigen zu können.

Creating Codes

Am Ende der Designphase sollte die Oberfläche dann aussehen wie in **Bild 1**. Im Anschluss kann man sich der Programmierung des Codes widmen. Zuerst sollten die benötigten Referenzen, Outlets für *TextField* und *ImageView* sowie der Ereignis-Handler für den Button (Action), angelegt werden. Neben den Referenzen und dem Ereignis-Handler wird außerdem noch eine Variable benötigt, in der das QR-Code-Image zwischengespeichert wird. Diese Variable trägt den Namen *qrcImage* und ist vom Typ *UIImage*.

Um den QR-Code erzeugen zu können, muss natürlich zuvor etwas Code geschrieben werden. Der erste Teil des benö-

tigten Codes wird im Ereignis-Handler *buttonPressed* hinterlegt (Listing 1).

Der Code beginnt mit einer Prüfung. Es wird abgefragt, ob in der Variablen *qrcCIImage* bereits ein Wert gespeichert wurde oder nicht. Nur wenn das nicht der Fall ist, soll es an die Erstellung des QR-Codes gehen.

Als Nächstes wird geprüft, ob ein Text in das TextField-Control eingegeben worden ist. Denn was als QC-Code codiert werden soll, muss ja auch vorhanden sein. Der Inhalt des TextField-Controls wird anschließend in der Variablen *data*, welche vom Typ *NSData* ist, gespeichert. Vor der Zuweisung findet noch eine Encodierung mit dem Zeichensatz *ISOLatin1* statt.

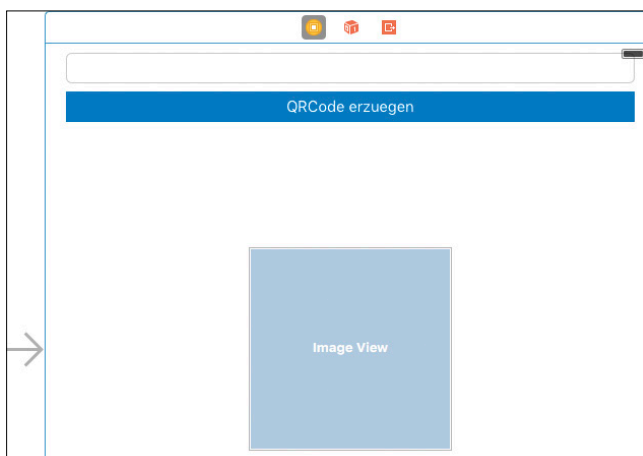
Anschließend wird eine neue Instanz der Klasse *CIFilter* angelegt. Als Parameter wird die Zeichenkette *CIQRCodeGenerator* übergeben. Damit wird der Filter angewiesen, aus einem zugewiesenen Parameter (*data*) den QR-Code zu erzeugen. Mittels der Methode *setValue* werden die Daten dann der *CIFilter*-Instanz zugewiesen. Das aus den Daten erzeugte Image wird dann der Variablen *qrcCIImage* zugewiesen.

QR-Code anzeigen!

Zuletzt wird im Listing die Methode *createQRCode* aufgerufen. Innerhalb dieser Methode werden dann die restlichen Schritte unternommen, um den QR-Code fertigzustellen und abschließend anzuzeigen (Listing 2).

Innerhalb der Methode *createQRCode* werden im ersten Schritt die Breite und Höhe der *ImageView* ermittelt und den Variablen *scaleX* und *scaleY* zugewiesen. Anschließend wird der in der Variablen *qrcCIImage* gespeicherte QR-Code mittels der Methode *imageByApplyingTransform* so umgewandelt, dass dieser im *ImageView*-Control angezeigt werden kann.

Zur Anpassung der Dimension wird die Funktion *CGAffineTransformMakeScale* aufgerufen und der Variablen als Parameter übergeben. Zuletzt erfolgt dann die Zuweisung des umgewandelten Bildes an die *Image*-Eigenschaft der *ImageView*-Instanz. Startet man die App jetzt, so wird nach Eingabe einer Zeichenkette in das TextField-Control und Betätigung des Buttons ein QR-Code erzeugt (Bild 2).



Die Oberfläche zur Erzeugung von QR-Codes (Bild 1)

Nicht nur beim Erzeugen von QR-Codes wird man von iOS unterstützt, auch für das Lesen von QR-Codes finden sich Funktionen in den Bibliotheken von iOS. Die Beispiel-App soll simpel aufgebaut sein. Sobald ein QR-Code in das Sichtfeld der App gerät, soll versucht werden den Inhalt eines vor der Kamera befindlichen QR-Codes zu entschlüsseln. Anschließend wird der entsprechende Text angezeigt. ►

Listing 1: Code zum Generieren des QR-Codes

```
@IBAction func buttonPressed(sender: AnyObject) {
    if qrcCIImage == nil {
        if uiTextField.text == "" {
            return
        }

        let data = uiTextField.text!.dataUsingEncoding(
            NSISOLatin1StringEncoding, allowLossyConversion:
            false)

        let filter = CIFilter(name: "CIQRCodeGenerator")

        filter!.setValue(data, forKey: "inputMessage")
        filter!.setValue("Q", forKey:
            "inputCorrectionLevel")

        qrcCIImage = filter!.outputImage
        uiTextField.resignFirstResponder()

        createQRCode()
    } else {
        uiImageView.image = nil
        qrcCIImage = nil
    }
}
```

Listing 2: Die Methode createQRCode

```
func createQRCode() {

    let scaleX = uiImageView.frame.size.width /
        qrcCIImage.extent.size.width

    let scaleY = uiImageView.frame.size.Height /
        qrcCIImage.extent.size.height

    let transformedImage =
        qrcCIImage.imageByApplyingTransform(
            CGAffineTransformMakeScale(
                scaleX, scaleY))

    uiImageView.image =
        UIImage(CIImage:transformedImage)
```

Zur Umsetzung des Projekts wird ein neues Projekt vom Typ *Single View Application* angelegt. Im Storyboard wird lediglich ein Label-Control am unteren Ende der View eingefügt und verankert. Danach kann es schon an die Codierung gehen.

Die Klasse *ViewController* wird von *UIViewController* sowie von *AVCaptureMetadataOutputObjectsDelegate* abgeleitet. Dieser Delegate ist erforderlich, um auf durch die Kamera angezeigte Metadaten zu reagieren. Neben einem Outlet für das Label-Control in der View werden folgende weiteren Variablen angelegt: *captureSession* (abgeleitet von *AVCaptureSession*), *captureVideoPreviewLayer* (abgeleitet von *AVCaptureVideoPreviewLayer*) sowie eine Instanz von *UIView*. QR-Codes können unterschiedliche Verschlüsselungen benutzen. Daher wird ein Array *supportedCodes* erzeugt, das die typischen verwendeten Codes enthält. Im folgenden Schritt muss die Kamera für das Lesen des QR-Codes initialisiert werden. Dies geschieht innerhalb der Methode *initDevice* (Listing 3).



Erzeugen: Der mit der App erzeugte QR-Code (Bild 2)

Um auf die Kamera zugreifen zu können, wird eine Referenz im Code benötigt. Diese Referenz wird mittels der Klasse *AVCaptureDevice* hergestellt. Die Methode *DefaultDeviceWithMediaType* liefert die voreingestellte Standardkamera zurück. Mit dem übergebenen Parameter *AVMediaTypeVideo* wird festgelegt, dass ein Video-Stream von der Kamera zurückgegeben werden soll.

Nachdem das Gerät, das für das Lesen des QR-Codes verwendet wird, spezifiziert wurde, geht es an die weitere Konfiguration. Zur Steuerung der Bildverarbeitung wird eine Instanz (*captureSession*) der Klasse *AVCaptureSession* benötigt. Diese Klasse enthält Eigenschaften und Methoden, die den Zugriff auf das gelesene Bild

ermöglichen. Das Gerät wird in der Variablen *deviceInput* gekapselt und mittels der Methode *addInput* der Instanz der Klasse *AVCaptureSession* übergeben.

Außerdem wird eine Variable *captureMetadataOutput* angelegt, in der das aufgezeichnete Ergebnis gespeichert wird.

Listing 3: Vorbereitung zum Lesen des QR-Codes

```
func initDevice() {
    let captureDevice = AVCaptureDevice.
    DefaultDeviceWithMediaType(
        AVMediaTypeVideo)

    do {
        let deviceInput = try
        AVCaptureDeviceInput(device:
        captureDevice)
        captureSession =
        AVCaptureSession()
        captureSession?.addInput(
        deviceInput)

        let captureMetadataOutput =
        AVCaptureMetadataOutput()
        captureSession?.addOutput(
        captureMetadataOutput)

        captureMetadataOutput.
        setMetadataObjectsDelegate(self,
        queue:
        dispatch_get_main_queue())
        captureMetadataOutput.
        metadataObjectTypes =
        supportedCodes

        captureVideoPreviewLayer =
        AVCaptureVideoPreviewLayer(
            session: captureSession)
        captureVideoPreviewLayer?.
        videoGravity =
        AVLayerVideoGravityResizeAspectFill
        captureVideoPreviewLayer?.frame =
        view.layer.bounds
        view.layer.addSublayer(
        captureVideoPreviewLayer!)
        captureSession?.
        startRunning()
        view.bringSubviewToFront(
        uiLabel)
        uiView = UIView()

        if let frameView = uiView {
            frameView.layer.borderColor =
            UIColor.greenColor().CGColor
            frameView.layer.borderWidth = 2
            view.addSubview(frameView)
            view.bringSubviewToFront(
            frameView)
        }
    } catch {
        print(error)
        return
    }
}
```


Auch diese Variable wird der Instanz *captureSession* durch Aufruf der Methode *addOutput* übergeben.

Als Nächstes wird eine Variable erzeugt, die zur Speicherung der aufgenommenen Daten verwendet wird. Die Variable *captureMetadataOutput* wird nach dem Anlegen der *captureSession*-Instanz durch einen Aufruf der Methode *addOutput* als Parameter übergeben.

Ohne Delegate geht nichts

Der Delegate für *captureMetadataOutput* wird als Nächstes initialisiert. Hierzu wird die Methode *setMetadataObjectsDelegate* aufgerufen und als Parameter eine Referenz auf die Instanz sowie die zu verwendende Warteschlange angegeben. Der Eigenschaft *metadataObjectTypes* wird das zu Beginn

definierte Array mit den Codes zugewiesen. Die Aufnahme des QR-Codes soll in der App wiedergegeben werden. Dies erledigt eine Instanz von *AVCaptureVideoPreviewLayer*.

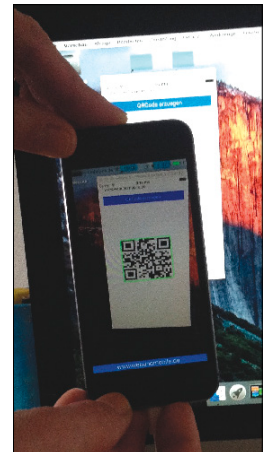
Hierzu wird die Instanz erst erzeugt, dann konfiguriert (Größe und Ausrichtung), um dann der View über die Methode *addSublayer* hinzugefügt zu werden. Jetzt muss das Scannen des QR-Codes nur noch gestartet werden. Hierfür genügt es, die Methode *startRunning* der *AVCaptureSession*-Instanz aufzurufen. Die weiteren Codezeilen in dieser Methode dienen lediglich der Anpassung der Ansicht der View. So wird zum Beispiel ein dünner grüner Rahmen angezeigt, wenn automatisch ein QR-Code erkannt worden ist.

Die Verarbeitung des QR-Codes findet innerhalb der Funktion *captureOutput* statt (Listing 4). Diese Methode wird automatisch wiederholt aufgerufen. Wird kein QR-Code erkannt, so wird ein entsprechender Text im *Label-Control* angezeigt. Erst wenn bei einer entsprechenden Quelle ein gültiger QR-Code gefunden wurde, wird im nächsten Schritt versucht, den Inhalt des QR-Codes zu entschlüsseln.

Zur Prüfung wird der gescannte Code mit den Codes aus dem Array verglichen. So wird ermittelt, ob der Inhalt des QR-Codes gelesen werden kann. War die Prüfung positiv (*if supportedCodes.contains*), so wird als Nächstes die Textinformation aus dem QR-Code gelesen, um anschließend diesen Text im *Label-Control* auszugeben. Das Ergebnis der letzten beiden Listings können Sie Bild 3 entnehmen.

Fazit

iOS bietet seit Version 7 Funktionen zum Erstellen und Lesen von QR-Codes an. Diese lassen sich mit wenig Aufwand in eine App integrieren. ■



Lesen des zuvor erzeugten QR-Codes (Bild 3)

Listing 4: Die Methode captureOutput

```
func captureOutput(captureOutput:
    AVCaptureOutput!,
    didOutputMetadataObjects
    outputMetadataObjects:
    [AnyObject]!, fromConnection
    connection:
    AVCaptureConnection!) {
    if outputMetadataObjects == nil ||
        outputMetadataObjects.count == 0 {
        UIView?.frame = CGRectZero
        UILabel.text =
            "Kein QRCode erkannt"
        return
    }
    let
        metadataMachineReadableCodeObject
        = outputMetadataObjects[0] as!
        AVMetadataMachineReadableCodeObject

    if supportedCodes.contains(
        metadataMachineReadableCodeObject
        .type)
    {
        let barCodeObject =
            captureVideoPreviewLayer?.
            transformedMetadataObjectForMetadataObject(
                metadataMachineReadableCodeObject)
        UIView?.frame =
            barCodeObject!.bounds

        if metadataMachineReadableCodeObject.
            stringValue != nil {
                UILabel.text =
                    metadataMachineReadableCodeObject.
                    stringValue
            }
    }
}
```

Links zum Thema

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



Christian Bleske

ist Autor, Trainer und Entwickler mit dem Schwerpunkt Client/Server und mobile Technologien. Sein Arbeitsschwerpunkt liegt auf Microsoft-Technologien.

cb.2000@hotmail.de

CROSS-PLATTFORM-ENTWICKLUNG MIT XAMARIN

.NET auf Abwegen

Mit Xamarin steht eine Laufzeitumgebung zur Verfügung, die .NET-Code unter Android und iOS ausführen kann.

Schon während der Konzeptphase achtete man bei Microsoft darauf, dass die damals als MSIL bezeichnete Intermediärsprache sorgfältig dokumentiert und für Drittanbieter implementierbar gestaltet wird. Kundenbindung sollte durch Bibliotheken und Betriebssystemservices erfolgen. Wenn ein Programm eine nur unter Windows zur Verfügung stehende Funktion nutzt, ist es nicht portierbar.

Für Windows Phone oder Windows RT geschriebener Code lässt sich mit Xamarin nicht eins zu eins weiterverwenden – von Qt umsteigende Entwickler müssen umdenken. Bei intelligenter Strukturierung der Applikation lässt sich zumindest die Engine recyceln. Wer die von Xamarin angebotenen Bibliotheken und Services in das eigene Programmdesign einbindet, reduziert den Portierungsaufwand weiter.

Für an Java oder Objective-C uninteressierte Entwickler ist Xamarin ebenfalls interessant. Die API-Mappings spiegeln die nativen Schnittstellen von Android und iOS nach C#. Sie lassen sich so auch ohne Kenntnis von Objective-C beziehungsweise Java nutzen.

Installation

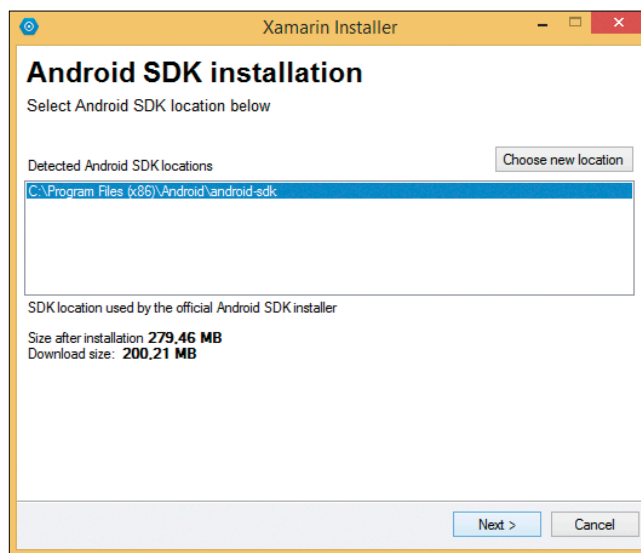
Xamarin setzt als Hostbetriebssystem Windows oder Mac OS X voraus. Eine Linux-Portierung der IDE wurde bisher nicht in Angriff genommen. Die in diesem Artikel gezeigten Screenshots entstanden in einer Windows-10-Partition, die von einer Ubuntu-Workstation via VirtualBox gehostet wurde. Da Xamarin Teile des .NET Frameworks von der Hostplattform übernimmt, sollten Sie im ersten Schritt Visual Studio 2013/2015 installieren. Dies muss unbedingt vor dem Deployment von Xamarin erfolgen.

Besuchen Sie danach den URL <https://xamarin.com/download> und füllen Sie das ebendort befindliche Formular aus, um den Installationsassistenten herunterzuladen. Die rund 2 MByte große Datei lädt weitere benötigte Komponenten dynamisch aus dem Internet herunter. Auch das Android SDK wandert so automatisch auf Ihre Maschine.

Im Installationsprozess findet sich eine kleine Besonderheit: Das in **Bild 1** gezeigte Finden von Installationsorten des Android SDK weist nicht darauf hin, dass sich dort schon eine Instanz der Google-Werkzeuge befindet.

Als Entwickler haben Sie im Bereich der Werkzeuge die Qual der Wahl. Neben der hier verwendeten Xamarin-eigenen IDE gibt es auch die Möglichkeit zur Integration in Visual Studio.

Nach dem erfolgreichen Abarbeiten des Installationsprozesses – Xamarin lädt dabei rund 2GByte Daten herunter – ist



Auch das Android-SDK wird automatisch installiert (Bild 1)

es an der Zeit, das Produkt via Startmenü (Eingabe *Xamarin Studio*) anzuwerfen. Ein im Rahmen der Initialisierung auftretender Fehler im Subversion-Modul ist für die folgenden Schritte irrelevant, weil wir unseren Code nicht in ein Versionskontrollsystem hochladen wollen.

Klicken Sie sodann auf *New Solution*, um den in **Bild 2** gezeigten Projekterstellungs-Assistenten auf den Bildschirm zu holen.

Die Entwicklung von auf mehreren Plattformen funktionsfähigen Engines ist kein besonderes Problem. Im Notfall hilft eine kleine, von Hand zusammengezippte HAL. Benutzerschnittstellen erweisen sich an dieser Stelle als kritischer, weil jeder Hersteller beim Design des GUI sein eigenes Süppchen kocht.

Xamarin bringt seit einiger Zeit einen als *Xamarin.Forms* bezeichneten GUI-Stack mit, der diese Problematik zu umgehen sucht. Wir wollen das vergleichsweise neue Produkt in diesem Artikel einspannen. Erstellen Sie eine Beispielanwendung anhand der Vorlage *Xamarin.Forms App*.

Die Festlegung des Namens und des Namespaces erfolgt erst im nächsten Schritt. Geben Sie *NMGTest1* ein, wenn Sie die maximale Ähnlichkeit zwischen Ihrem Beispiel und dem hier abgedruckten Code wünschen. Achten Sie zudem darauf, die Einstellung *Shared Code* auf *Use Shared Library* zu setzen. Die Nutzung von portablen Klassenbibliotheken ist zum Zeitpunkt der Drucklegung noch nicht ausgereift.

Die restlichen Einstellungen im Assistenten können unverändert übernommen werden. Nach der erfolgreichen Erstellung des Projektskeletts präsentiert Xamarin Studio die in **Bild 3** gezeigte Benutzerschnittstelle.

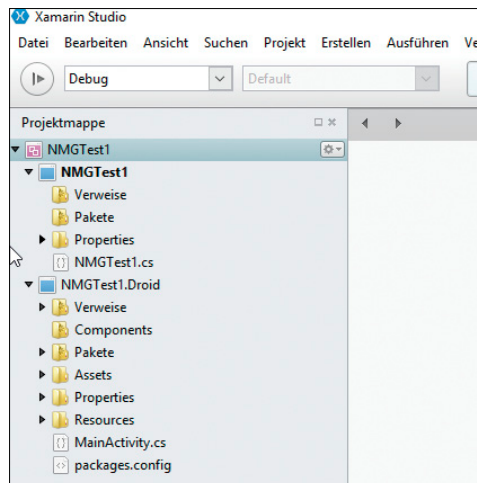
In der auf der linken Seite des Bildschirms eingeblendeten Skelettansicht ist klar ersichtlich, dass auf Xamarin.Forms basierende Projektmappen in zwei Elemente aufgeteilt sind. Beginnen wir unsere Betrachtungen mit der Datei *NMGTest1.cs*, die die aus Android und Co. hinreichend bekannte Activity implementiert.

Xamarin implementiert seit jeher einen vollwertigen Satz von Bindungen, die als Makler zwischen C# und den nativen Android- beziehungsweise iOS-APIs dienen. Unser Code beginnt mit dem Einbinden von rund einem Dutzend Namespaces. Der Großteil davon ist Android-spezifisch:

```
using System;
```

```
using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
```

Plattformspezifischer Code wird in Unter-Namespaces des Hauptprojekts untergebracht. Als Activity dienende Klassen



Xamarin Studio basiert auf dem aus der Unix-Welt bekannten MonoDevelop (**Bild 3**)

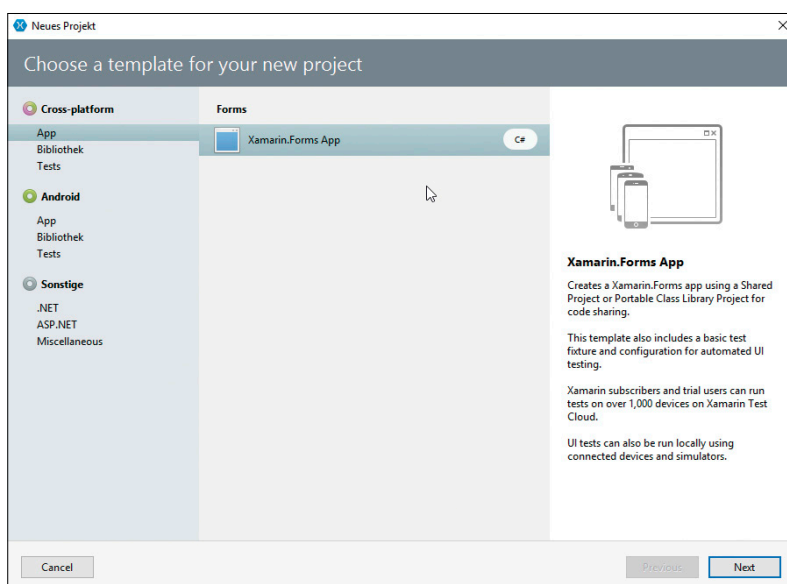
```
base.OnCreate (bundle);
global::Xamarin.Forms.Forms.Init (this, bundle);
LoadApplication (new App ());
}
}
```

Der eigentliche Start der Applikation erfolgt sodann durch *OnCreate* und *LoadApplication*. *Forms.Init* sorgt dabei dafür, dass der GUI-Stack hochfährt – das eigentliche Konfigurieren findet sich in der zwischen allen Plattformen geteilten Applikationsklasse.

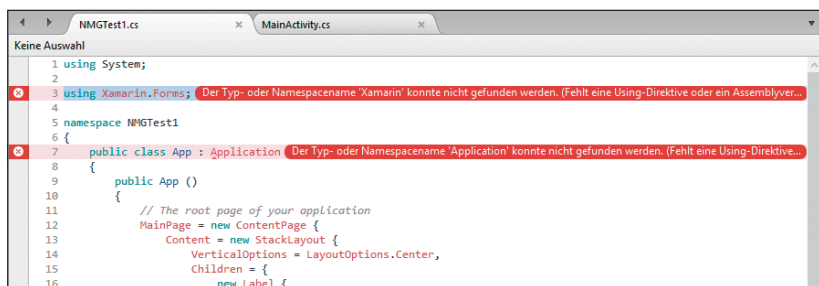
NMGTest1.cs enthält eine von *Application* abgeleitete Klasse. Im Rahmen ihres Konstruktors wird ein neues Formular aus diversen Steuerelementen zusammengebaut – da es in der *MainPage*-Eigenschaft liegt, erscheint es automatisch am Bildschirm.

```
namespace NMGTest1 {
    public class App : Application {
        public App () {
            MainPage = new ContentPage {
                Content = new StackLayout {
                    VerticalOptions =
                        LayoutOptions.Center,
                    Children = {
                        new Label {
                            XAlign = TextAlignment.Center,
                            Text = "Welcome to Xamarin
                                Forms!"
                        }
                    }
                }
            };
        }
    }
}
```

Zu guter Letzt gibt es noch die drei von den diversen nativen Betriebssystemen bekannten Event Handler, die für das Starten, Pausieren ►



Projektvorlagen: Nicht alle in Xamarin enthaltenen Projektvorlagen sind Cross-Plattform-geeignet (**Bild 2**)



Konfiguration: Bei einer unvollständigen Konfiguration erscheinen die gezeigten Warnungen (Bild 4)

und Beenden der Hauptapplikation zuständig sind:

```
protected override void OnStart () {
}
protected override void OnSleep () {
}
protected override void OnResume () {
}
```

Xamarin Forms funktioniert übrigens auch unter iOS.

Nach diesen einführenden Überlegungen wollen wir unser Programm zur Ausführung freigeben. Klicken Sie dazu auf den in der oberen linken Bildschirmkante eingeblendeten Play-Knopf, um den Debugprozess anzuwerfen. Die daneben angezeigte Combobox erlaubt die Auswahl des als Target zu verwendenden Geräts – Xamarin Studio unterstützt sowohl VMs als auch angeschlossene Telefone.

Falls Ihre Konfiguration nicht komplett ist, scheitert das Deployment mit den in Bild 4 gezeigten Warnungen.

Wer dieses Problem durch den von Visual Studio bekannten Verweise-Ordner zu lösen sucht, kommt nicht weiter. Die benötigten Assemblies sind noch nicht Teil der Solution. Klicken Sie stattdessen den Ordner *Pakete* mit der rechten Maustaste an und wählen Sie im daraufhin erscheinenden Pop-up-Menü die Option *Add Packages* aus.

Xamarin Studio reagiert darauf mit dem Einblenden des NuGet-Fensters, in dem das Paket *Xamarin.Forms* eingepflegt wird. Wenn die Kompilation ohne Probleme durchläuft, droht als nächste Unbill der in Bild 5 gezeigte Lizenzierungsdialog. Melden Sie sich mit einer beliebigen E-Mail-Adresse beim Anbieter an, um die 30 Tage dauernde Testphase zu beginnen. In Xamarin Studio müssen Sie dann nur noch die Daten eingeben, um das Testabo zu aktivieren.

Nach der erfolgreichen Anmeldung führt Xamarin Studio den Deploymentprozess weiter aus, liefert die Binärdateien aber nicht auf das Telefon. Dies lässt sich erst durch abermaliges Anklicken des Play-Buttons bewerkstelligen.

Steuerelemente herbei

Windows Phone und iOS sind aus Hardwaresicht einfach zu handhaben: Entwickler müssen sich nur um wenige Auflösungen kümmern. Android ist im Vergleich dazu ein Dschungel erster Güte. Am Markt finden sich sowohl Minitelefone mit QVGA-Display als auch Giganten mit Full HD.

Das Betriebssystem begegnet dieser Situation durch das von Qt bekannte Konzept der Layouts. Es handelt sich dabei um abstrakte Steuerelemente, die Informationen über die ideale Anordnung ihrer Kinder enthalten. Die eigentliche Positionierung der Widgets erfolgt so erst zur Laufzeit. Bei korrekter Parametrierung sieht das entstehende Programm sowohl im Landscape- als auch im Portrait-Mode perfekt aus.

Wir wollen dies durch die Erstellung eines Widerstandsrechners testen. Es handelt sich dabei um ein Programm, das den zur Kenn-

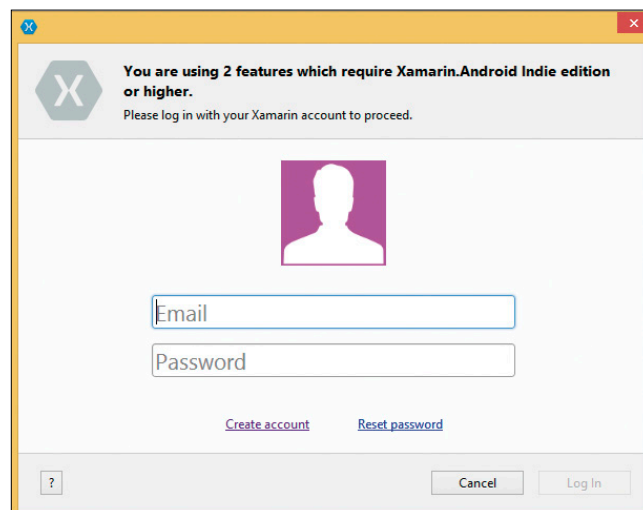
zeichnung elektrischer Bauteile verwendeten Farbcode decodiert. Das in Bild 6 kurz beschriebene Codierschema ist für diesen Artikel nicht wirklich wichtig. Wir müssen unsere Arbeit mit der Einrichtung eines Buttons und eines Labels beginnen, die übereinander auf der Unterseite des Bildschirms erscheinen sollen.

Dazu müssen wir die in *NMGTest1.cs* angelegte Klasse an einigen Stellen erweitern. Als Erstes benötigen wir zwei Variablen, die die zu erstellenden Objektinstanzen referenzieren. Bei kleinen Applikationen reicht es aus, diese nach folgendem Schema als globale Member anzulegen:

```
Public class App : Application {
    Label myLabel;
    Button myButton;
```

Im Konstruktor müssen wir die beiden Variablen bevölkern. Xamarin-Entwickler nutzen dabei gern die im Snippet gezeigte Inline-Objektdeklaration. Das von Microsoft vor einigen Jahren eingeführte Feature war an sich als Entgegenkommen an JavaScript-Entwickler gedacht, macht aber auch hier eine gute Figur:

```
Public App () {
    myButton = new Button {
```



Lizenz: Hier ist eine Lizenz erforderlich (Bild 5)

```
Text = "Berechnen",
BorderWidth = 1,
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.CenterAndExpand
};
```

Layouts setzen zusätzliche Positionierungsinformationen voraus, die über die *HorizontalOptions*- und *VerticalOptions*-Eigenschaften in die Widgets wandern. *LayoutOptions* enthält dabei eine Liste aller zulässigen Attribute, die sich zum Zeitpunkt der Drucklegung folgendermaßen präsentiert:

```
public struct LayoutOptions
{
    public static readonly LayoutOptions Start;
    public static readonly LayoutOptions Center;
    public static readonly LayoutOptions End;
    public static readonly LayoutOptions Fill;
    public static readonly LayoutOptions StartAndExpand;
    public static readonly LayoutOptions
        CenterAndExpand;
    public static readonly LayoutOptions EndAndExpand;
    public static readonly LayoutOptions FillAndExpand;
```

Wir nutzen hier die Eigenschaften *Center* und *CenterAndExpand*. Erstere platziert das Steuerelement in der Mitte des Elterncontainers, während die Zweitgenannte zusätzlich eine Expansion in die jeweilige Achse vorschreibt.

Deklaration des Labels

Widgets wachsen nur so lange, wie sie den zusätzlichen Bildschirmplatz sinnvoll ausnutzen können. Im Fall primitiver Steuerelemente wie Buttons legen die GUI-Stacks oft zusätzliche Constraints fest. Auf manchen Betriebssystemen ist etwa die Höhe von Knöpfen fix vorgegeben.

Die Deklaration des Labels unterscheidet sich nur insofern vom Knopf, als statt *BorderWidth* nun die Eigenschaft *FontSize* festgelegt wird:

```
myLabel = new Label
{
    Text = "Widerstand hier",
    FontSize = 14,
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.CenterAndExpand
};
```

In *MainPage* schreiben wir die beiden Steuerelemente anhand der Variablen in das *Children*-Array des *StackLayout* ein:

```
MainPage = new ContentPage
{
    Content = new StackLayout
    {
        VerticalOptions = LayoutOptions.Center,
        Children = {
```

```
myButton, myLabel
    }
};
```

Die beiden Steuerelemente erscheinen nun in der Mitte des Bildschirms.

Unser Programm erfüllt im Moment nur einen Teil der Spezifikation. Die beiden Widgets sind zwar übereinander, befinden sich aber noch nicht am richtigen Platz. Der einfachste Weg zur Lösung dieses Problems ist das Einfügen der fehlenden Steuerelemente.

Widerstände sind im Handel sowohl mit vier als auch mit fünf Farbringen erhältlich. Wir wollen uns hier aus Gründen der Bequemlichkeit auf vierringige Bauteile beschränken. Die zur Auswahl der Farben notwendigen vier nebeneinanderliegenden Listen würden die Deklaration des Formulars stark erweitern und nur schwer wartbar werden lassen.

Zur Lösung dieses Problems bietet sich die Auslagerung der Logik in eine separate Klasse an. Klicken Sie das Projekt *NMGTest1* mit der rechten Maustaste an und wählen Sie im daraufhin erscheinenden Kontextmenü die Option *Hinzufügen* und *Neue Datei*. Nutzen Sie die Vorlage *General* und *Leere Klasse* und vergeben Sie als Name *ListManagerClass*.

Nach dem Anklicken des *New*-Buttons erzeugt Xamarin Studio eine leere neue Klasse. Als *Mater*-Objekt soll abermals ein *StackLayout* dienen, weshalb der Korpus im ersten Schritt folgendermaßen angepasst wird:

```
public class ListManagerClass : StackLayout {
    public ListManagerClass () : base() {
        this.Orientation = StackOrientation.Horizontal;
        this.Spacing = 2;
        var aLabel = new Label ();
        aLabel.Text = "A";
        this.Children.Add (aLabel);
        aLabel = new Label ();
        aLabel.Text = "B";
        this.Children.Add (aLabel);
    }
}
```

Durch die Ableitung von *StackLayout* verhält sich unsere Klasse aus Sicht eventueller Aufrufer beziehungsweise Nutzer wie ein normales Steuerelement beziehungsweise Layout. Im Konstruktor erstellen wir zu Demonstrationszwecken zwei Labels, die in die von der *Mater*-Klasse vorgegebene *Children*-Auflistung wandern.

Die Einbindung in die Applikationsklasse erfolgt nach dem gewohnten Schema. Wir erstellen im ersten Schritt eine neue Instanz der *ListManagerClass*, die daraufhin in die *Children*-Eigenschaft des *StackLayout* wandert:

```
public class App : Application
{ ...
    ListManagerClass myListManager;
    public App () {
        ...
```

```

myListManager = new ListManagerClass ();
MainPage = new ContentPage {
    Content = new StackLayout {
        VerticalOptions = LayoutOptions.Center,
        Children = {
            myListManager, myButton, myLabel
        }
    }
};
}

```

Layouts lassen sich nach Belieben verschachteln: Ein Kinder-Layout ist aus Sicht seines Eltern-Objekts ein normales Widget, das nach der Zuweisung von Platz seine Unterelemente anordnet.

Einfügen der Listenelemente

Die im Moment nicht korrekt funktionierende Größenberechnung ist insofern kein Problem, als die beiden Labels nur zur Vorführung der prinzipiellen Vorgehensweise im Bereich Layout vorgesehen waren. Wirklich interessant wird der Widerstandsrechner erst durch das Einfügen der Listenelemente mit den eigentlichen Farbwerten.

Mit klassischen Versionen von Visual Basic aufgewachsene Entwickler müssen an dieser Stelle umdenken. Xamarin-Listen sind Arrays von gleichartigen Steuerelementen, die das eigentliche Rendern der Inhalte übernehmen. Es gibt keine primitive Liste, die ihre Informationen aus einem String-Array bezieht.

Wer eine ListView zu erstellen sucht, beginnt im Idealfall mit der Generierung der Datenklasse. Für unser Beispiel reicht die hier abgedruckte *ColorClass* aus, die neben der Farbe des Rings auch einen String mit dem Farbnamen entgegennimmt:

```

public class ColorClass {
    public ColorClass(string _aName, Color _aCol) {
        Name=_aName;
        RingColor=_aCol;
    }
    public string Name { private set; get; }
    public Color RingColor { private set; get; }
}

```

An dieser Stelle wartet ein kleines Gotcha auf den Entwickler: Die Klasse *Color* ist in mehreren Namespaces enthalten. Zur Nutzung in Xamarin's GUI-Stack ist die Inklusion des Namespaces *Xamarin.Forms* notwendig. Es kann aus Klarheitsgründen sinnvoller sein, die Klasse nach folgendem Schema jedes Mal vollständig zu qualifizieren:

```

public class ColorClass {
    public ColorClass(string _aName,
        Xamarin.Forms.Color _aCol)
    ...
    RingColor=_aCol; }
    ...
}

```

```

public Xamarin.Forms.Color RingColor {
    private set; get; }
}

```

Das Exponieren der Eigenschaften per Property ist kein Manierismus des Autors. Xamarin bindet die Listenelemente über das von WPF bekannte Data Binding an die im Hintergrund liegenden Modellklassen. Als öffentliche Variable beziehungsweise Member deklarierte Elemente stellen kein gültiges Bindungsziel dar.

Im nächsten Schritt müssen wir eine Liste deklarieren, die die anzuzeigenden Werte enthält. Im Fall des ersten Rings sind hierzu die neun im Listing gezeigten Felder notwendig:

```

public ListManagerClass () : base()
{
    ...
    List<ColorClass> ringList = new List<ColorClass> ();
    ringList.Add(new ColorClass("Braun",
        Color.FromRgb(128,0,0)));
    ringList.Add(new ColorClass("Rot",Color.Red));
    ringList.Add(new ColorClass("Orange",
        Color.FromRgb(255,69,0)));
    ringList.Add(new ColorClass("Gelb",Color.Yellow));
    ringList.Add(new ColorClass("Grün",Color.Green));
    ringList.Add(new ColorClass("Blau",Color.Blue));
    ringList.Add(new ColorClass("Violett",Color.Pink));
    ringList.Add(new ColorClass("Grau",Color.Gray));
    ringList.Add(new ColorClass("Weiss",Color.White));
}

```

Im Hinblick auf .NET erfahrene Entwickler wundern sich darüber, dass einige der Farben aus RGB-Werten zusammengesetzt werden, während andere direkt aus *Color* stammen. Xamarin's Farbklass ist vom Umfang her nicht mit der von .NET vergleichbar. Zum Zeitpunkt der Drucklegung fanden sich dort nur die im durch Reflection entstandenen Listing gezeigten Farbcodes:

```

public struct Color
{
    public static readonly Color Teal;
    public static readonly Color Yellow;
    public static readonly Color White;
    public static readonly Color Silver;
    public static readonly Color Red;
    public static readonly Color Pink;
    public static readonly Color Purple;
    public static readonly Color Olive;
    public static readonly Color Navy;
    public static readonly Color Maroon;
    public static readonly Color Green;
    public static readonly Color Gray;
}

```

```

[Obsolete ("Fuschia is obsolete as of version 1.3,
please use the correct spelling of Fuchsia")]
public static readonly Color Fuschia;
public static readonly Color Fuchsia;

```


Farbcodierung von Widerständen mit 4 Ringen					
Farbe		Widerstandswert in Ω			Toleranz
		1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	
„keine“	×	—	—	—	$\pm 20\%$
silber		—	—	$10^{-2} = 0,01$	$\pm 10\%$
gold		—	—	$10^{-1} = 0,1$	$\pm 5\%$
schwarz		—	0	$10^0 = 1$	—
braun		1	1	$10^1 = 10$	$\pm 1\%$
rot		2	2	$10^2 = 100$	$\pm 2\%$
orange		3	3	$10^3 = 1.000$	—
gelb		4	4	$10^4 = 10.000$	—
grün		5	5	$10^5 = 100.000$	$\pm 0,5\%$
blau		6	6	$10^6 = 1.000.000$	$\pm 0,25\%$
violett		7	7	$10^7 = 10.000.000$	$\pm 0,1\%$
grau		8	8	$10^8 = 100.000.000$	$\pm 0,05\%$
weiß		9	9	$10^9 = 1.000.000.000$	—

Widerstandswerte: Der Wert eines mit vier Ringen versehenen Rings lässt sich durch Multiplikation ermitteln (Bild 6)

```
public static readonly Color Blue;
public static readonly Color Black;
public static readonly Color Aqua;
public static readonly Color Transparent;
public static readonly Color Lime;
```

Die Erstellung der *ListView* ist vergleichsweise einfach. Die Grundvariante kommt mit der Deklaration der als Modellklasse dienenden *ItemsSource* und einer Generatormethode für die eigentlichen Zeilenelemente aus:

```
public ListManagerClass () : base() {
    ...
    var ringListView=new ListView();
    ringListView.ItemsSource=ringList;
    ringListView.ItemTemplate = new DataTrigger
        (typeof(null));
    Children.Add (ringListView);
}
```

Leider ist der Code so noch nicht lauffähig: Wer eine *ListView* mit einem leeren Item-Konstruktor ausstattet, provoziert einen Compilerfehler oder eine *NullReferenceException*. Die eigentliche Generierung der Listenelemente ist etwas komplexer. Der dazu notwendige Code sieht so aus:

```
public class ListChildClass:ViewCell {
    public ListChildClass () {
        Label nameLabel = new Label();
        nameLabel.SetBinding(Label.TextProperty, "Name");
        BoxView boxView = new BoxView();
        boxView.SetBinding(BoxView.ColorProperty,
```

```
"RingColor");
        StackLayout aLayout=new StackLayout();
        aLayout.Orientation=StackOrientation.Horizontal;
        aLayout.Children.Add(boxView);
        aLayout.Children.Add(nameLabel);
        this.View=aLayout; }
}
```

ListChildClass ist von der im Xamarin GUI-Stack enthaltenen Mater-Klasse *ViewCell* abgeleitet. Im Rahmen des Konstruktors erzeugen wir im ersten Schritt zwei Steuerelemente, deren Eigenschaften durch die *SetBinding*-Funktion mit den exponierten Attributen der Datenmodellklasse Verbindung aufnehmen. Darauf folgt die Errichtung eines *StackLayouts*, in dem die beiden Items unterkommen. Das Einpflegen in die *View*-Instanz sorgt dafür, dass der Renderer die Steuerelemente auch wirklich findet.

Damit können wir den Konstruktor folgendermaßen anpassen – führen Sie das Programm danach aus, um sich am in Bild 7 gezeigten Formular zu erfreuen:

```
var ringListView=new ListView();
ringListView.ItemsSource=ringList;
ringListView.ItemTemplate = new DataTemplate
    (typeof(ListChildClass));
Children.Add (ringListView);
```

Achten Sie darauf, dass die zur Darstellung der Datenzeilen verwendeten Elemente nicht allzu komplex werden dürfen. Bei Listen mit Dutzenden von Einträgen leidet mit der Zeit die Performance aufgrund des Vorhandenseins von zu vielen Steuerelementen.

Der einfachste Weg zur Steigerung der Performance ist das Recyceln der für Zeilen zuständigen DOM-Bäume. Dieses sowohl unter iOS als auch Android verfügbare Feature arbeitet mit einem Handapparat von DOM-Trees, die beim Scrollen dynamisch mit neuen Informationen befüllt werden.

Ab einem gewissen Komplexitätsgrad empfiehlt Xamarin das Implementieren eines selbst entwickelten Renderers, der den Bildschirm dynamisch anhand der im Modell befindlichen Informationen in eine Bitmap zeichnet. Diese Methode führt zu immens hoher Performance (Stichwort Scrolling), erfordert aber viele Mannstunden Implementierungsaufwand.

Da unser Programm nicht sonderlich komplex ist, wollen wir dem Thema an dieser Stelle keinen weiteren Raum geben. Falls Sie mit Ihrer Liste unter Performanceproblemen leiden, finden sich weitere Informationen in der unter <https://developer.xamarin.com/guides/cross-platform/xamarin-forms/user-interface/listview/performance> einsehbaren Dokumentation.

Deklaration weiterer Listenelemente

Widerstandsfarbcodes weisen die angenehme Eigenheit auf, teilweise aufeinander aufgebaut zu sein. Dies lässt sich zur Verkürzung der Deklaration der weiteren Listenelemente einspannen. Die *ItemsSource* der umfangreicheren zweiten *ListView* entsteht durch das Klonen und Erweitern der ers- ►

ten Instanz. Beachten Sie in Bezug auf die Farben die weiter oben erfolgte Erklärung: *Black* liegt in *Color* als vorgefertigter Member bereit:

```
List<ColorClass> ring2List = new List<ColorClass> ();
ring2List.Add(new ColorClass("Schwarz",Color.Black));
ring2List.AddRange (ring1List);
var ring2View=new ListView();
ring2View.ItemsSource=ring2List;
ring2View.ItemTemplate = new DataTemplate
(typeof(ListChildClass));
Children.Add (ring2View);
```

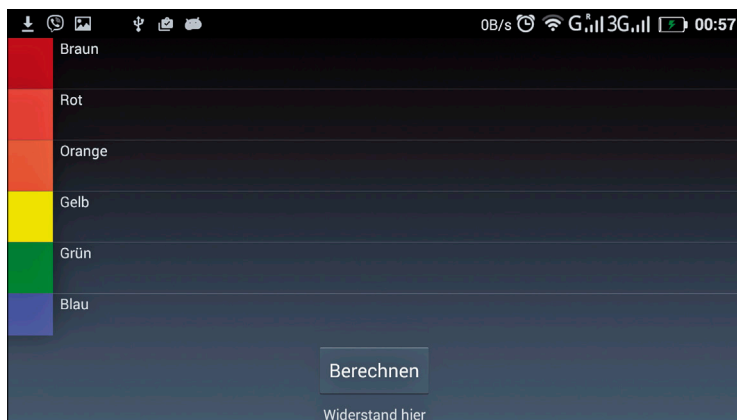
Angemerkt sei, dass diese Vorgehensweise nur dann funktioniert, wenn die anzuzeigenden Elemente komplett statisch sind. Wenn die in den Zeilen verwendeten Widgets die Beeinflussung der im Modell enthaltenen Informationen erlauben, kommt es zu Überschneidungen. Die *AddRange*-Funktion erzeugt keine neuen Instanzen der *ColorClass*-Objekte,

umgeht diese Falle dadurch, dass er mit der Entfernung des höchstwertigen Items beginnt:

```
List<ColorClass> ring4List = new List<ColorClass> ();
ring4List.AddRange (ring3List);
ring4List.RemoveRange (3, 2);
ring4List.RemoveRange (0,1);
```

Xamarins GUI-Stack betrachtet Listen als Steuerelement höherer Priorität. Das bedeutet, dass ihnen bei Änderungen des Layouts automatisch zusätzlicher Platz zugute kommt. **Bild 8** und **Bild 9** zeigen das Programm auf einem Android-Smartphone mit einem Full-HD-Display.

Das Layout ließe sich durch die Einführung weiterer Metadaten – insbesondere im Portrait-Mode – stark verbessern. Wir besprechen die Layout-Engine an dieser Stelle aus didaktischen Gründen nicht weiter. Wer Xamarin ernsthaft verwenden möchte, muss sich auf jeden Fall mit dem GUI-Stack auseinandersetzen. Das Anpassen dieses Beispiels ist hierfür eine geradezu ideale Übungsaufgabe.



Liste: Die erste Liste funktioniert problemlos (**Bild 7**)

sondern fügt nur Referenzen auf die schon vorhandenen Elemente ein. Der Aufbau der dritten Liste setzt das Entfernen einiger Items voraus. Aus Sicht des Garbage Collectors ergibt sich hier kein undefinierter Zustand. Die Klassen werden auch nach dem Aufruf von *Remove* nicht abgetragen, da sie in einer der Listen nach wie vor verwendet werden:

```
List<ColorClass> ring3List = new List<ColorClass> ();

ring3List.AddRange (ring2List);
ring3List.RemoveRange (8, 2);
ring3List.Add (new ColorClass("Silber",
Color.FromRgb (90, 90, 90)));
ring3List.Add (new ColorClass("Gold",
Color.FromRgb (255,215,0)));
```

Beim Design der vierten Liste können Probleme bei der Berechnung der Indizes auftreten. Wer das nullte Item entfernt, verschiebt alle anderen um eine Stelle nach oben. Das Finden des nächsten Ziels wird dadurch erschwert. Unser Code

Event-Verarbeitung leicht gemacht

C# bietet mit Delegates eine bequeme und standardisierte Methode zur Weiterleitung von Ereignissen an. Die Fertigstellung unseres Programms setzt das Auswerten des Click-Events für den Knopf voraus – in *NMGTest1.cs* ist dazu die folgende Änderung notwendig:

```
public App () {
    myButton = new Button
    {
        ...
    };
    myButton.Clicked += OnButtonClicked ;
    ...
    void OnButtonClicked(object sender, EventArgs e)
    {
    }
}
```

Zur Ermittlung der ID des gerade ausgewählten Feldes muss die *App*-Klasse auf die in *ListManagerClass* erstellten *List-View*-Instanzen zugreifen können. Der einfachste Weg zur Herstellung dieses Zustands ist das Exponieren durch vier globale Membervariablen.

Nach dieser Änderung können wir uns der eigentlichen Berechnung zuwenden. Dazu müssen wir die vier selektierten Ringe ermitteln. Für die Bestimmung der zweistelligen Basiszahl ist etwas Integralarithmetik ausreichend. Das Absichern gegen Nicht-Selektiertsein von Listenelementen sorgt für zusätzliche Zuverlässigkeit:

```
void OnButtonClicked(object sender, EventArgs e)
{
    int sel1 = myListManager.ring1List.IndexOf
        ((ColorClass)myListManager.ring1View.SelectedItem);
    int sel2 = myListManager.ring2List.IndexOf
```



Im Landscape-Mode ist alles in bester Ordnung (Bild 8)

```
((ColorClass)myListManager.ring2View.SelectedItem);
int sel3 = myListManager.ring3List.IndexOf
((ColorClass)myListManager.ring3View.SelectedItem);
int sel4 = myListManager.ring4List.IndexOf
((ColorClass)myListManager.ring4View.SelectedItem);

if(sel1==1 || sel2==1 || sel3==1)
{
    myLabel.Text = "Fehlende Daten";
    return;
}
double myBase=(sel1+1)*10+sel2;
```

Xamarin erschwert die Ermittlung des Index des gerade ausgewählten Elements dadurch, dass die *SelectedItem*-Eigenschaft einen Zeiger auf das Item zurückliefert. Dieser muss durch die *ItemsSource* gejagt werden, um den numerischen Index zu erhalten.

Beim Multiplikationsfaktor ist etwas Zusatzarbeit erforderlich, weil wir herauf- und herabskalierende Farbwerte unterscheiden müssen. Danach muss der errechnete Wert nur noch in das Label wandern:

```
if (sel3 == 8)
    myBase *= 0.1;
else if (sel3 == 9)
    myBase *= 0.01;
else {
    for (int i = 0; i < sel3; i++)
        myBase *= 10;
}
myLabel.Text = myBase.ToString();
```

Für die Ermittlung der Toleranz böte sich die Nutzung einer Switch-Anweisung oder eines Lookup-Arrays an. Die Nutzung beider Strukturen wird an dieser Stelle als bekannt vorausgesetzt. Damit ist das Programm so gut wie fertig.

Noch mehr Events

Beim Design von Handcomputerapplikationen sollten Entwickler die Anzahl der zur Erreichung einer notwendigen

Aufgabe notwendigen Klicks minimieren. Im Fall unseres Widerstandsrechners ist der Berechnen-Button ein Anachronismus – ob des minimalen Rechenaufwands könnten wir die Anzeige direkt aktualisieren, sobald der Nutzer eine der vier Listen berührt. Richtig smarte Entwickler kommen dabei sogar mit nur einem Event Handler aus.

Da der Autor dieser Zeilen vor vielen Jahren einen Lektor an einer renommierten österreichischen Fachhochschule für Mobilcomputertechnik damit in die Verzweiflung trieb, wollen wir den Weg zur Optimallösung gemeinsam beschreiten.

C#-Event-Handler sind im Grunde genommen eine fortgeschrittenere Form des Funktionspointers. Es gibt keinerlei Regel, die besagt, dass eine Handler-Methode nicht mit mehreren Ereignissen verdrahtet sein darf. Das bedeutet, dass wir im Konstruktor von *NMG-Test1* alle vier Listboxen mit ein- und derselben Methode verbinden:

```
public App () {
    ...
    myListManager = new ListManagerClass ();
    myListManager.ring1View.ItemSelected +=
        OnItemSelected;
    myListManager.ring2View.ItemSelected +=
        OnItemSelected;
    myListManager.ring3View.ItemSelected +=
        OnItemSelected;
    myListManager.ring4View.ItemSelected +=
        OnItemSelected;
```

Die ist keine höhere Kunst, da wir nur den weiter oben erstellten *OnButtonClicked*-Handler aufrufen. Das Übergeben von null-Parametern ist unkritisch, weil der Code der Methode die beiden Werte in keiner Weise auswertet.

Windows Phone-Unterstützung

Xamarin unterstützt nicht nur Android, das Produkt ist auch unter iOS und Windows Phone lauffähig. Besitzer eines Apple-Computers können das iPhone per Xamarin Studio ansteuern. Zudem lässt sich ein Mac als Buildserver für andere Maschinen im Netzwerk freigeben.

Die Entwicklung von Applikationen für Windows Phone ist insofern komplizierter, als Xamarin das Microsoft-Betriebssystem nur in Visual Studio unterstützt. Diese seit der ersten Version von Xamarin bestehende Beschränkung wird laut dem Unternehmen auch in Zukunft nicht geändert. Weitere Informationen hierzu finden sich unter <http://forums.xamarin.com/discussion/15125/can-i-use-xamarin-studio-for-windows-phone-8-on-my-mac>.

Zum Zeitpunkt der Drucklegung ist Xamarin für Windows in drei Varianten verfügbar: Windows Phone wird durch Silverlight 8 unterstützt, während es für die klassische Metro-Laufzeitumgebung ebenfalls eine dedizierte Version gibt. Universal Windows Apps entstehen im Moment mit einer Technical Preview, die sich von den beiden Vorgängern ►

durch das Vorhandensein eines Mapping-Steuererelements unterscheidet.

Der Weg zur Portierung unserer App auf ein Windows Phone 8 ist einfach: Öffnen Sie Visual Studio 2015 und öffnen Sie die von Xamarin Studio erstellte Solution-Datei wie gewohnt. Klicken Sie die Solution sodann rechts an, um sie um ein neues Projekt vom Typ *Windows Phone* und *Blank App* zu erweitern. In den folgenden Schritten wurde als Name *NMGTest1.WinPho* angenommen.

Klicken Sie das neu erstellte Projekt sodann rechts an, um über die Option *Manage NuGet Packages...* die Erweiterung *Xamarin.Forms* hinzuzufügen. Im nächsten Schritt ist eine Referenz auf das geteilte Codeprojekt erforderlich, die sich unter *Add, Reference, Shared Projects* und *Projects* findet.

Auf Seiten des Codes sind einige Anpassungen notwendig, um das Laden der Forms-Assembly zu ermöglichen. Als Erstes wird die *Init*-Methode von *App.xaml.cs* um die folgende Passage erweitert:

```
protected override void OnLaunched
(LaunchActivatedEventArgs e)
{
    ...
    Xamarin.Forms.Forms.Init(e);
    if (e.PreviousExecutionState ==
        ApplicationExecutionState.
        Terminated){
        // TODO: Load state from
        // previously suspended application
    }
    ...
}
```

MainPage.xaml implementiert im Moment ein auf dem normalen Windows Phone-GUI-Stack basierendes Formular. Ersetzen Sie den Inhalt der Datei durch folgendes Formular, das als eine Art Container für die anzuzeigenden Informationen dient:

```
<forms:WindowsPhonePage
    x:Class="NMGTest1.WinPho.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:NMGTest1.WinPho"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:forms="using:Xamarin.Forms.Platform.WinRT"
    mc:Ignorable="d">
```



Im Portrait-Mode geht es stellenweise etwas gedrängt zu (Bild 9)

```
Background="{ThemeResource
    ApplicationPageBackgroundTheme
    Brush}">
```

```
</forms:WindowsPhonePage>
```

Zu guter Letzt sind im Code-behind zwei Modifikationen notwendig. *MainPage* darf nicht mehr von *PhonePage* abgeleitet sein, und im Konstruktor rufen wir *LoadApplication* auf:

```
public sealed partial class MainPage
{
    public MainPage() {
        this.InitializeComponent();
        LoadApplication(new
            NMGTest1.App());
        this.NavigationCacheMode =
            NavigationCacheMode.Required;
    }
}
```

Damit ist auch diese Variante des Programms einsatzbereit.

Generationen von Informatikern haben ihre Benutzerschnittstellen in Code zusammengebaut. In Zeiten von ständig steigender Rechenleistung ist dieses vergleichsweise arbeitsaufwendige Verfahren nicht unbedingt sinnvoll. Die Nutzung von auf XML basierenden Beschreibungssprachen erleichtert die Erstellung der Formulare.

Sehen wir uns die Nutzung dieses Features an einem separaten Projekt im Detail an. Starten Sie Xamarin Studio neu und erstellen Sie eine Applikation namens *NMGTest2*. Die Vorlage kann vom vorigen Mal unverändert übernommen werden, achten Sie auf die Einstellung des korrekten Code-Teilungssystems. Klicken Sie *NMGTest2* im nächsten Schritt mit der rechten Maustaste an und wählen Sie die Option *Hinzufügen* und *Neue Datei*. Wechseln Sie im Erstellungsassistenten in die Rubrik *Forms* und erstellen Sie ein neues Element auf Basis des Typs *Forms ContentPage XAML*. Xamarin Studio reagiert darauf mit der Erstellung von zwei Dateien: Neben der XAML-Datei gibt es auch eine XAML.CS-Datei.

In der Applikationsklasse muss der Inhalt von *App()* folgendermaßen angepasst werden, um das Laden der – noch leeren – XAML-Page anzuweisen:

```
public class App : Application {
    public App () {
        MainPage = new MyTestPage();
    }
}
```

Öffnen Sie die Datei *MyTestPage.xaml* per Doppelklick und ersetzen Sie den dort befindlichen Inhalt durch eine um einige Steuerelemente erweiterte Variante:

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/
```

```
forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/
xaml" x:Class="NMGTest2.MyTestPage">
```

```
<ContentPage.Content>
<StackLayout>
    <Slider x:Name="MySlider" VerticalOptions=
        "CenterAndExpand" />

    <Button x:Name="CmdAdd" Text="Addieren"
        HorizontalOptions="Center"
        VerticalOptions="CenterAndExpand" />
    <Button x:Name="CmdSub" Text="Subtrahieren"
        HorizontalOptions="Center"
        VerticalOptions="CenterAndExpand" />

</StackLayout>
</ContentPage.Content>
</ContentPage>
```

Für Xamarin vorgesehene XAML-Dateien unterscheiden sich von ihren normalen Varianten dadurch, dass sie ein zusätzliches Schema laden. Davon abgesehen verhält sich die Beschreibungssprache im Großen und Ganzen wie von Windows gewohnt.

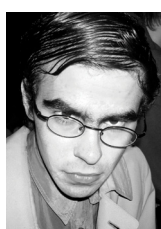
Unter Windows Phone ist XAML dafür bekannt, dass sich Event Handler und sonstige Logik mit geringem Aufwand realisieren lassen. Für das Entgegennehmen von in den beiden Knöpfen entstehenden Klickereignissen ist folgende Anpassung des Markups notwendig:

```
<Button x:Name="CmdAdd" ...
    Clicked="OnButtonAddClicked" />
<Button x:Name="CmdSub" ...
    Clicked="OnButtonSubClicked" />
```

Auf Seiten des Code-behinds müssen Sie die Methoden mit korrekter Signatur anlegen, um der XAML-Runtime das Verdrahten zu ermöglichen.

Fazit

Xamarin gibt in der Dokumentation ohne Beschönigung zu, dass Forms seine Stärke immer dann ausspielen kann, wenn das Programm nicht zu 100 Prozent nativ aussehen muss. Besitzer von großen C#-Codebasen sollten dem Produkt trotzdem Aufmerksamkeit schenken. Das Nicht-Portieren-Müssen der Geschäftslogik ist eine immense Erleichterung. ■



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

devbooks in 2016

.NET WPF
Moderne Desktop-
Anwendungen mit .NET

Neu!

02/
2016

ASP.NET
Microsofts neuer Weg
der Webentwicklung

Neu!

03/
2016

Data Science
Intelligente Verarbeitung
großer Datenmengen

Neu!

04/
2016

.NET-Architektur
So wird aus Anforderungen
eine perfekte Anwendung

Neu!

05/
2016

**Flexible Projektstrukturen
für .NET**

Neu!

06/
2016

Angular 2
Modulares Frontend
für den Browser

Neu!

07/
2016

**Digitale Revolution
in der IT**

Neu!

09/
2016

IN-MEMORY-DATENBANK REDIS

Multitalent für viele Einsatzfälle

Redis ist mehr als nur ein RAM-basierter Key-Value-Store.

Tatsächlich ist Redis ein hervorragend skalierendes Multitalent für viele Einsatzfälle. Bei Twitter, StackExchange, Pinterest und anderen namhaften Firmen bildet Redis mittlerweile einen unverzichtbaren Teil des Technologiestacks. Aber auch bei deutlich kleinerem Datenaufkommen spielt Redis seine Stärken aus.

In einem früheren Artikel haben wir uns neben Memcached und Azure AppFabric auch Redis schon einmal näher angesehen. Neben der Eignung zum externen Caching von Anwendungsdaten hat Redis aber weitaus mehr Funktionalität zu bieten. Lassen Sie uns daher im Folgenden einige typische Anwendungsfälle zusammentragen, die sich mit Redis umsetzen lassen. Wir beschränken uns dabei absichtlich auf die Sicht des Software-Architekten beziehungsweise Anwendungsentwicklers und gehen auf administrative Themen nur insoweit ein, als dies für unsere Zwecke nötig ist.

Key-Value-Store

Der Name Redis steht für »remote dictionary server« und beschreibt eine Hauptcharakteristik recht gut: Redis kann primär der Gruppe der Key-Value-Stores zugerechnet werden. Sowohl Schlüssel als auch Werte können theoretisch bis zu 512 MByte groß werden. Was für Werte völlig in Ordnung ist, sollte man in der Praxis für Schlüssel jedoch aus Performancegründen besser nicht ausreizen. Natürlich kann und sollte man lesbare Schlüssel statt unhandlicher Abkürzungen verwenden, dennoch empfiehlt die Dokumentation mit gutem Grund im Zweifelsfall die Wahl des kürzeren Schlüssels.

Da ein typischer Key-Value-Store eben aus Schlüssel-Wert-Paaren besteht, eignet er sich für die Ablage strukturierter Daten zunächst nur bedingt. Eine in der Praxis deshalb häufig anzutreffende Verfahrensweise ist es daher, Informationen über die Datenstruktur in den Schlüsselnamen zu verpacken. So könnte man etwa Daten eines Kunden mit folgenden Schlüsseln ablegen:

```
Kunde:4711:Name  
Kunde:4711:Vorname  
Kunde:4711:Adresse  
Kunde:4711:LetzteBestellung
```

und so weiter. Als Trennzeichen wurde hier ein Doppelpunkt (colon) verwendet, der im Redis-Umfeld als anerkannte Konvention gilt. Das ist aber keine feste Einschränkung, man

kann stattdessen tatsächlich auch jedes beliebige andere Trennzeichen wählen. Um solcherlei zusammenhängende, strukturierte Daten in Redis zu speichern, gibt es aber noch weitaus bessere Optionen, auf die wir später noch zurückkommen werden.

Eine weitere wichtige Eigenschaft von Redis ist die garantierte atomare Ausführung jedes Befehls. Das wird dadurch erreicht, dass der Redis-Server alle eingehenden Befehle und Abfragen nacheinander in einem einzigen Thread abarbeitet. Das ist nicht unbedingt ein Nachteil – zum einen hält es die nötige Infrastruktur klein, da man faktisch auf jegliche Sperren verzichten kann, was wiederum gut für die Performance ist. Zum anderen sollte man sich vor Augen halten, dass auch so populäre Systeme wie Node.js single-threaded laufen, was übrigens auch der Grund dafür ist, dass sich Letzteres zwar prima für I/O-lastige Aufgabenstellungen eignet, hingegen weniger für CPU-lastige Tasks.

Ein analoges Problem hat man bei Redis erst dann, wenn man die Möglichkeit nutzt, in Lua geschriebene Skripts auszuführen. Genau wie alle anderen Befehle wird auch ein sol-

Die Terminologie innerhalb StackExchange

Redis ist einerseits durchdacht, dennoch gewöhnungsbedürftig.

Zwar sind die Befehle in Analogie zu Redis selbst sauber in logische Gruppen eingeteilt und jeweils durch ein gemeinsames Präfix gekennzeichnet. Das hat den Vorteil, dass man alle Funktionen für Hashes, Sets und Listen schnell anhand des Präfixes auffinden kann. Leider provoziert das verwendete Namensschema bei flüchtiger Betrachtung auch ärgerliche Missverständnisse.

Die Funktion `SetRandomMember()` setzt eben gerade keinen Wert, sondern es handelt sich um eine Set-bezogene Funktion, die einen Wert zurückliefert. Wenn im Redis-Umfeld von Hash-Funktionen die Rede ist, sind damit fast nie Algorithmen wie SHA-256 gemeint, sondern Befehle, die sich auf die Redis-Datenstruktur der Hashes beziehen. Um die Verwirrung komplett zu machen, gibt es auch noch Hash-Tags innerhalb von Redis-Schlüsseln – diese allerdings bezeichnen wiederum etwas völlig anderes.

ches Lua-Skript im Ganzen atomar ausgeführt. Daraus folgt allerdings auch unmittelbar, dass ein einziges, schlecht geschriebenes Lua-Skript den Bearbeitungs-Thread und damit effektiv den kompletten Redis-Server lahmlegen kann. Auf die Erweiterbarkeit mittels Lua werden wir im Folgenden nicht weiter eingehen, es sei daher in diesem Fall auf die umfangreiche Online-Dokumentation [3] verwiesen.

Dem aufmerksamen Leser ist sicherlich nicht entgangen, dass die Formulierung bezüglich der atomaren Ausführung sich auf einzelne Befehle bezog. Das ist auch korrekt und bedeutet nichts anderes, als dass eine aus mehreren Befehlen bestehende Sequenz wie die Folgende eben nicht zwingend atomar ausgeführt wird und folglich unter Umständen fehlerhafte Ergebnisse zeitigt (Pseudocode):

```
saldo1 = GET konto:1:saldo
saldo2 = GET konto:2:saldo
SET konto:1:saldo saldo1 + betrag
SET konto:2:saldo saldo2 - betrag
```

Dennoch ist auch das mit Redis durchaus zuverlässig umsetzbar, die Auflösung dieses Problems folgt etwas weiter unten.

Installation und Setup

Redis selbst kann in der Download-Abteilung auf der Webseite [3] einfach heruntergeladen und mit recht wenigen Handgriffen installiert werden. In vielen Linux-Distributionen ist Redis natürlich auch via Paketmanager installierbar, wobei das dann aber nicht unbedingt auch immer die neueste Version sein wird. Normalerweise muss man nach dem plattformspezifischen Setup nur das Konfigurationstemplate kopieren und kann den Server auch schon mit sinnvollen Voreinstellungen starten. Empfehlenswert ist lediglich noch die Vergabe eines Passworts in besagter Konfigurationsdatei – darüber hinaus bietet Redis nämlich keinerlei Sicherheitsmechanismen an.

Für Windows ist Redis ebenfalls verfügbar, das MS-Open-Tech-Projekt unterhält einen sehr gut gepflegten Windows-Port der Software auf GitHub [4]. Hier kann man aktuelle Versionen für 64-Bit-Systeme als fertiges Setup-Paket herunterladen. Die 32-bittige Version spielt für Server, auf denen Redis hauptsächlich verwendet wird, kaum noch eine Rolle, weshalb die 32-Bit-Version auch nicht mehr als Download verfügbar ist. Allerdings ist es bei Bedarf mit relativ wenig Aufwand möglich, den GitHub-Sourcode zu klonen und sich daraus selbst eine passende Version zu kompilieren. Aktuelle Verlautbarungen nach wird es die 32-Bit-Unterstützung auch weiterhin geben, nur wird der Build eben nicht mehr als offizielles Download-Paket verfügbar gemacht.

Apropos GitHub: Hier ist auch die C#-Bibliothek zu finden, die StackExchange selbst für ihre diversen Websites verwendet. Das Projekt mit dem leicht zu merkenden Namen StackExchange.Redis ist nicht nur aktiv gepflegt, sondern auch erprobt und bietet dem geneigten C#-Entwickler einen wirklich erstklassigen, Mono-tauglichen Abstraktionslayer über das native, Socket-basierte API von Redis an. Darüber hinaus kann man das Paket auch bequem per Nuget installieren. So-

Listing 1: Einbindung von StackExchange.Redis

```
static private ConnectionMultiplexer Redis = null;

static private void InitializeRedisConnection() {
    var options = new ConfigurationOptions();

    options.EndPoints.Add("localhost");

    options.EndPoints.Add("localhost", 7000);
    options.EndPoints.Add("localhost", 7001);
    options.EndPoints.Add("localhost", 7002);
    options.Password = "mypassword";
    Redis = ConnectionMultiplexer.Connect(options);
}
```

fern nicht anders angegeben, setzen alle folgenden Quellcode-Beispiele diese Bibliothek voraus, wobei für die bibliotheksspezifischen Details und Optionen zur Einbindung selbst wiederum auf die entsprechende Dokumentation [5] verwiesen sei. Auf derselben Plattform sind unter [8] auch alle Beispiele zu diesem Artikel zu finden.

Im Listing 1 ist deshalb nur eine minimale Einbindung gezeigt. Die Clientapplikation initialisiert hier eine einzige globale Instanz, die sich auf einem der vier möglichen Ports verbinden wird, bis ein geeigneter Server gefunden wird. Im Fall von StackExchange.Redis kann dieser ConnectionMultiplexer threadübergreifend verwendet werden. Genauer gesagt, wird diese Verfahrensweise sogar explizit empfohlen, ist aber eben spezifisch für diese eine Bibliothek und nicht verallgemeinerbar. Verwenden Sie eine andere Bibliothek oder gar eine andere Zielsprache, konsultieren Sie bitte die entsprechende Dokumentation.

Zum Abschluss der Installation sollte man sich unbedingt nochmals vergewissern, dass die Firewall-Einstellungen für alle von Redis verwendeten Ports korrekt gesetzt sind. Da praktisch die einzige eingebaute Sicherheitsbarriere aus einem im Klartext in einer Konfigurationsdatei abgelegten Passwort besteht, sind über das Web oder Intranet erreichbare Redis-Server immer ein dankbares Ziel für Angriffe.

Als weiteren Schutz lassen sich einzelne Kommandos auch per Konfiguration abschalten, jedoch stellt auch das letztlich nur eine Hürde dar. Zudem erfordert es immer das Abwägen, welche Kommandos man abschalten kann und welche besser nicht, mit der latenten Gefahr, etwas Wichtiges zu übersehen. Im Idealfall installiert man Redis also gar nicht erst auf einer Maschine, die über das Internet erreichbar ist.

Ereignisse abonnieren mit Publish and Subscribe

Moment mal – hatten wir nicht weiter oben erwähnt, dass Redis ein Key-Value-Store ist, also eine Datenbank? Wieso steht dann da jetzt *Publish and Subscribe*? Ganz einfach – weil Redis neben der Datenbankfunktionalität eben auch als Publish-Subscribe-Server eingesetzt werden kann. Der typische Einsatzfall hierfür ist auf jeden Fall das applikationsüber- ►

greifende Messaging zur Information über verschiedenste Ereignisse. So bietet etwa der ELK-Stack [2] eine Schnittstelle zur Verarbeitung eingehender Redis-Events an.

Dank der unkomplizierten Schnittstelle lässt sich gerade Publish-Subscribe auch nativ recht einfach implementieren, in C# könnte dies mittels StackExchange.Redis etwa so wie in den folgenden Beispielen aussehen. Um die in Listing 2 verschickten Events zu empfangen, registriert man zuvor einen oder mehrere Event Handler.

Wie Listing 3 zeigt, kann man dazu entweder einen voll qualifizierten Namen angeben, alternativ sind aber auch Wildcards möglich, um ganze Ereignisgruppen zu abonnieren. Die Daten sind im Beispiel zwar recht simpel gehalten, können in der Praxis aber natürlich nahezu beliebig komplex sein.

Redis als Datenbank

Auch wenn Publish-Subscribe mit Redis schnell aufgesetzt ist und faktisch sofort problemlos funktioniert, ist die Kernfunktionalität von Redis natürlich nach wie vor die Speicherung von Daten. Für simple Fälle und wenig strukturierte Daten wird man die Daten analog zu Listing 4 als einfache Schlüssel-Wert-Paare ablegen. Das lässt sich optional mit einem Verfallsdatum kombinieren, der sogenannten Time To Live (TTL).

Daten mit Verfallsdatum sind für zeitabhängige Informationen äußerst nützlich, man kann darüber aber auch die Gültigkeit von in Redis gecachten Daten steuern und somit erzwingen, dass der Cache bei sich schnell ändernder Datenbasis auch regelmäßig aktualisiert wird. Wichtig ist noch zu wissen, dass die TTL bei jeder Schreiboperation auf dem Datensatz auch wieder neu angegeben werden muss.

Wird die TTL nicht angegeben, sind die Daten zumindest theoretisch zunächst unbegrenzt haltbar. In der Praxis kann dies aber trotzdem noch durch verschiedene Ereignisse limitiert werden. Redis ist ja ein Hauptspeicherbasiertes System. Zwar sichert es seine Daten regelmäßig in konfigurierbaren Abständen in ein Logfile, dennoch kann es vorkommen, dass bei plötzlichem Ausfall des Systems Daten verlorengehen. Darüber hinaus hat Redis einen (ebenfalls konfigurierbaren) Mechanismus, der dafür sorgt, dass der vorhandene RAM nicht überlastet wird.

Der Konfigurationsschlüssel hierzu hört auf den Namen *maxmemory-policy*. Die Einstellung ist per Default so konfi-

Listing 2: Pub-Sub-Queues publizieren

```
var sub = Redis.GetSubscriber();
sub.Publish("Work.Hard", 1);
sub.Publish("Work.Harder", 99999);
sub.Publish("Work.Smarter", 42);
sub.Publish("Forecast.Weather.Today",
    "Sunny, 30°C");
```

Listing 3: Work-Events über Wildcard abonnieren

```
var sub = Redis.GetSubscriber();
sub.Subscribe("Work.*", (channel, nr) => {
    Console.WriteLine("Your lucky number is {0}",
        nr);
});

// Wetter von heute abonnieren und empfangen

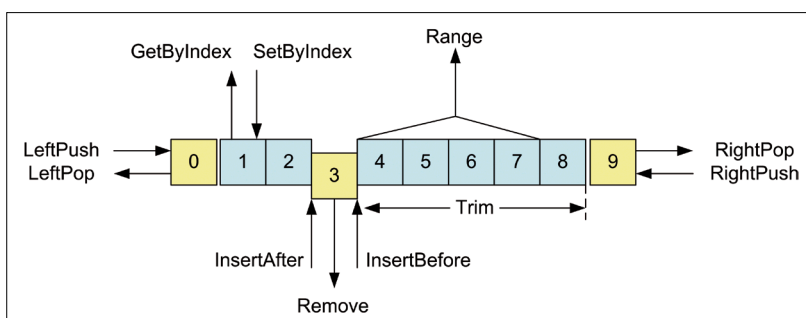
var sub = Redis.GetSubscriber();
sub.Subscribe("Forecast.Weather.Today", (channel,
    msg) => {
    Console.WriteLine("Weather today: {0}", msg);
});
```

guriert, dass bei Überschreitung der Speichergrenze Werte anhand eines LRU-Algorithmus verworfen werden – also die Werte, auf die am längsten nicht zugegriffen wurde. Für weitere Einstellmöglichkeiten sei wiederum auf die Dokumentation verwiesen.

Strukturierte Daten

Gerade für strukturierte Daten ist das Speichern einfacher Schlüssel-Wert-Paare doch recht mühselig. Zudem ist es schwierig, zusammengehörige Informationen auszulesen. Abhilfe bieten hier Hashes, die genau diese beiden Probleme lösen. Hashes in Redis stellen zunächst wiederum einen Schlüssel-Wert-Speicher dar. Jedoch ist es möglich, mehrere solcher Schlüssel-Wert-Paare unter einem gemeinsamen Schlüssel zusammenzufassen. Dieser Schlüssel könnte wie im Listing 5 aus Typ und ID bestehen, zum Beispiel der Kennung *Kunde* und einer Kundennummer. Die Werte der Instanz legt man dann als Schlüssel-Wert-Paare unterhalb dieses Hashes ab.

Das hat verschiedene Vorteile. Unter anderem ist es somit möglich, mit Hilfe der diversen HashXxx-Funktionen beispielsweise alle Schlüssel einfach zu iterieren, sei es mit den zugehörigen Datenwerten oder ohne diese. Im Gegensatz zum einfachen Key-Value-Verfahren sind Operationen auf Redis-Hashes in der Regel überaus performant. Ein Manko haben



Listenbefehle: Für Listen stellt Redis einen ganzen Satz Funktionen bereit (Bild 1)

Listing 4: Werte speichern und lesen

```

const string key = "Kunde:1";
db.StringSet(key + ":Name", "Mustermann");
db.StringSet(key + ":Vorname", "Mario");
db.StringSet(key + ":Saldo", -815.47);

if (db.StringIncrement(key + ":Mahnungen") >= 3)
{
    // Drei offene Mahnungen? Sperre für 30 Tage
    // setzen, falls nicht bereits gesperrt!
    db.StringSet(key + ":Sperre", true,
        TimeSpan.FromDays(30), When.NotExists,
        CommandFlags.FireAndForget);
}

Console.WriteLine("Informationen über Kunde #1");
var keys = new RedisKey[5] {
    key + ":Name",
    key + ":Vorname",
    key + ":Saldo",
    key + ":Mahnungen",
    key + ":Sperre"
};
var data = db.StringGet(keys);
for (var i = 0; i < keys.Length; ++i)
{
    Console.WriteLine("{0}: {1}", keys[i], data[i]);
}
Console.WriteLine("");

// Ausgabe:
// Informationen über Kunde #1
// Kunde:1:Name: Mustermann
// Kunde:1:Vorname: Mario
// Kunde:1:Saldo: -815.470000000000003
// Kunde:1:Mahnungen: 4
// Kunde:1:Sperre: 1

```

Hashes aber dennoch. Sie unterstützen nämlich kein TTL, für solche Fälle ist man auf normale Datenwerte angewiesen.

Bitte hinten anstellen!

Ein häufiger Anwendungsfall in jedem Programm sind Listen, die gleichartige Einträge zusammenhalten. Auch hierfür stellt Redis einen ganzen Satz Funktionen bereit. Über die Gruppe der Listenbefehle (**Bild 1**) lassen sich Einträge einfügen, über ihren Index ansteuern, entnehmen, lesen und aktualisieren. Neben den üblichen Einsatzfällen sind Redis-Listen aber noch aus einem anderen Grund interessant.

Gerade bei verteilten Anwendungen, aber auch beim Einsatz eines Threadpools, wird gern eine Warteschlange oder Queue verwendet, um Arbeitspakete flexibel speichern und für die Bearbeitung wieder entnehmen zu können. In Redis

Listing 5: Werte per Hash gruppieren

```

HashEntry[] fields = {
    new HashEntry( "Name", "Musterfrau" ),
    new HashEntry( "Vorname", "Traudel" ),
    new HashEntry( "Saldo", 14.99 )
};

const string key = "Kunde:2";
db.HashSet( key, fields);

// Guthaben! Schnell etwa Werbung verschicken ...
if (db.HashIncrement(key, "Flyer") < 5)
{
    // -> Hash unterstützt leider kein TTL ...
    db.StringSet(key + ":Sperre", true,
        TimeSpan.FromDays(14), When.NotExists,
        CommandFlags.FireAndForget);
}

Console.WriteLine("Informationen über Kunde #2");
foreach (var entry in db.HashGetAll(key))
{
    Console.WriteLine( "{0}: {1}", entry.Name,
        entry.Value);
}

// Ausgabe:
// Informationen über Kunde #2
// Name: Musterfrau
// Vorname: Traudel
// Saldo: 14.99
// Flyer: 2

```

gibt es zwar keine speziellen Queue-Befehle, das ist aber auch gar nicht nötig. Denn die Listen bringen bereits alles Notwendige für eine typische FIFO-Queue mit. Erforderlich ist hier lediglich eine Vereinbarung, welches Ende der Liste die Einfüllseite und welches die Entnahmeseite der Queue sein soll.

Nehmen wir an, dass die Worker links ihre Aufgaben entnehmen und etwaige Clients per Vereinbarung ihrerseits zu bearbeitende Aufgaben rechts einstellen. Der entsprechende Code könnte dann so aussehen wie in **Listing 6** und **Listing 7**. Die Arbeitsaufgaben werden über ein beliebiges Verfahren serialisiert, hierfür bietet sich beispielsweise JSON als platzsparendes, aber dennoch gut lesbares Format an. Die notwendige Synchronisation erledigt wiederum Redis, durch die garantierte Atomizität jedes Aufrufs sind unsererseits keine weiteren Vorkehrungen nötig.

Analog lassen sich Listen durch flexible Kombination der Push- und Pop-Befehle als auch LIFO-Queues, Stacks oder dergleichen einsetzen. Solange sich alle Parteien an das vereinbarte Protokoll halten, funktioniert alles reibungslos und erfordert lediglich minimalen Implementationsaufwand. ►

In Programmen und im realen Leben haben wir es häufig mit Sets zu tun. Mit Mengen also, deren Elemente zum einen bekannt sind und zum anderen höchstens einmal vorkommen können. Ein handelsübliches und regelkonformes Skatblatt etwa besteht aus 32 Karten, von denen der typische Skatspieler zu Beginn jeden Spiels 10 Karten auf der Hand hat (Bild 2).

Obwohl es vier Farben und pro Farbe je acht Kartenwerte gibt, ist jede Karte nur einmal im Spiel, somit handelt es sich aus Entwicklersicht um ein Set aus bis zu 32 möglichen Elementen. Ganz analog verhalten sich Redis-Sets, auch hier wird vom System sichergestellt, dass jeder Wert entweder nicht oder höchstens einmal im Set auftaucht (Listing 8).

Listing 6: Client fügt Aufgaben in eine Queue ein

```
internal static void Run() {
    const string QUEUE = "Test:Workpacks";
    var db = Redis.GetDatabase();

    while (true)
    {
        var work = new Workpack() {
        };

        RedisValue[] data = { Tools.Serialize(work) };

        db.ListRightPush( QUEUE, data,
            CommandFlags.FireAndForget);
        Console.WriteLine(".");
    }
}
```

Listing 7: Worker verarbeiten die Aufgaben

```
internal static void Run() {
    const string QUEUE = "Test:Workpacks";
    var db = Redis.GetDatabase();

    while (true)
    {
        var msg = db.ListLeftPop(QUEUE);
        if( msg.IsNullOrEmpty)
        {
            Thread.Sleep(0);
            continue;
        }

        var work = Tools.Deserialize<Workpack>((byte[])
            msg);
        if (work != null)
            ProcessWorkPack( work);
    }
}
```

Richtig Spaß machen Mengen aber erst mit dem Einsatz von Mengenoperatoren, mit denen sich Mengen vereinigen und voneinander abziehen lassen. Oder man bildet die Schnittmenge aus zwei Mengen. Nachdem eine Reihe Datensätze mit Geburtstagen eingepflegt wurde, können wir sowohl für einen gegebenen Tag als auch für einen Monat alle zugehörigen Namen schnell ermitteln, indem wir uns das entsprechende Set geben lassen. Um nun die Menge aller bekannten Personen zu ermitteln, die an einem ganz bestimmten Datum Geburtstag haben, bilden wir einfach die Schnittmenge aus den beiden Sets.

Einsatz im Webumfeld

Denkbare Einsatzfälle im Webumfeld wäre die Zählung von Zugriffen auf bestimmte Webseiten oder die Ermittlung der Unique Visitors pro Zeiteinheit. Um Duplikate bei mehrfachem Zugriff auf die Seite auszufiltern, wird jeder Besucher bei jedem Zugriff immer dem aktuell aktiven Set hinzugefügt. Am Ende des Zeitraums können dann alle enthaltenen Einträge entnommen werden, wobei alle Duplikate bereits durch Redis entfernt wurden.

Weitere interessante Möglichkeiten bieten sich durch die Option, Redis ein zufälliges Element aus einem Set liefern zu lassen (Listing 9). Analog zu *SetRandomMember()* liefert auch *SetPop()* ein zufälliges Element. Der Unterschied zwischen den beiden Varianten liegt darin, dass *SetPop()* das Element

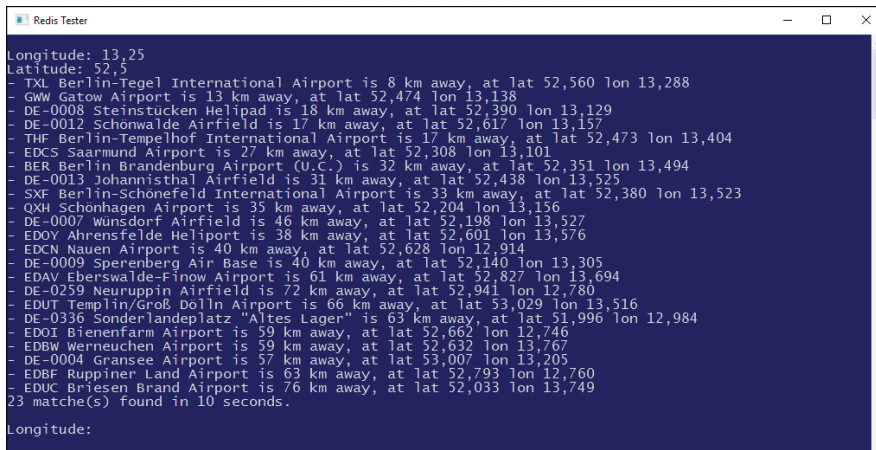
Listing 8: In Sets kommt jedes Element einmal vor

```
var db = Redis.GetDatabase();
db.SetAdd( "farben", "rot");
db.SetAdd( "farben", "gelb");
db.SetAdd( "farben", "grün");
db.SetAdd( "farben", "gelb");
db.SetAdd( "farben", "rot");
db.SetAdd( "farben", "blau");

farben = db.SetMembers("farben");
Console.WriteLine( "{0} Elemente", members.Length);
foreach (var farbe in farben)
    Console.WriteLine(farbe);

Console.WriteLine("");
var zufall = db.SetRandomMember("farben");
Console.WriteLine("Eine zufällige Farbe: {0}",
    zufall);

// Ausgabe:
// 4 Elemente
// grün
// gelb
// rot
// blau
//
// Eine zufällige Farbe: gelb
```

Mengenoperationen über sortierten Sets in Aktion (Bild 2)

auch gleich aus dem Set entfernt, während *SetRandomMember()* das nicht tut. Weiter ließen sich über Sets eindeutige Verbindungen zwischen anderen Daten herstellen, um etwa Graphen-Strukturen zu modellieren.

Gelegentlich ist es darüber hinaus auch wünschenswert, wenn man Einträgen eines Sets eine Bewertung mitgeben könnte. Der klassische Einsatzfall wäre ein sogenanntes Leaderboard, beispielhaft skizziert in Listing 10. Ein Leaderboard zeigt den aktuellen Stand aller Beteiligten an, sortiert nach Rang oder erreichten Punkten. Leaderboards sind im Zuge der Gamification-Welle schon lange nicht mehr auf reine Spiele beschränkt, man findet sie zum Beispiel auch bei Stackoverflow.com oder ähnlichen Seiten, bei denen man für die unterschiedlichsten Aktivitäten Punkte einheimsen (oder verlieren) kann.

Genau das ist mit Sorted Sets möglich. Diese verhalten sich tatsächlich ähnlich wie Sets, haben aber ein eigenes Satz an Befehlen. Jeder Eintrag eines sortierten Sets besteht aus dem Wert selbst und einem zugeordneten Score. Das Set sor- ►

Listing 9: Mengen lassen sich kombinieren

```
private const string keyTest = "Tests:Set";

internal void Execute() {
    var db = Redis.GetDatabase();
    FillKnownBirthdays();

    Console.Write("Sage mir deinen Geburtstag! ");
    Console.Write("Zuerst den Tag (1..31) ");
    int tag = Int32.Parse(Console.ReadLine());

    Console.Write("Primal Jetzt noch den Monat (1..12) ");
    int monat = Int32.Parse(Console.ReadLine());
    Console.WriteLine("");

    var keyTag = keyTest + ":day:" + tag.ToString();
    var keyMonat = keyTest + ":month:" + monat.ToString();

    var matches = db.SetMembers(keyMonat);
    Console.WriteLine("Im selben Monat haben Geburtstag:");
    foreach (var match in matches)
        Console.WriteLine("- " + match);
    Console.WriteLine("");

    matches = db.SetMembers(keyTag);
    Console.WriteLine("Am selben Tag im Monat:");
    foreach (var match in matches)
        Console.WriteLine("- " + match);
    Console.WriteLine("");

    matches = db.SetCombine(
        SetOperation.Intersect,
        new RedisKey[] { keyTag, keyMonat }
    );
    Console.WriteLine("Mit dir haben Geburtstag:");
    foreach (var match in matches)
        Console.WriteLine("- " + match);
    Console.WriteLine("");
}

private void AddBDay(int day, int month, string name)
{
    var db = Redis.GetDatabase();
    db.SetAdd(keyTest + ":day:" + day.ToString(), name);
    db.SetAdd(keyTest + ":month:" + month.ToString(), name);
}

private void FillKnownBirthdays() {
    // http://www.xemico.com/.../
    FamousPeopleBirthdays.adc
    AddBDay(3, 1, "J.R.R.Tolkien");
    AddBDay(3, 1, "Cicero");
    AddBDay(6, 1, "Joan of Arc");
    AddBDay(8, 1, "Elvis Presley");
    AddBDay(8, 1, "Stephen Hawking");
    AddBDay(15, 1, "Martin Luther King Jr");

    // ... weitere Daten ...
}
```

tiert alle Einträge anhand des Score-Werts, und man kann auch über die Scores suchen. Ganz allgemein sind Sorted Sets für jede Art numerischer Sortieraufgaben geeignet.

Da die Score-Werte innerhalb Redis als Gleitkommazahlen abgebildet werden, lassen sich damit auch ganz praktische Anwendungen bauen.

Die wesentlichsten Daten aus der Liste aus Flughäfen und deren Koordinaten von [6] lassen sich in Form von vier Elementen in Redis abbilden: zwei Sorted Sets aus Airport-IDs, deren Score der Längen- beziehungsweise Breitengrad ist; sowie je ein Hash zum Mappen der IDs auf Bezeichnungen und internationale Flughafencodes. Damit lassen sich die nächsten 20 Flughäfen um einen gegebenen geografischen Punkt, etwa wie im Listing 11, ermitteln.

Mengenoperationen

Der gesamte Algorithmus baut komplett auf Mengenoperationen auf. Um den gegebenen Ausgangspunkt herum wird der Suchradius so lange geometrisch vergrößert, bis wenigstens 20 Treffer gefunden wurden. Die Treffer im aktiven Suchfeld werden ermittelt, indem die Schnittmenge aus den Treffern auf Längen- und Breitengrad ermittelt wird. Diese Treffer wurden zuvor anhand einer Range-Operation über den Scores ermittelt, der ja dem Längen- beziehungsweise Breitengrad entspricht. Damit der Anwender nicht zu lange warten muss – man stelle sich dazu probenhalber Ausgangskordinaten im Pazifik vor – werden gefundene Treffer sofort ausgegeben.

Da bei größerem Radius natürlich auch wieder die Treffer der vorherigen Runde gefunden werden, müssen diese in einem Hilfs-Set zwischengespeichert werden. Damit ist es in der nächsten Runde möglich, die Differenzmenge der aktuellen Treffermenge mit der schon bekannten Treffermenge zu bilden und nur die neu hinzugekommenen Treffer auszugeben. Zusätzlich ergibt sich der nette Nebeneffekt, dass die Treffer mit der kürzesten Entfernung tendenziell auch weiter oben stehen.

Listing 10: Sorted Sets

```
private static void LeaderBoard() {
    const string key = "Tests:sortedsets";
    var db = Redis.GetDatabase();

    db.SortedSetRemoveRangeByRank(key, 0, -1);

    db.SortedSetAdd(key, "Alice", -10);
    db.SortedSetAdd(key, "Bob", -20);
    db.SortedSetAdd(key, "Claire", -30);
    db.SortedSetAdd(key, "Doris", -40);

    db.SortedSetAdd(key, "Egon", -30);

    var score = db.SortedSetScore(key, "Bob").Value;
    db.SortedSetAdd(key, "Bob", score-40);

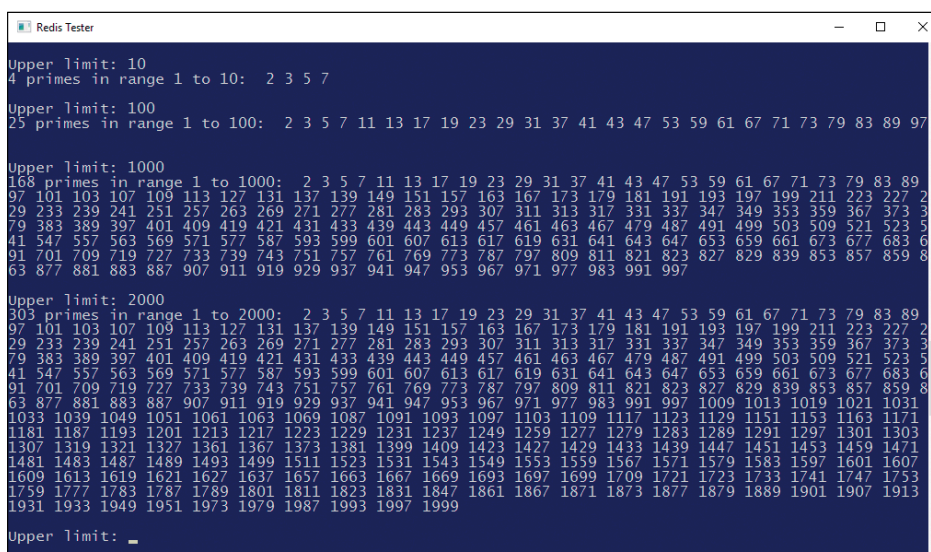
    var entries =
        db.SortedSetRangeByRankWithScores(key);
    Console.WriteLine("Punkte Name");
    foreach (var x in entries)
        Console.WriteLine("{0,6} {1}", -x.Score,
            x.Element);
}
```

In der gezeigten Form hat der Algorithmus noch einige eklatante Schwächen. So wird etwa der Bereich um die Datumsgränze nicht korrekt berücksichtigt, die etwas simple Kilometerberechnung funktioniert nur in Äquatornähe richtig und das Suchfeld ist aufgrund der Schnittmengenbildung nicht wirklich rund, sondern rechteckig. Dennoch ergibt sich rein mit Redis-Bordmitteln bereits eine grob funktionierende Lösung, die je nach Anwendungsfall durchaus auch bereits völlig ausreichend sein kann.

Bitte ein Bit – oder auch ein paar mehr

In der Computerei dreht sich bekanntlich alles um Nullen und Einsen, binäre Zustände also. Sein oder Nichtsein. Für solche tiefeschürfenden, philosophischen Fragen hat der pragmatische Programmierer ein probates Mittel parat: Er zeigt Flagge und setzt einfach ein Flag. Braucht man mal ein paar mehr davon, werden schon auch mal ganze Bitfelder angelegt.

Dank Redis ist auch das Bestellen und Beackern solcher Bitfelder kein Problem, da Redis nämlich eingebaute Funktionalität hierfür mitbringt. Intern werden



Primzahlen sieben mit Bitsets (Bild 3)

Listing 11: Nächsten Flughafen ermitteln

```

Redis = redis;
var db = Redis.GetDatabase();

Console.Clear();
const string keyTest = "{Tests:SortedSets}";

if (db.HashLength(keyTest + ":CodeByIdent") < 1) {
    EmptySets();
    FillAirports();
}

while (true) {
    double lon, lat, range;

    Console.WriteLine("");
    Console.Write("Longitude: ");
    if (!Double.TryParse(Console.ReadLine(), out lon))
        break;
    Console.Write("Latitude: ");
    if (!Double.TryParse(Console.ReadLine(), out lat))
        break;

    while( db.SetLength(keyTest + ":matchLast") > 0)
        db.SetPop(keyTest + ":matchLast");

    range = 0.001;
    var funde = 0;
    var start = DateTime.Now;
    while (funde < 20) {
        Console.WriteLine("\rSearching {0} km range, {1:N0}
secs and {2} matches so far ... ", DegToKm(range),
(DateTime.Now - start).TotalSeconds, funde);

        var matchLat = db.SortedSetRangeByScoreWithScores
(keyTest + ":IdentByLat", lat - range, lat +
range);
        var matchLon = db.SortedSetRangeByScoreWithScores
(keyTest + ":IdentByLon", lon - range, lon +
range);

        while (db.SetLength(keyTest + ":latMatch") > 0)
            db.SetPop(keyTest + ":latMatch");
        foreach (var match in matchLat)
            db.SetAdd(keyTest + ":latMatch", match.Element);

        while (db.SetLength(keyTest + ":lonMatch") > 0)
            db.SetPop(keyTest + ":lonMatch");
        foreach (var match in matchLon)
            db.SetAdd(keyTest + ":lonMatch", match.Element);

        db.SetCombineAndStore(
            SetOperation.Intersect,
            keyTest + ":matchThis",
            new RedisKey[] {
                keyTest + ":lonMatch",
                keyTest + ":latMatch"
            });

        db.SetCombineAndStore(
            SetOperation.Difference,
            keyTest + ":matchNew",
            new RedisKey[] {
                keyTest + ":matchThis",
                keyTest + ":matchLast"
            });

        db.SetCombineAndStore(
            SetOperation.Union,
            keyTest + ":matchLast",
            new RedisKey[] {
                keyTest + ":matchThis",
                keyTest + ":matchLast"
            });

        foreach (var elm in db.SetMembers(keyTest +
":matchNew")) {
            var ident = elm;
            var code = db.HashGet(keyTest + ":CodeByIdent",
elm);
            var name = db.HashGet(keyTest + ":NameByIdent",
elm);
            var lat_found = db.SortedSetScore(keyTest +
":IdentByLat", elm).Value;
            var lon_found = db.SortedSetScore(keyTest +
":IdentByLon", elm).Value;
            var km = DegToKm(
                Math.Sqrt(
                    (lat_found - lat) * (lat_found - lat) +
                    (lon_found - lon) * (lon_found - lon)));

            ++funde;
            Console.WriteLine(
                "\r- {0} {1} is {2} km away, at lat {3:N3}
lon {4:N3}",
                code, name, km, lat_found, lon_found);
        }

        range = Math.Ceiling( 11000 * range) / 10000;
    }

    Console.WriteLine(
        "\r{0} matche(s) found in {1:N0} seconds.",
        funde, (DateTime.Now-start).TotalSeconds);
}

Console.Write("<ENTER> to close ");
Console.ReadLine();

```

auch diese Bitfelder wieder als Strings abgelegt, sodass sich eine Maximalgröße von 512 MByte = 2^{32} Bit ergibt. Damit kann man schon einiges anfangen.

Zum Beispiel kann man damit mal schnell das Sieb des Erathostenes implementieren, um Primzahlen zu finden (Listing 12). Zunächst wird dazu der gesamte Bereich mit gesetzten Bits initialisiert. Leider unterstützt Redis kein Bit-Kommando zum Setzen oder Löschen mehrerer Bits in einem Arbeitsschritt, deshalb muss man hier etwas Kreativität walten lassen (Bild 3).

Um also den Bereich von 0 bis zum Limit mit 1 zu initialisieren, setzen wir zunächst das letzte Bit des Bereichs auf 0. Diese für sich genommen scheinbar völlig sinnfreie Operation bewirkt aber, dass Redis den gesamten benötigten Speicherplatz auf einen Schlag alloziert und uns damit ein komplett mit 0 gefülltes Bitset in der richtigen Größe bereitstellt. Dieses Bitset müssen wir nun nur noch mit der entsprechenden Bitoperation namens *Bitwise.Negation* flink invertieren, und nach wenigen Millisekunden steht ein komplett mit 1 initialisiertes Bitset zur gefälligen Benutzung bereit. Einziger Wermutstropfen ist die Tatsache, dass Redis hinter den Kulissen

geringfügig mehr Bits alloziert als wir angefordert haben. Gibt man als Bereich etwa 1.000.000 (1 Million) ein und invertiert man das Bitset, so erhält man 1.000.008 gesetzte Bits.

Diese wenigen überzähligen Bits sind aber schnell gelöscht, außerdem müssen natürlich die beiden Bits an Index 0 und 1 noch als *nicht prim* markiert werden. Über weitere Redis-Bit-Befehle auf dem Datenfeld lassen sich nun Schritt für Schritt alle Nichtprimzahlen herausstreichen.

Natürlich kann man Bitsets nicht nur zum Ermitteln von Primzahlen gebrauchen. Sie eignen sich ganz allgemein für alle Arten von booleschen Werten, Zuordnungen oder dergleichen, die auch einmal in größerer Zahl auftreten können. Wie schon mit Sets könnte man jedem Bit ein 5-Minuten-Intervall zuordnen und die Anwesenheit eines Besuchers auf der Webseite tracken.

Mit mehreren Bitsets bekommt man auch heraus, welcher Besucher welchen Bereich der Seite betreten hat. Wird Redis in der Applikation zur Performancesteigerung eingesetzt, ließe sich mit Hilfe mehrerer Bitsets auch schnell ein Bloom-Filter implementieren. Man kann mit Bitsets also auf einfache und effiziente Art Dinge zählen, eingetretene Ereignisse mar-

Listing 12: Sieb des Erathostenes

```
Redis = redis;
var db = Redis.GetDatabase();
RedisKey keyPrimes = "Tests:FindingPrimes";

while (true) {
    Console.WriteLine("Upper limit: ");
    int bereich;
    if( ! Int32.TryParse( Console.ReadLine(),
        out bereich))
        break;
    bereich = Math.Max(2, Math.Min(Int32.MaxValue,
        bereich)) + 1;

    Console.WriteLine("\rInitializing ...");
    db.StringSet(keyPrimes, "");

    db.StringSetBit(keyPrimes, bereich, false);
    // allocate
    db.StringBitOperation(Bitwise.Not, keyPrimes,
        keyPrimes);

    var count = db.StringBitCount(keyPrimes);
    while (count > bereich)
        db.StringSetBit(keyPrimes, --count, false);
    Debug.Assert(db.StringBitCount(keyPrimes) ==
        bereich);

    db.StringSetBit(keyPrimes, 0, false); // not prime
    db.StringSetBit(keyPrimes, 1, false); // not prime

    long teiler = 2;

    long letzter = (long)Math.Ceiling( Math.Sqrt(
        bereich));
    while (teiler <= letzter) {
        count = db.StringBitCount(keyPrimes);
        Console.WriteLine(
            "\rCurrent = {0}, {1} candidates remaining ...",
            teiler, count);

        var ofs = 2 * teiler;
        while (ofs <= bereich) {
            db.StringSetBit(keyPrimes, ofs, false);
            ofs += teiler;
        }

        while (++teiler <= letzter)
            if (db.StringGetBit(keyPrimes, teiler))
                break;
    }

    count = db.StringBitCount(keyPrimes);
    Console.WriteLine(
        "\r{0} primes in range 1 to {1}: ",
        count, bereich-1);
    for (var i = 2; i <= bereich; ++i)
        if (db.StringGetBit(keyPrimes, i))
            Console.WriteLine(String.Format(" {0}", i));
    Console.WriteLine();
    Console.WriteLine();
}

Console.ReadLine();
```

kieren und vieles mehr. Natürlich geht das genauso auch mit Sets. Geht es aber lediglich um reine Ja/Nein-Werte, sind Bitsets und Bitoperationen meist die effizientere Alternative.

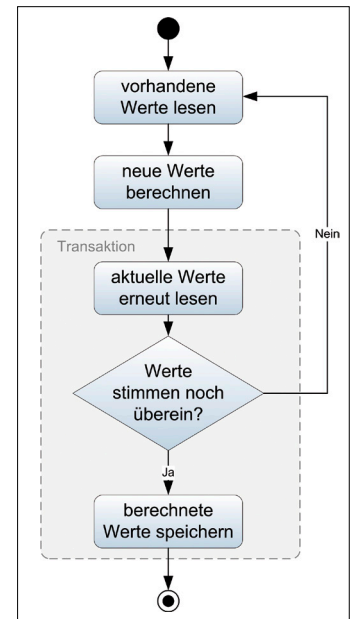
Locks und Transaktionen

Wie schon weiter oben erläutert ist jeder Aufruf in Redis immer garantiert atomar. Das ist für Operationen, die mit einem einzigen Aufruf abgehandelt werden können, auch ausreichend und bedarf keiner weiteren Maßnahmen zur Synchronisation. Nun kommt es aber in der Praxis durchaus vor, dass sich mehrere Clients über die Nutzung einer Ressource detaillierter abstimmen müssen. Das betrifft nicht nur in Redis selbst abgelegte Werte, sondern ganz allgemein jede beliebige Art begrenzter Ressourcen.

Das übliche Vorgehen in solchen Fällen ist, eine sogenannte logische Sperre zu setzen. Bei logischen Sperren wird nicht das Objekt selbst gesperrt, sondern eine Sperre über einem besser geeigneten Ersatzobjekt erstellt. Dieses Hilfsobjekt wird dann nach einem von allen beteiligten Parteien vereinbarten, einheitlichen Namensschema identifiziert.

Hierfür kommen häufig Systeme wie Zookeeper zum Einsatz, aber auch mit Redis und StackExchange. Redis lassen sich entsprechende Locking-Verfahren nutzen und implementieren. Im simpelsten Fall nutzt man dazu einfach die Lock-Funktion, die uns StackExchange.Redis anbietet (Listing 13). Das Verfahren funktioniert genauso einfach, wie es aussieht.

In der Praxis ist es jedoch wünschenswert, Algorithmen nach Möglichkeit lockfrei zu gestalten. Das lässt sich aber nur umsetzen, wenn das zugrunde liegende System eine Möglichkeit bietet, mehrere Operationen zusammenzufassen und als atomare Einheit auszuführen. Mit den Befehlen `multi` und `watch` kann man genau das implementieren, allerdings ist die händische Umsetzung über das native Socket-Interface mehr als umständlich. Hier spielen Biblio-



Compare and Store:

Jede Transaktion besteht aus zwei atomar abgearbeiteten Phasen (Bild 4)

Listing 13: Logische Sperren

```

var db = Redis.GetDatabase();
var key = "MyApp:Locks:Resource:100845";

var msec = TimeSpan.FromSeconds(30);
var lockId = Guid.NewGuid().ToString();
try {
    if( db.LockTake( key, lockId, msec))
    {
        if( db.LockExtend(key, lockId, msec))
        {
            // ... weitere Operationen ...
        }
    }
}
Finally {
    db.LockRelease( key, lockId);
}
  
```

Listing 14: Transaktionen in Redis

```

void Buchung( string ktoSoll, string ktoHab, double
betrag) {
    const string keyTest = "Tests:Transaction:";
    var db = Redis.GetDatabase();

    var soll_key = keyTest + ktoSoll;
    var haben_key = keyTest + ktoHab;

    bool success = false;
    while( ! success)
    {
        var soll = (double)db.HashGet(soll_key,
            "saldo");
        var haben = (double)db.HashGet(haben_key,
            "saldo");

        var trans = db.CreateTransaction();

        trans.AddCondition(
            Condition.HashEqual( soll_key, "saldo", soll));
        trans.AddCondition(
            Condition.HashEqual( haben_key, "saldo",
            haben));

        trans.HashIncrementAsync( haben_key, "saldo",
            betrag);
        trans.HashDecrementAsync( soll_key, "saldo",
            betrag);

        success = trans.Execute();
    }
}
  
```


theiken wie StackExchange.Redis ihre wahren Stärken aus, indem komplexe Operationen hinter einem sympathischen und nahezu intuitiven API verborgen werden.

Das grundsätzliche Prinzip hinter einer Redis-Transaktion folgt dem Compare-and-Store-Paradigma (Bild 4). Jede Transaktion besteht also aus zwei atomar abgearbeiteten Phasen, deren erste das Prüfen aller Ausgangsbedingungen ist. War diese Prüfung erfolgreich, schließt die zweite Phase an, in der die zuvor berechneten neuen Werte gespeichert werden.

Schlägt die Prüfung fehl, weil in der Zwischenzeit Daten geändert wurden und somit nicht mehr alle Randbedingungen erfüllt sind, folglich auch die zuvor berechneten neuen Werte falsch wären, wird die gesamte Operation abgebrochen. Wie das in der Praxis aussehen könnte, zeigt Listing 14.

An dieser Stelle soll jedoch nicht unerwähnt bleiben, dass Redis kein SQL-Server ist. Auch wenn die Terminologie die-

Timings hierfür lassen sich allesamt in der Konfigurationsdatei festlegen.

Hat man aber mehr Daten, als in einen einzelnen Master passen, bietet sich eine verteilte Lösung in Form eines Clusters an. Redis ist grundsätzlich clusterfähig, allerdings ist es ratsam, hierfür die Version 3 zu verwenden. Zum einen unterstützt die ältere Version nicht immer alle Befehle, und zweitens wurden in den neueren Versionen auch immer wieder Bugfixes eingearbeitet. Die Windows-Builds für Version 3 waren zum Zeitpunkt der Entstehung des Artikels zwar noch im Beta-Stadium, liefen aber auch im Clustermodus stabil.

Für den Clusterbetrieb gibt es einige Dinge zu beachten. Zuerst gilt es, die Nodes entsprechend zu konfigurieren. Erforderlich sind mindestens drei Nodes. Für ein produktives Minimal-Setup werden insgesamt sechs Server empfohlen, von denen drei als Masternodes als Cluster zusammengeschaltet werden. Die anderen drei Server werden als Slaves für je einen der Master konfiguriert. Wie das im Detail geht, ist in der Cluster-Dokumentation auf der Webseite gut erklärt und würde hier nur den Rahmen sprengen.

Für den praktischen Einsatz gibt es aber auch algorithmisch einige Fallstricke. Dazu muss man wissen, wie Redis die Daten auf die Rechner im Cluster verteilt. Das Problem ist, dass ein Server wie Redis nicht vorhersagen kann, welche Schlüssel verwendet werden und wie wohl deren Häufigkeitsverteilung aussehen wird. Deswegen greift man zu einem Trick und schickt einfach jeden Schlüssel durch eine Hashfunktion. Das Ergebnis teilt man über eine Modulo-Operation in eine zuvor festgelegte Anzahl Gruppen ein (Bild 5).

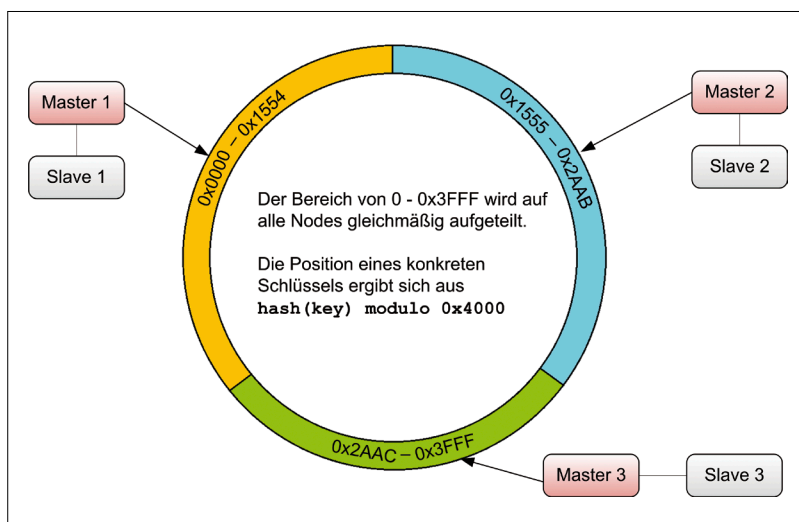
Diese Gruppen wurden bei der Einrichtung des Clusters festgelegt und möglichst gleichmäßig auf die vorhandenen Server verteilt. Dieses Vorgehen ist einfach und sichert den-

noch eine gleichmäßige Auslastung aller Nodes im Cluster ab. Darüber hinaus verringert es Konflikte durch Zugriffe auf benachbarte Schlüssel, da diese nach dem Hashing nicht nur in einem komplett anderen Bereich, sondern recht wahrscheinlich auch auf einen anderen Node liegen.

Der Sharding-Algorithmus

Solche Algorithmen stellen zwar gleichmäßige Lastverteilung sicher, haben aber auch Nachteile. Im Bezug auf Redis stellt sich nämlich die Frage, wie die Atomizität der Ausführung weiterhin garantiert werden wird? Die Antwort ist so einfach wie ernüchternd: Im Prinzip gar nicht. Redis erlaubt nämlich keine Operationen, die über mehrere Clusternodes gehen. Das betrifft nicht nur Transaktionen, sondern auch scheinbar simple Aufrufe wie *MULTI GET* zum Abruf mehrere Werte in einem Rutsch.

Das Problem hierbei ist nun: Woher weiß die Anwendung, welche Daten wo liegen, und wie kann man die Verteilung steuern, obwohl man sie ja eigentlich eben gerade nicht steuern können soll? Für diesen Zielkonflikt gibt es eine prakti-



Aufteilung des Keyspace im Cluster (Bild 5)

selbe zu sein scheint, so unterscheidet sich eine Transaktion in Redis in einigen Punkten von der handelsüblicher relationaler Datenbanken. Während in Letzteren das Ende einer Transaktion erst mit dem Commit oder Rollback besiegelt ist, führt Redis einfach stupide alle Schritte bis zum Ende aus, sofern alle Vorbedingungen erfüllt sind. Auch wenn einer der Befehle fehlschlägt, so werden dennoch alle weiteren Befehle ausgeführt.

Außerdem kennt Redis keine Konsistenz-Level wie *repeatable read* oder dergleichen. In diesem Zusammenhang sei auch auf die wie immer aufschlussreichen Jepsen-Tests von Kyle Kingsbury verwiesen, der auch Redis bezüglich CAP auf Herz und Nieren getestet hat [9].

Redis im Cluster-Modus

Ein einzelner Redis-Server ist bereits eine nette Sache. Um die Ausfallsicherheit zu erhöhen, kann man diesem einen Server auch noch ein oder mehrere Slaves zur Seite stellen, die alle Operationen des Master-Servers übernehmen und bei Ausfall relativ nahtlos einspringen können. Die genauen

Listing 15: Zuverlässiges Locking im Cluster

```

var db = Redis.GetDatabase();
const string TEMPLATE =
"[{Cluster:App:Object:{0}}]:{1}";

RedisKey MakeLockKey(string objectId) {
    return String.Format( TEMPLATE, objectId, "Lock");
}

RedisKey MakeDataKey(string objectId) {
    return String.Format( TEMPLATE, objectId, "Data");
}

bool LockAcquire(string Id, uint msec, out string
lockId) {
    RedisKey keyLock = MakeLockKey(Id);
    RedisKey keyData = MakeDataKey(Id);
    lockId = Guid.NewGuid().ToString();
    var time = TimeSpan.FromMilliseconds(msec);
    var trans = db.CreateTransaction();
    trans.AddCondition(Condition.KeyNotExists(keyLock));
    trans.StringSetAsync(keyLock, lockId, time);
    if( trans.Execute())
        return true;
    lockId = String.Empty;
    return false;
}

bool LockedAction( string Id, string lockId,
Action<ITransaction> action) {
    RedisKey keyLock = MakeLockKey(Id);
    Debug.Assert(!String.IsNullOrEmpty(lockId));
    var trans = db.CreateTransaction();

    trans.AddCondition(
        Condition.StringEqual(keyLock, lockId));
    action(trans);
    return trans.Execute();
}

bool LockExtend(string Id, string lockId, uint msec) {
    RedisKey keyLock = MakeLockKey(Id);
    return LockedAction(db, Id, lockId, (trans) => {
        var time = TimeSpan.FromMilliseconds(msec);
        trans.KeyExpireAsync(keyLock, time);
    });
}

void Unlock(string Id, string lockId) {
    RedisKey keyLock = MakeLockKey(Id);
    LockedAction(db, Id, lockId, (trans) => {
        trans.KeyDeleteAsync(keyLock);
    });
}

bool WriteAndUnlock(string Id, string lockId, string
data) {
    RedisKey keyLock = MakeLockKey(Id);
    RedisKey keyData = MakeDataKey(Id);
    return LockedAction(db, Id, lockId, (trans) => {
        trans.StringSetAsync(keyData, data);
        trans.KeyDeleteAsync(keyLock);
    });
}

```

kable und auch recht einfache Lösung, bei der allerdings in den Applikationscode eingegriffen werden muss. Formatiert man einen Redis-Schlüssel mit sogenannten Hash-Tags nach folgendem Schema:

```

{Eins:Zwei}
{Eins:Zwei}.Drei
{Eins:Zwei}:Drei:Vier

```

so verfüttert Redis nur den Teil innerhalb der geschweiften Klammern an die Hashfunktion. Das aber führt dazu, dass diese drei Schlüssel garantiert auf demselben Node landen.

Der Client weiß nun zwar immer noch nicht, welcher Node das genau ist, aber das braucht er für die erfolgreiche Abarbeitung des Algorithmus auch gar nicht zu wissen. Nachteilig ist eben nur, dass man eine erfolgreich und problemlos laufende Single-Node-Redis-Anwendung eben nicht mal einfach mit einem Cluster verbinden kann, da bestimmte Konstrukte dann nicht mehr ohne Weiteres funktionieren werden.

Apropos nicht funktionieren: Im Clustermodus funktioniert das Verlängern eines Locks mit der `StackExchange.Redis`-Bi-

bliothek aus irgendeinem unerfindlichen Grund in seltenen Fällen nicht korrekt – `LockExtend()` liefert zwar `true`, aber das nachfolgende `Unlock()` schlägt fehl – mit der Meldung, dass der Lock angeblich schon lange abgelaufen sei.

Der Verdacht liegt nahe, dass es sich hier um eine Race Condition handelt, da im Code einige `FireAndForget`-Flags verwendet werden, die das Zeitverhalten des Aufrufs ändern können. Glücklicherweise ist es nicht schwer, mit Hilfe von Transaktionen und geeigneter Namenswahl für alle beteiligten Schlüssel ein zuverlässiges Locking zu schreiben ([Listing 15](#)). Mit dieser Variante umgeht man überdies auch das latente Problem einer zu kurzen TTL für den Lock.

Korrekterweise muss man nämlich die Aktionen in eine Transaktion verpacken, innerhalb derer man als Vorbedingung zunächst prüft, ob der Thread den Lock überhaupt noch besitzt. Andernfalls besteht auch hier die latente Gefahr einer Race-Condition, wenn der Thread nämlich glaubt, noch im Besitz des Locks zu sein, dieser aber wegen ungünstigen Laufzeitverhaltens eines Teilalgorithmus schon längst abgelaufen ist. Leider kommen ungewöhnlich lange Laufzeiten in der Praxis durchaus vor. Auch wenn dies nur sporadisch ►

der Fall ist, so muss man doch damit rechnen. Schlecht performende Subsysteme, temporär überlastete Netzwerke oder ungeeignete Algorithmen, die auf zu viele Daten treffen, lassen sich ganz einfach nicht hundertprozentig ausschließen. Das sollte dennoch keinesfalls zu Datensalat führen.

Betreibt man mehrere unabhängige Redis-Server, also ohne diese in einen gemeinsamen Cluster einzubinden, wird das Management eines echten verteilten Locks über unabhängige Server schon deutlich komplexer und bedarf eines geeigneten 2-Phasen-Protokolls.

Da das Thema definitiv nicht mehr trivial genug ist, um im begrenzten Rahmen dieses Artikels Platz zu finden, sei an dieser Stelle ein weiteres Mal auf die Dokumentation verwiesen. Dort findet man einen Vorschlag für einen verteilten Red-Lock-Algorithmus, nebst Links zu diversen Implementierungen. Alternativ könnte man auch auf bewährte Algorithmen wie Paxos oder Raft zurückgreifen.

Bewährte Best Practices

Wie wir anhand einiger ausgewählter Beispiele gesehen haben, bietet sich mit Redis eine Fülle an Optionen und Möglichkeiten für vielerlei Einsatzfälle. Für den perfekten Einsatz dieses skalierbaren und performanten Systems ist es darüber hinaus ratsam, einige grundlegende Kniffe und Methodiken zu kennen.

Der erste Ratschlag ist, sich frühzeitig über die Aufteilung und Strukturierung des für Schlüsselbezeichner zur Verfügung stehenden Namensraums zu beschäftigen. Bevor man überhaupt damit beginnt, ernsthaft Daten in Redis zu speichern oder via Publish-Subscribe zu versenden, sollte analog zum Softwareprojekt selbst auch ein klares Konzept zur Namensstruktur der Schlüssel festgelegt worden sein.

In diesem Zusammenhang ist es zudem empfehlenswert, immer auch Raum für spätere Erweiterungen oder neuere Versionen der Datenstrukturen zu lassen. Ein bestimmter Bereich sollte reserviert werden, einerseits für Konfigurationsoptionen der verteilten Applikation, andererseits für häufig benötigte Eintrittspunkte in die Datenstrukturen. Letzteres könnten zum Beispiel die Wurzelknoten einer Baum- oder Graphen-Struktur sein, die so einfach und schnell lokalisiert werden können.

Der zweite Ratschlag bezieht sich auf den Wechsel in den Cluster-Modus. Wie wir gesehen haben, ist der Wechsel vom Single-Server zum Cluster nicht ganz so trivial und reibungslos, wie man meinen könnte. Beachtet man einige grundlegende Dinge, ist dieser Schritt zwar auch nicht wirklich kompliziert, dennoch sollte man die Applikation auch in diesem Umfeld getestet haben, bevor man damit in den produktiven Einsatz geht, um unliebsame Überraschungen zu vermeiden. Besonderes Augenmerk muss auch hier wieder auf die geeignete Benennung der Schlüssel gelegt werden, da man ja einerseits eine funktionierende Lösung haben möchte, andererseits aber eben auch die Daten auf alle Nodes möglichst gleichmäßig verteilt werden sollen.

Steht man vor der Forderung nach besserer Skalierung bei Lastspitzen, muss es aber nicht immer gleich ein Cluster sein. Gerade wenn Redis sowohl als Datenspeicher als auch als

Publish-Subscribe-System eingesetzt wird, kann es möglicherweise sinnvoller sein, diese beiden Aspekte einfach auf zwei getrennte Redis-Server zu verteilen. Neben der offensichtlichen Lastverteilung erlaubt dies auch ein zielgerichtetes Tuning der Konfigurationsparameter auf den jeweiligen Einsatzzweck hin.

Last not least ist es zweckmäßig, sich Gedanken über den sinnvollen Einsatz der TTL bei den gespeicherten Daten zu machen. Gerade bei einem RAM-basierten System wie Redis ergibt es keinen Sinn, wertvollen Hauptspeicher mit längst veralteten Daten zu blockieren. Insbesondere wenn es sich lediglich um gecachte Daten aus anderen Systemen handelt, sollte man ein an der Änderungshäufigkeit der zugrunde liegenden Originaldaten sinnvoll bemessenes Timeout setzen.

Trotz strikter Fokussierung auf Features und Anwendungsfälle ist es im Rahmen eines solchen Artikels nicht möglich, auf jedes Detail einzugehen. Einige Features wurden daher absichtlich ausgeklammert. Hier seien nur beispielhaft die probabilistische Datenstruktur HyperLogLog, die Command-Flags, Lua-Skripts, asynchrone Befehle und die Möglichkeit zum Empfang von Keyspace-Events genannt. Ebenfalls nicht weiter eingegangen sind wir auf die Verfügbarkeit von Redis innerhalb Microsoft Azure, was im Übrigen auch von der StackExchange-Bibliothek unterstützt wird. Festzuhalten bleibt aber in jedem Fall, dass Redis ein performantes, gut skalierendes und insgesamt recht pflegearmes System ist, das aus gutem Grund in den Technologie-Stacks vieler Firmen im Webumfeld zu finden ist. ■

- [1] Jens Geyer, *Ausgefeiltes Caching*,
web & mobile developer 1/2014 S. 84
- [2] Jens Geyer, *Logfiles auswerten und visualisieren*,
web & mobile developer 4/2015 S.96
- [3] Redis-Webseite, <http://redis.io>
- [4] MS-OpenTech Redis,
<https://github.com/Microsoft/Redis>
- [5] StackExchange.Redis,
<https://github.com/StackExchange/StackExchange.Redis>
- [6] Our Airports data download, <http://ourairports.com/data>
- [7] Jim Dennis, »What are 5 mistakes to avoid when using Redis?«, <https://www.quora.com/What-are-5-mistakes-to-avoid-when-using-Redis>
- [8] Beispielcode auf GitHub,
<https://github.com/Jens-G/Redis-Samples>
- [9] Kyle Kingsbury, »Call me maybe. Redis redux«, <https://aphyr.com/posts/307-call-me-maybe-redis-redux>



Jens Geyer

entwickelt bei der VSX Vogel Software GmbH Lösungen, die bei Kunden weltweit im Einsatz sind. Sein Fokus liegt auf der effizienten Entwicklung hochwertiger paralleler und verteilter Anwendungen.

<http://jensgeyer.net>

Downloaden, aufschlauhen!



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



Foto: Fotolia / everythingpossible

BUSINESS-INTELLIGENCE-ANWENDUNGEN

Datenanalyse mit BI-Tools

Die besten Werkzeuge zur Verarbeitung und Visualisierung von Unternehmensdaten.

Noch vor wenigen Jahren war die Einführung von Business Intelligence eine unternehmensweite strategische Entscheidung, von langer Hand geplant, sorgfältig vorbereitet und von den Experten der IT-Abteilung umgesetzt. Analysewerkzeuge und Reporting-Tools sollten vor allem zur Konsolidierung und Standardisierung der Unternehmensprozesse beitragen; sie wurden zentral verwaltet und nur ausgewählten Nutzern zur Verfügung gestellt, die meist über Expertenwissen und Programmierkenntnisse verfügten.

Heute sind es die Fachabteilungen, die die Einführung neuer BI-Tools vorantreiben. Oft geschieht dies, ohne strukturierte Prozesse und konzernweite Policies einzuhalten. Die Anwender wollen schnell und interaktiv Geschäftsdaten bewerten können, statt tage- oder gar wochenlang auf vorgefertigte Reports aus der IT-Abteilung zu warten. Diese Entwicklung hat zu neuen Anbietern und einer Fülle von BI-Werkzeugen geführt. Allein 73 Hersteller von BI-Software listet die Webseite Software Advice (www.softwareadvice.com), das Beratungsunternehmen BARC zählt sogar 250 Anbieter,

die 400 Programme im Portfolio haben. Nachfolgend werden die aktuell interessantesten BI-Tools vorgestellt, die ohne lange Implementierungsphase und Einarbeitungszeit schnell zu aussagekräftigen Analyseergebnissen führen.

IBM Watson Analytics

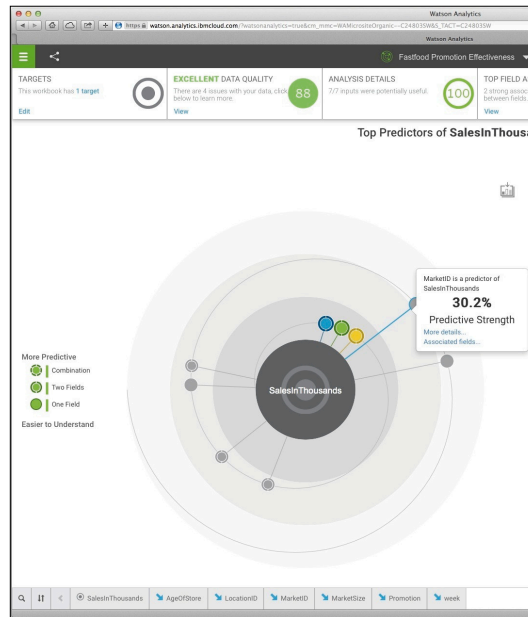
Mit Cognos bietet IBM eine mächtige BI-Plattform, die klassisch über die Unternehmens-IT implementiert und zentral verwaltet wird. Fachanwender sind mit solchen Lösungen zunehmend unzufrieden und wenden sich Tools zu, die sich ohne Beteiligung der IT-Abteilung implementieren und nutzen lassen. IBM spürt diesen Wettbewerbsdruck und steuert der Abwanderung von Kunden mit einem eigenen Self-Service-BI-Produkt gegen, der cloudbasierten Plattform Watson Analytics (Bild 1).

Die Plattform will es dem Anwender so einfach wie möglich machen, relevante Strukturen in den eigenen Daten zu finden. Weder vertiefte Statistikenkenntnisse noch Programmierfähigkeiten oder das Wissen um die richtige Strukturierung

von Abfragen sind nötig. Stattdessen können Fragen in natürlicher Sprache gestellt werden. Watson könnte damit den Data Scientist im Unternehmen wenn nicht überflüssig machen, so doch entlasten.

Wer angesichts der Datenmengen gar nicht weiß, wo er anfangen soll, dem schlägt Watson Analysen vor, die das System aufgrund der Datenstruktur für sinnvoll hält. Für Vorhersagen kann die Plattform geeignete Prädiktoren erkennen und Prognosen erstellen. Wie für die meisten BI-Tools, die auf Fachanwender abzielen, ist auch in Watson Analytics die Visualisierung von Daten eine wichtige Funktion. IBM geht noch einen Schritt weiter und will Zusammenhänge nicht nur anschaulich machen, sondern es dem Nutzer ermöglichen, eine Story zu erzählen. Die Lösung empfiehlt von sich aus, welche grafische Darstellung der Nutzer wählen soll, um seine Geschichte am besten zu illustrieren und seine Argumente am überzeugendsten darstellen zu können.

Watson Analytics ist in einer kostenlosen Variante nutzbar. Diese ist unter anderem auf 500 MByte Speicher, maximal 50 Spalten und 100.000 Datenreihen beschränkt und lässt sich nicht erweitern.



Watson Analytics: In einem übersichtlichen Diagramm zeigt das Programm die Vorhersagestärke verschiedener Parameter und Kombinationen (Bild 1)

Die Edition Personal erlaubt mit einer Million Datenreihen in 256 Spalten die Analyse erheblich größerer Projekte. Der im Paketpreis enthaltene Speicherplatz von 2 GByte lässt sich in 10-GByte-Schritten aufstocken. Mit der Personal-Variante kann man außerdem deutlich mehr Datenquellen anzapfen, so die IBM-Datenbanken DB2, dashDB und SQL Database for Bluemix oder Twitter-Feeds.

Die Professional-Edition schließlich unterstützt 500 Spalten mit 10 Millionen Datenreihen, enthält 100 GByte Speicher, der sich in 50-GByte-Schritten erweitern lässt, und ermöglicht zusätzlich die Integration in Cognos und das Teilen von Daten, Analysen, Vorhersagen und Ansichten mit anderen Anwendern.

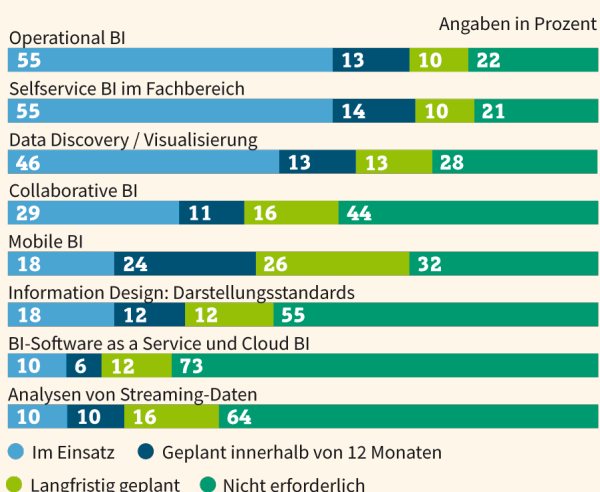
Nach einer Registrierung für die Free-Variante bietet der Startbildschirm in einer an Windows erinnernden Kacheloberfläche direkte Einstiegsmöglichkeiten sowie eine Vielzahl von Tutorials und Beispielvideos. Watson Analytics ist derzeit allerdings nicht lokalisiert, alle Inhalte und Bedienelemente sind auf Englisch. Zudem werden die Tutorial-Filme in einem Fenster abgespielt, das sich über die Bedienoberfläche von Watson Analytics legt, sodass man die gezeigten Beispiele nicht direkt nachverfolgen kann. Die für die Tutorials verwendeten Beispieldateien sind nicht hinterlegt, lassen sich aber schnell per Internetsuche finden.

Bei der Arbeit mit Watson Analytics ist an vielen Stellen erkennbar, dass der Hersteller Wert auf einen möglichst niedrighschwelligsten Einstieg gelegt hat. Die BI-Plattform nimmt den Nutzer an die Hand und bietet zahlreiche Einstiegsmöglichkeiten in Analyse, Prognosenbildung und Visualisierung. So bietet Watson unter *Explore* beispielsweise nach Auswahl des Datensatzes gleich eine Reihe von Fragen an, die man auf Basis dieser Daten sinnvollerweise stellen könnte. Der Nutzer kann auch frei formulierte Fragen eingeben, eine Antwort bekommt man allerdings nur, wenn die Spaltenüberschriften wörtlich in der Frage vorkommen. Versteht Watson die Frage nicht so recht, bietet das Programm sinnverwandte Fragestellungen zur Auswahl an. Ähnlich an die Hand genommen wird man auch in den Bereichen *Predict* (Vorhersagen treffen), *Assemble* (Dashboards und Infografiken erstellen) und *Refine* (Daten verbessern und Muster erkennen).

Microsoft Power BI

Kaum ein Unternehmen dürfte ohne die Tabellenkalkulation Excel auskommen. Seit Langem ist sie das Werkzeug, mit dem vor allem auf Abteilungsebene Daten erfasst und analysiert werden. Die Pivot-Funktion in Excel erlaubt seit jeher auch komplexere Analysen. Mit Erweiterungen wie Power ►

Die Trends bei BI-Tools



BI als Self-Service: Einfach zu bedienende BI-Tools für Fachanwender sind in Unternehmen mittlerweile genauso verbreitet wie große, zentral gemanagte BI-Lösungen (Operational BI).

web & mobile developer 2/2016

Quelle: BARC 2014; rundungsbedingt z. T. nicht 100 Prozent

Pivot, Power Query oder Power View lässt sich das Spreadsheet-Tool noch weiter in Richtung BI aufbohren.

Von Power View, das zur Visualisierung von Daten dient, ist es dann nur noch ein kleiner Schritt zu Power BI, einem cloudbasierten Analysedienst, den Microsoft im vergangenen Jahr ins Leben gerufen hat (Bild 2). Wie Watson Analytics versteht die Plattform in natürlicher Sprache eingegebene Fragen, etwa »Wie hoch war der Umsatz pro Produkt im vergangenen Jahr?« Als Antwort gibt das System eine grafische Auswertung in Form eines Diagramms aus.

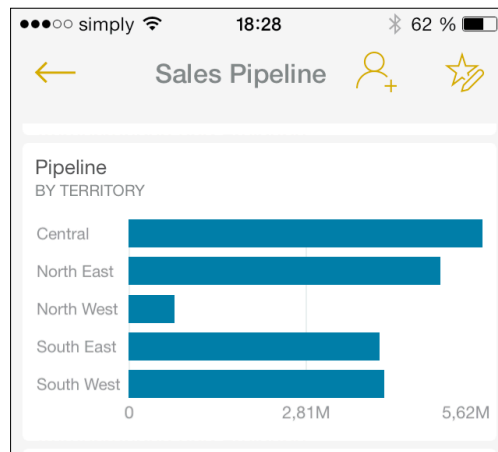
Nach der kostenfreien Registrierung kann der Nutzer aus einer Reihe von Dashboards für die gängigsten Geschäftsanwendungen wählen, darunter Salesforce.com, Google Analytics und Microsoft Dynamics. Daten verschiedenster Quellen wie Excel-Arbeitsblätter, Streaming-Dienste oder lokale Datenbanken lassen sich kombinieren und gemeinsam auswerten. Bei Bedarf werden die Analysen in Echtzeit aktualisiert.

Weitergehende Funktionen sind der kostenpflichtigen Version Power BI Pro vorbehalten, so etwa die gemeinsame Nutzung von Power BI in Office-365-Gruppen, Inhaltspakete für Teams und Einzelpersonen mit vordefinierten Dashboards, Berichten und Datensätzen sowie die Verwaltung der Nutzer über Active Directory. Power-Nutzer dürfen außerdem bis zu 10 GByte an Daten analysieren, in der Free-Version sind es nur 1 GByte.

Neben dem browserbasierten Zugriff bietet Microsoft native Apps für mobile Endgeräte unter Windows, iOS und Android. Power BI lässt sich zudem über ein REST-API in andere Anwendungen integrieren. Für den lokalen Einsatz am PC oder Notebook gibt es Power BI Desktop. Das ebenfalls kostenlose Tool ist als MSI-Paket in 32-Bit- und 64-Bit-Versionen erhältlich und läuft unter Windows 7 und höher sowie unter Windows Server 2008 R2 und neuer. Es bietet deutlich mehr Query- und Modellierungsfunktionen als die Cloud-Variante und lässt sich sowohl mit Power BI als auch mit Power BI Pro nutzen. Auch der Zugriff auf den Pyramid Analytics Server ist damit möglich.

Beim Import von Excel-Dateien zeigt sich Power BI wählerisch. Die Daten darin müssen zwingend im Tabellenformat vorliegen, sonst erhält man die Fehlermeldung »Wir haben in Ihrer Excel-Arbeitsmappe keine Daten gefunden«. Alternativ kann man die Daten auch in eine CSV-Datei exportieren und diese dann in Power BI hochladen.

Die Versuche, durch Fragen in natürlicher Sprache Analyseergebnisse zu erzielen, scheiterten beim Testen häufig. Mit Formulierungen der Art »Wie war der Umsatzverlauf über die vergangenen zwölf Monate?« oder »Wie verteilen sich die Umsätze auf verschiedene Produktkategorien?« konnte das



Microsoft Power BI: Dashboards lassen sich über eine App anzeigen und mit anderen Nutzern teilen (Bild 2)

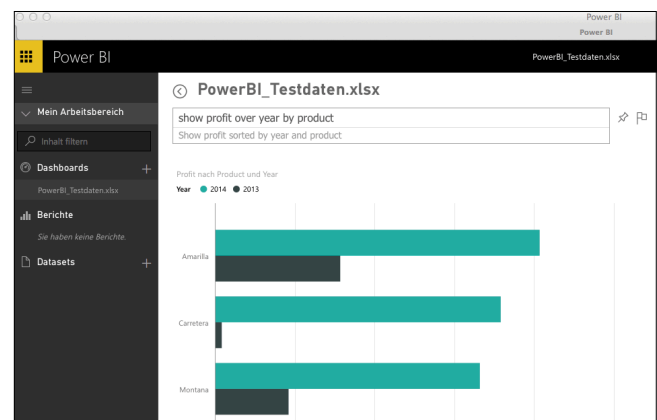
Tool nichts anfangen. Erst wenn man die Spaltenüberschriften aus dem Excel-Import wörtlich in die Fragen einbaut, erzielt man brauchbare Ergebnisse. Fragen wie »Welche products hatten den höchsten profit?« können aber eigentlich nicht mehr als natürliche Sprache bezeichnet werden. Besser klappt es ohnehin, wenn man komplett ins Englische wechselt (Bild 3). Alternativ zur Analyse per Frage bietet Power BI die Möglichkeit, Diagramme per Drag and Drop erstellen zu lassen.

MicroStrategy Analytics

MicroStrategy ist seit über 25 Jahren am Markt. Im Unterschied zu den

meisten großen Wettbewerbern entstand die BI-Plattform nicht durch Zukäufe oder Übernahmen, sondern ist organisch gewachsen. Die Stärke des MicroStrategy-Produkts Analytics liegt in seiner Skalierbarkeit, die es auch großen Unternehmen mit komplexen Anforderungen ermöglicht, zentralisierte BI-Umgebungen mit integrierter Verwaltung und einem Metadaten-Management zu betreiben (Bild 4). Seit 2014 wird die Analyse zudem durch die PRIME-Technologie (Parallel Relational In-Memory Engine) beschleunigt, die dank massiver paralleler Datenverarbeitung laut Hersteller Berechnungen auf bis zu 4,6 TByte Volumen und 200 Milliarden Datenreihen durchführen kann.

MicroStrategy Analytics unterstützt so ziemlich jede Datenquelle, die man sich denken kann, angefangen von textbasierten CSV-Dateien und Excel-Sheets über klassische relationale Datenbanken wie MySQL oder Microsoft SQL Server bis hin zu diversen Hadoop-Distributionen, SAP HANA und Salesforce.com. Auch Social-Media-Informationen und andere Cloud-Daten kann man integrieren. Zur Beschleunigung von Big-Data-Analysen kann Analytics Petabyte-große



Noch nicht polyglott: Microsoft Power BI kann Daten auf Basis von Fragen analysieren. Das funktioniert auf Englisch allerdings deutlich besser als auf Deutsch (Bild 3)

Datenmengen gezielt eindampfen, sodass schnelle In-Memory-Analysen durchgeführt werden können.

Trotz des eher traditionellen, zentralistischen Ansatzes und der Fokussierung auf Großkunden hat der Hersteller die aktuellen Trends nicht verschlafen und bietet den Anwendern in den Fachabteilungen Werkzeuge zur interaktiven Analyse, die als eigenständige Software oder integriert in die Analytics-Plattform verwendet werden können. Ein Cloud-Angebot gibt es ebenso wie die Möglichkeit, mit mobilen Endgeräten auf Daten und Analysen zuzugreifen.

Standalone-Variante

Die Standalone-Variante, MicroStrategy 10 Desktop, lässt sich 30 Tage testen. Die Software ist für Windows-Betriebssysteme ab Windows 7 SP 1 (64 Bit) und Mac OS X ab 10.8 erhältlich. Die integrierte Plattform für Reporting, Dashboards und Analyse spielt ihre Stärken vor allem in großen, komplexen Umgebungen aus. Nach der Installation bietet das Programm verschiedene Einstiegshilfen an, darunter Video-Tutorials und Beispieldateien. Eigene Daten lassen sich mit *Neue Daten hinzufügen...* integrieren. Mit *Daten vorbereiten* kann der Nutzer den Import überprüfen. Wie andere BI-Tools erkennt MicroStrategy 10 automatisch, ob es sich bei den Einträgen um Kategorien, hier Attribute genannt, oder kontinuierliche Werte (Metriken) handelt. Falls bei dieser Einteilung ein Fehler auftritt, lassen sich Attribute auch von Hand in Metriken umwandeln und umgekehrt.

Mit *Umbauen* kann der Anwender zudem direkt die Quelldaten reorganisieren. Die importierten Daten dienen als Basis für Grafiken, die im automatisch erstellten neuen Dashboard platziert werden. Dazu wählt man zunächst eine Darstellungsform. Zur Auswahl stehen unter anderem die etwas seltsam als Wärmekarte übersetzte Heat Map sowie Balken-, Kreis- und Liniendiagramme (Bild 5). Etwas aus dem üblichen Rahmen fallen die Blasendiagramme sowie die Netzwerke.

Anders als etwa in Watson Analytics ist der Nutzer allerdings auf sich selbst gestellt, wenn er die passende Darstellungsform für seine Fragestellung herausfinden will.

Die Grafiken halten weitere Analysemöglichkeiten bereit. So kann man sich etwa mit der Drill-Funktion zusätzliche Details zu einzelnen Wertepaaren anzeigen lassen. Mit einem Klick auf *Visualisierung einfügen* kann der Nutzer weitere Grafiken in sein Dashboard integrieren. Dashboards lassen sich speichern und als PDF oder Grafik im PNG-Format exportieren. Auch einzelne Diagramme kann der Anwender anderen Nutzern per Export zur Verfügung stellen. Ne-



MicroStrategy: In die Mobile-Plattform kann man neben BI-Analysen auch andere Geschäftsprozesse integrieren (Bild 4)

ben dem Speichern als PDF ist in diesem Fall auch die Weitergabe der Daten im CSV-Format möglich.

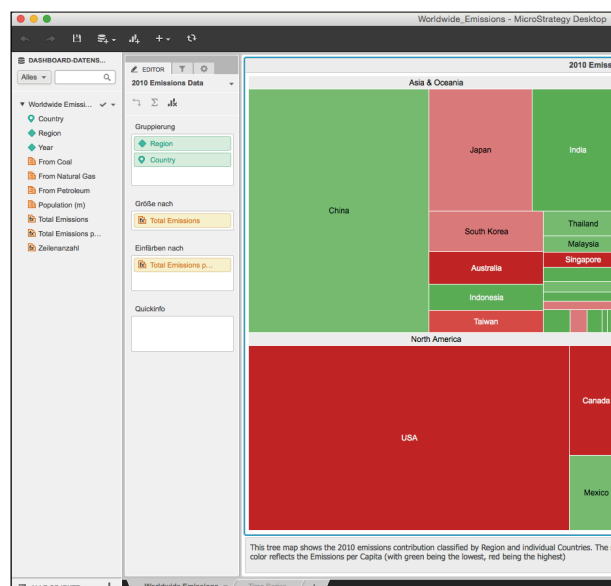
Qlik Sense

Qlik ist vor allem für seine Fähigkeiten beim Aufspüren von Daten und der Aufdeckung von Zusammenhängen bekannt. Das Unternehmen bietet mit QlikView eine zentralisierte BI-Plattform, die es IT-Experten und erfahrenen Nutzern ermöglicht, interaktive Dashboards zu erstellen (Bild 6).

Der Hersteller kann sich dem Trend zu nutzerfreundlicheren Tools für Fachanwender ebenfalls nicht entziehen und hat mit Qlik Sense eine BI-Werkzeugsammlung herausgebracht, die es auch dem weniger versierten Anwender erlaubt, ei-

gene Analysen durchzuführen und Dashboards mit aussagefähigen Grafiken zu erstellen (Bild 7). Wie QlikView wird auch Sense zentral gemanagt – für viele, gerade größere Unternehmen sicher ein wichtiger Vorteil.

Qlik Sense bietet dem Anwender in der Fachabteilung Komfort und Flexibilität, ohne dass die IT-Abteilung die Kontrolle über den BI-Einsatz verlieren würde. Es besteht aus einer ganzen Reihe von Modulen: Qlik Sense Enterprise heißt die eigentliche Plattform, die eine zentrale Verwaltung ermöglicht. Mit Qlik Management Console (QMC) können Administratoren alle Instanzen im Unternehmen verwalten ►



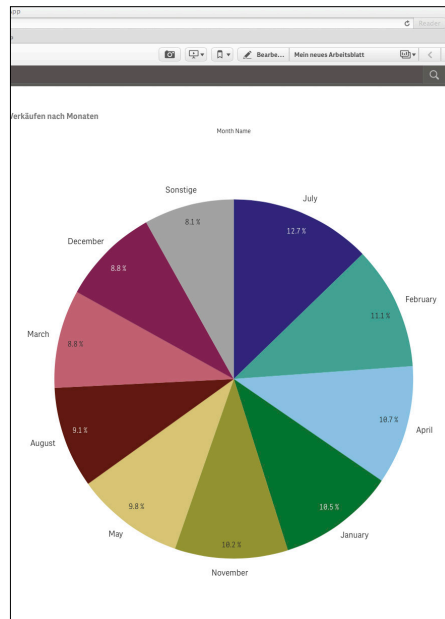
MicroStrategy 10: Analysen lassen sich per Drag and Drop zusammenstellen und auch als »Heat Map« darstellen (Bild 5)

und überwachen. Zugriffe lassen sich regelbasiert steuern, zentrale Bibliotheken sorgen für Datenkonsistenz.

Die Nutzer greifen über den sogenannten Hub per Browser auf den Datenstand zu, die Berechnungen erfolgen im Arbeitsspeicher über die Qlik Sense Engine (QIX), was Analysen erheblich beschleunigt. Die 8 GByte RAM, die Qlik als Mindestanforderung nennt, sind wohl kaum ausreichend, um die Leistungsfähigkeit dieses In-Memory-Systems zu nutzen, hier sollte man nicht sparen. Die anderen Systemvoraussetzungen – Microsoft Windows Server 2008 R2 SP1 und höher, Microsoft Active Directory und Windows Integrate Authentication, .NET Framework ab Version 4.5.1 sowie eine PostgreSQL-Datenbank – sollten sich dagegen leicht realisieren lassen.

Qlik Sense wird über sogenannte Tokens lizenziert und abgerechnet. Ein Token ermöglicht dem User den Zugriff auf Streams und Applikationen einer Qlik Sense Site. Eine Qlik Sense Site kann aus einem oder mehreren Knoten bestehen, die zu einem zentralen Node zusammengefasst werden. Die Tokens können nutzerbasiert als *User Access* oder zeitbasiert als *Login Access* verwendet werden. Der Kunde kann selbst entscheiden, wie er seine Tokens einsetzen will, und kann so die Nutzungsform flexibel an die eigenen Anforderungen anpassen.

Wer Qlik Sense ausprobieren möchte oder keine unternehmensweite Anbindung braucht, kann das kostenlose Qlik Sense Desktop verwenden. Es ist für Einzelanwender gedacht, die per Drag and Drop Grafiken, Berichte und Dashboards erstellen. Die Software ist nur für 64-Bit-Windows-Betriebssysteme ab Windows 7 erhältlich. Nutzer können gespeicherte Analysen über die Qlik Sense Cloud kostenfrei mit bis zu fünf weiteren Anwendern teilen.



Qlik Sense: Das Programm punktet mit umfangreichen Verwaltungsmöglichkeiten und einer schnellen In-Memory-Engine (Bild 6)

Ebenso ist es möglich, neue Anwendungen in der Cloud zu erstellen. Über den Qlik Data Market können Anwender ihre Analysen mit zusätzlichen Informationen aus externen Quellen wie Wetterdaten, demografische Entwicklungen, gesellschaftliche Kennzahlen oder Währungsschwankungen anreichern.

Um in der Qlik Sense Cloud eine App zu erstellen, klickt man im Bereich *Meine persönliche Cloud* auf *Neue App*, gibt dieser einen Namen und wählt eine Datenquelle. Mit *Datenprofil erstellen* lassen sich die Daten auf Probleme untersuchen. Die Funktion schlägt beispielsweise Alarm, wenn zwei Datenquellen Felder mit demselben Namen, aber unterschiedlichen Datenformaten enthalten.

Wenn keine Fehler gemeldet werden, kann man mit *Laden und Abschließen* die Datenanalyse beginnen. Nun wird ein leeres Arbeitsblatt ange-

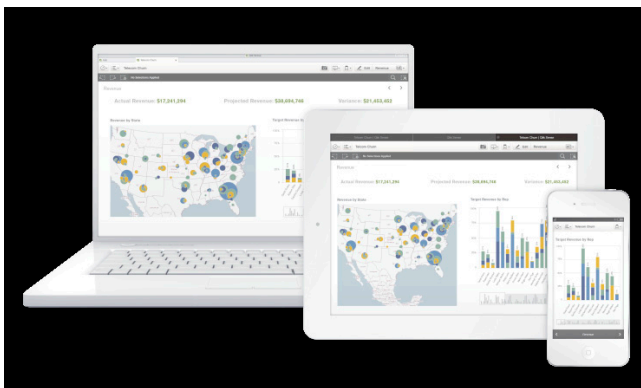
zeigt, in das sich per Drag and Drop Diagramme, Tabellen und Pivot-Tabellen ziehen lassen. Dimensionen und Kennzahlen fügt man aus einer Dropdown-Liste auf der rechten Seite der Arbeitsfläche hinzu.

SAP Lumira

Seit der Übernahme von BusinessObjects im Jahr 2007 hat SAP sein Angebot an Werkzeugen zur Datenanalyse kontinuierlich ausgebaut. Die BI-Plattform BusinessObjects zielt dabei vor allem auf große Unternehmen, die ihre Business-Intelligence-Umgebung zentral zur Verfügung stellen und verwalten wollen. Dank der In-Memory-Datenbanktechnologie HANA des Herstellers sind auch große Datenmengen und umfangreiche Analysen in kurzer Zeit bearbeitbar.

Den Bedarf an BI-Tools, mit denen Fachanwender selbst Analysen durchführen und Schaubilder erstellen können, decken die Walldorfer mit SAP Lumira, das 2013 aus dem Vorgänger Visual Intelligence hervorging. Es ermöglicht dem Nutzer, über eine einfach zu bedienende Oberfläche auf Daten zuzugreifen, diese zu bearbeiten und Zusammenhänge zu visualisieren.

Mit SAP Lumira können Fachanwender ohne Unterstützung der IT Daten analysieren und Grafiken erstellen. Das BI-Tool spielt seine Stärken vor allem in Kombination mit anderen SAP-Werkzeugen aus. Lumira ist als Desktop-Version für 64-Bit-Windows-Systeme ab Windows 7 SP 1 und als Cloud-Angebot verfügbar. Mobil kann man mit einem HTML5-fähigen Browser darauf zugreifen, für iOS gibt es außerdem die App SAP Business Objects Mobile, die auch Lumira-Daten anzeigen kann. Die Personal Edition von Lumira Desktop ist kostenfrei erhältlich. Sie kann als Datenquellen allerdings lediglich Excel-, CSV-Dateien und andere Daten in reinen Textformaten wie TXT, LOG, PRN oder TSV nutzen.



Für mobile Endgeräte: Qlik stellt einen browserbasierten Client zur Verfügung, der HTML5-fähig ist und per Touch-Eingabe bedient werden kann (Bild 7)

Mit der Standard Edition lassen sich dagegen zahlreiche Datenbanken anzapfen, darunter Amazon EMR Hive, Cloudera Impala, OData, IBM DB2, Ingres, Microsoft SQL Server, Netezza, Oracle 10, 11, 12 und Exadata, MySQL 5, PostgreSQL und viele mehr. Besonders gut versteht sich Lumira naturgemäß mit den hauseigenen Datenformaten.

Ob HANA, ERP 6, R/3 Release 4, my SAP ERP 2004 oder SAP MII (Manufacturing and Intelligence) – die Treiber sind bereits integriert und vorkonfiguriert. Ist im Unternehmen ein SAP Business Warehouse ab Version 7.0 im Einsatz, kann Lumira auch damit Daten austauschen.

Ebenfalls kostenlos ist die Cloud-Variante. Sie ermöglicht neben der Visualisierung von Datenzusammenhängen auch das Erstellen von »Storyboards«, wie SAP dies nennt (Bild 8). Das sind aber eigentlich nur Dokumentvorlagen, mit denen man per Drag and Drop leicht Reports zusammenstellen kann. Grafiken und Storyboards lassen sich über die Cloud veröffentlichen und teilen.

Alternativ stehen für die interne Kommunikation drei Server-Varianten zur Verfügung: Lumira Server for Teams ist für kleine Arbeitsgruppen mit maximal zehn gleichzeitig zugreifenden Benutzern gedacht.

Mit Server for BI lassen sich mit Lumira erstellte Dokumente in eine BusinessObjects-BI-Plattform integrieren, was alle Vorteile einer zentralen Verwaltung wie Rechtsmanagement, Zugriffskontrollen oder Versionsverwaltung mit sich bringt. Der Server ist zudem für die Offline-Verarbeitung der

Daten im Arbeitsspeicher zuständig. Das System ist nicht mit HANA identisch, übernimmt aber viele Konzepte daraus, zum Beispiel die In-Memory-Datenverarbeitung oder die Art der Speicherung, Parallelisierung und Komprimierung von Datenreihen.

Sehr große Datenmengen oder Nutzerzahlen lassen sich mit Lumira Server for HANA verwalten beziehungsweise verarbeiten. Dieser ist direkt in HANA integriert und nutzt die In-Memory-Berechnungsfunktionen der Plattform. An ein SAP Business Warehouse (BW) lässt sich Lumira nicht direkt anbinden. Ab Version 1.27 von Lumira gibt es aber einen Konnektor, der es den Nutzern ermöglicht, auf Daten in BW zuzugreifen und diese in Lumira zu bearbeiten.

250
Anbieter haben
400 BI-Programme
im Portfolio

Quelle: BARC

Tableau Desktop

Tableaus Stärke liegt in der Aufdeckung und Visualisierung von Daten. Die Bedienoberfläche ist leicht verständlich und erlaubt es, in wenigen Schritten per Drag and Drop erste Ergebnisse zu generieren. Die leichte Bedienbarkeit hat Tableau vor allem in den Fachabteilungen viele Anhänger beschert, und so kommt die Software oft nicht über einen strukturierten Beschaffungsprozess durch die IT-Abteilung ins Unternehmen, sondern durch die Anwender selbst. Die BI-Tools von Tableau Software lassen sich intuitiv bedienen und bieten umfangreiche Möglichkeiten der Darstellung.

Tableau Desktop setzt zur Datenanalyse nicht auf Assistenten oder Skripts, sondern auf die visuelle Abfragesprache ►

Die Business-Intelligence-Tools im Überblick

Produkt	Beschreibung	Lizenz	Kosten	Internet
IBM Watson Analytics	Nimmt die Nutzer an die Hand; viele Einstiegsmöglichkeiten in Analyse, Prognosen und Visualisierung	pro Benutzer und Monat	Free: kostenlos, Personal: 28,94 Euro, Professional: 77,19 Euro	https://watson.analytics.ibmcloud.com
Microsoft Power BI	Der Cloud-Service Power BI von Microsoft ist in der Einstiegsversion kostenlos	pro Benutzer und Monat; Jahresabonnement erforderlich	Power BI: kostenlos, Power BI Pro: 8,40 Euro, Volumenlizenzen erhältlich	https://powerbi.microsoft.com
MicroStrategy Analytics	Die integrierte Plattform für Reporting, Dashboards und Analyse ist vor allem in großen, komplexen Umgebungen stark	pro Benutzer (Desktop-Version); Testphase: 30 Tage	600 Dollar (circa 535 Euro)	www.microstrategy.com/de/analytics
Qlik Sense	Umfangreiche Verwaltungsmöglichkeiten, schnelle In-Memory-Engine	pro Token (User oder Login Access)	Qlik Sense: Preise auf Anfrage, Qlik Sense Desktop und Cloud: kostenlos	https://global.qlik.com/de/explore/products/sense
SAP Lumira	Gut nutzbar für Fachanwender; stark mit anderen SAP-Anwendungen	pro Benutzer (Desktop-Version); Testphase: 30 Tage	Lumira Desktop Personal Edition: kostenfrei, Standard Edition: 820 Euro	www.sap.com/lumira
Tableau Desktop	Intuitiv bedienbar, viele Darstellungsoptionen	pro Benutzer; Testphase: 14 Tage	Personal Edition: 999 Dollar (circa 898 Euro), Professional Edition: 1999 Dollar (circa 1796 Euro)	www.tableau.com/de-de

VizQL. Sie erlaubt es, Analysen per Drag and Drop zusammenzuklicken, übersetzt diese in Datenbankabfragen und zeigt die Ergebnisse wieder in grafischer Form an (Bild 9). Die Daten können dabei in klassischen SQL-Datenbanken liegen, aus Big-Data-Plattformen wie Hadoop oder Google Big-Query kommen oder anderen Quellen, etwa Google Analytics, entstammen.

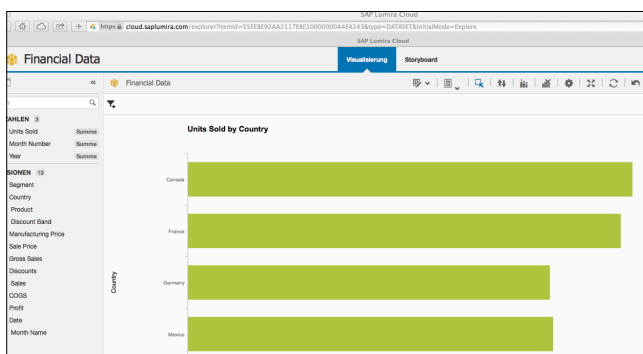
Um in Tableau Desktop eine Analyse zu starten, muss der Anwender eine Verbindung zu den Datenquellen herstellen oder eine bereits gespeicherte Arbeitsmappe öffnen. Wird eine neue Datenquelle ausgewählt, muss der Nutzer zunächst festlegen, welche Arbeitsblätter oder Tabellen er analysieren möchte.

Dazu zieht er diese einfach aus der Seitenleiste in das Bearbeitungsfenster. Weitere Arbeitsmappen oder Tabellen lassen sich per Drag and Drop oder Doppelklick hinzufügen. Tableau Desktop erstellt dann automatisch eine Standardverknüpfung, die sich per Klick auf das Verbindungssymbol ändern lässt.

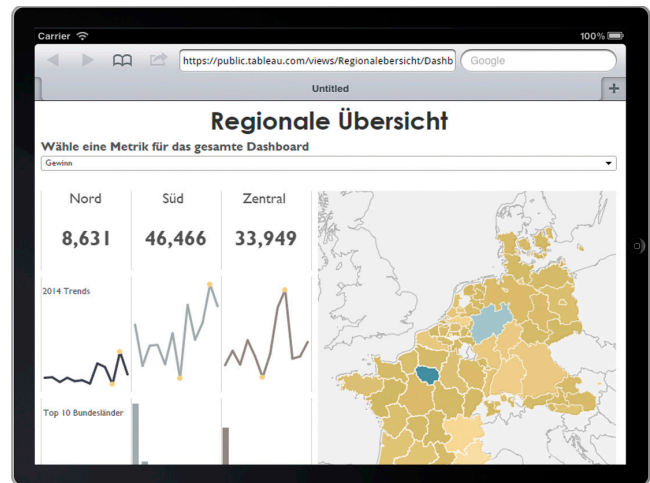
Je nach Datenquelle kann der Anwender wählen, ob die Verbindung zu den Ausgangsdaten laufend live aktualisiert werden soll oder ob er diese in die Tableau-Daten-Engine importieren und dort analysieren möchte. Letzteres beschleunigt die Verarbeitung und entlastet die Datenbanken von Abfragen, ist aber eben nur dann sinnvoll, wenn die zu analysierenden Daten nicht ständig aktualisiert werden müssen.

Tableau Desktop teilt die Daten in Dimensionen und Kennzahlen ein. Dimensionen sind typischerweise Kategorien mit diskreten Ausprägungen wie Kundenname, Ort oder Region. Kennzahlen bestehen üblicherweise aus kontinuierlichen Werten wie Umsatz oder Menge.

Um eine Analyse zu starten, zieht man Dimensionen und Kennzahlen aus den Ursprungsdaten in die Felder *Spalten* und *Zeilen*. Sofort werden die Ergebnisse als Grafik angezeigt. Will man die Ausprägungen einer Dimension farblich unterschiedlich darstellen, zieht man diese einfach auf das Feld *Farbe*. So ist mit wenigen Mausbewegungen eine erste Analyse erstellt. Dieses Drag-and-Drop-Prinzip zieht sich durch die komplette Bedienlogik von Tableau Desktop. Auch komplexe Analysen und Berechnungen lassen sich entweder durch Ziehen und Ablegen durchführen oder indem man aus einem Dropdown-Menü den entsprechenden Eintrag wählt.



SAP Lumira: In der kostenlosen Cloud-Version kann man neben Analysen auch Storyboards (Dokumentvorlagen) erstellen (Bild 8)



App Tableau Mobile: Damit lassen sich Dashboards von einem Tableau-Server abrufen und auf einem Android-Tablet oder iPad präsentieren (Bild 9)

Tableau Desktop läuft unter Windows ab Vista sowie auf Microsoft Server 2008 R2 oder neuer. Auch für Mac OS X ab 10.8.1 gibt es einen Client.

Das Programm ist in zwei Varianten erhältlich, die sich in der Unterstützung von Datenquellen unterscheiden. Die Personal Edition kann nur Daten aus Microsoft Access 2003 oder höher, Excel 2007 oder höher, der Microsoft-Cloud Azure, Open Data Protocol (OData), eigenen Tableau-Datenextraktionen und CSV-Dateien extrahieren.

Die Professional Edition unterstützt dagegen mehr als 40 Datenquellen, darunter Hadoop-Distributionen, Google Analytics und BigQuery, IBM DB2 und Netezza sowie Oracle- und SAP-Produkte.

Um Ergebnisse aus Tableau Desktop zu veröffentlichen und per Browser oder mobilem Endgerät zugänglich zu machen, benötigt der Anwender die Server-Komponente des Herstellers. Diese steht auch als Software as a Service in einer Cloud-Variante zur Verfügung.

Für Desktop-, Server- und Online-Variante gibt es kostenlose Testversionen, die sich 14 Tage nutzen lassen. Unbegrenzt kostenfrei ist Tableau Public (<http://public.tableau.com>), ein Online-Dienst, über den sich Daten kombinieren, die Zusammenhänge visualisieren und die Ergebnisse direkt veröffentlichen lassen.

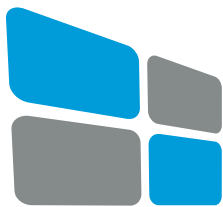
Ebenfalls gratis ist der Tableau Reader, mit dem man Analysen anzeigen, filtern, sortieren und drucken kann. ■



Dr. Thomas Hafen

ist seit mehr als 15 Jahren als Redakteur und Journalist tätig, unter anderem für die IT-Fachzeitschriften NetworkWorld Germany und ChannelPartner sowie die Fotoseiten Seen.by und Digitalkamera.de.

<http://thomas-hafen.de>



Internet World

Die E-Commerce Messe

01.-02. März 2016, München



Das Event des E-Commerce 2016

- 360 Aussteller
- 150 Vorträge
- 16.000 Besucher

internetworld-messe.de

Wir sind
umgezogen:

**Halle
B5 + B6**

Eingang Ost

**Besuch der
Messe kostenlos.**

**Jetzt
registrieren!**

FACEBOOK API MIT PHP ANSPRECHEN

Facebook & PHP

Facebook bietet für PHP ein leistungsfähiges SDK an, über das man zahlreiche Facebook-Features direkt ansprechen kann.

Es ist relativ einfach, mit dem Facebook API direkt zu kommunizieren. Neben SDKs für JavaScript und viele anderen Sprachen gibt es natürlich auch eine Version für PHP.

Das Facebook SDK für PHP in der Version 5.0.0 unterstützt vollständig PSR-4:Autoloader (www.php-fig.org/psr/psr-4), was insbesondere bedeutet, dass es sich stark auf Namespaces verlässt, damit Klassendateien automatisch geladen werden können. Wer mit den Konzepten Namespaces und Autoloading noch nicht vollständig vertraut ist, kann sich unter dem obigen Link eine Auffrischung abholen.

Facebook Graph API

Das Facebook Graph API ist ein sehr einfaches, HTTP-basiertes API. Es wurde nach dem Konzept des sogenannten Social Graph benannt.

Der Social Graph ist eine Repräsentation der Informationen, die in Facebook existieren, sowie ihrer Struktur. Dabei besteht der Graph aus den folgenden Elementen:

- Nodes (Knoten): Als Knoten werden alle direkt erreichbaren Ziele im Graph bezeichnet, zum Beispiel ein User, ein Foto, eine Seite oder ein Kommentar.
- Edges (Kanten): Die Verbindung zwischen allen Dingen (zum Beispiel der Besitz eines Albums, oder das Ansehen

eines Videos) beziehungsweise die Aktionen zwischen Nodes werden als Edges bezeichnet (zum Beispiel die Fotos eines Albums oder ein Like).

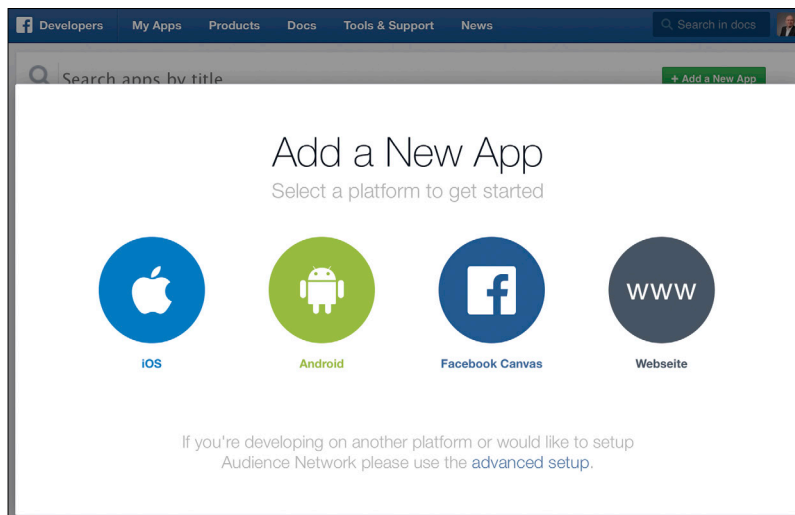
- Fields (Felder): Die Eigenschaften eines Nodes werden als Fields bezeichnet, zum Beispiel die E-Mail-Adresse eines Users.

Im Zentrum des Social Graphs (der eine Abbildung aus Nodes, Edges und Fields ist), steht der User (Node). Der User ist mit anderen Users befreundet (Edges). Eventuell besitzt (Edge) der User ein Album (Node), das gegebenenfalls mehrere Bilder (Nodes) beinhaltet (Edges). Diese haben vielleicht Likes (Edges) erhalten.

Berechtigungen erforderlich

Die richtigen Berechtigungen vorausgesetzt, können die Beziehungen und Daten über das Facebook Graph API abgerufen werden. Da das API auf HTTP basiert, kann die Ansprache mit jeder beliebigen Programmiersprache erfolgen. Wir verwenden für den vorliegenden Artikel PHP und dafür ein speziell von Facebook entwickeltes PHP SDK.

Da das API auf HTTP beruht, könnten wir auch eine Abfrage über einen URL durchführen: <https://graph.facebook.com/me>. Dies liest die Eigenschaften des eigenen Users aus. Allerdings ist dies ohne einen sogenannten Access Token nicht möglich, sodass wir hier zunächst eine Fehlermeldung erhalten. Wichtig ist aber der Teil `/me` des URL. Dieser wird



Auswahl des App-Typs (Bild 1)

Tabelle 1: API-Endpoints

API-Endpoint	Bedeutung
<code>/me</code>	Der aktuelle User. Man kann hier auch stattdessen einen Usernamen angeben, um auf einen anderen User zuzugreifen
<code>/me/feed</code>	Der Feed des aktuellen Users
<code>/me/album</code>	Ein Fotoalbum
<code>/me/comment</code>	Ein einzelner Kommentar auf einem Node
<code>/me/message</code>	Eine einzelne Nachricht
<code>/me/photo</code>	Ein einzelnes Foto
<code>/user</code>	Ein spezieller User
<code>/user/accounts</code>	Alle Accounts, die ein User verwaltet

als API-Endpoint bezeichnet. Davon gibt es in dem API eine größere Anzahl. Die **Tabelle 1** zeigt einige Beispiele.

Die Systemvoraussetzungen für das SDK sind recht gering: PHP 5.4 oder höher, die PHP-Extension mbstring und Composer (optional). Wir starten in diesem Tutorial mit einem leeren Webprojekt. Es ist aber natürlich auch jederzeit möglich, das SDK zu einem bestehenden Projekt hinzuzufügen.

Im Folgenden verwenden wir die Installation via Composer. Sollten Sie allerdings keinen Shell-Zugriff haben (zum Beispiel bei Shared-Hosting-Umgebungen), so können Sie das SDK natürlich auch manuell installieren:

<https://developers.facebook.com/docs/php/gettingstarted>

Legen Sie im Document-Root eine Datei mit dem Namen *composer.json* und folgendem Inhalt an:

```
{
  "require" : {
    "facebook/php-sdk-v4" : "~5.0"
  }
}
```

Nun müssen Sie Composer installieren, wenn Sie es noch nicht auf Ihrem System haben. Unter Linux und Mac OS X reicht hierfür Folgendes aus:

```
$ curl -sS https://getcomposer.org/installer | php
$ sudo mv composer.phar /usr/local/bin/composer
```

Die zweite Zeile kopiert Composer ins ausführbare Verzeichnis um, damit die Eingabe von *composer* ausreicht, um diesen auszuführen.

Wenn Sie keine Root-Berechtigung haben, können Sie Composer mit *./composer.phar* ausführen.

Installation des SDK

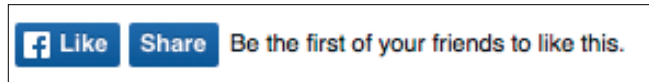
Entsprechende Installationsanweisungen für Windows finden Sie unter <https://getcomposer.org/doc/00-intro.md>. Nun können Sie die Installation des SDK ganz einfach starten:

```
$ composer install
Loading composer repositories with package information
Installing dependencies (including require-dev)
- Installing facebook/php-sdk-v4 (5.1.2)
  Downloading: 100%
```

```
facebook/php-sdk-v4 suggests installing guzzlehttp/guzzle
(Allows for implementation of the Guzzle HTTP client)
Writing lock file
Generating autoload files
```

Im PHP-Skript genügt jetzt die Angabe des Autoloaders:

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
...
```



Ausgabe eines Like- und Share-Buttons (Bild 2)

Sollten Sie Composer nicht verwenden wollen, finden Sie unter dem URL <https://developers.facebook.com/docs/php/gettingstarted/5.0.0> eine Möglichkeit, das SDK über eine Download-Datei zu installieren.

Facebook-App anlegen

Um das SDK zu verwenden, benötigen Sie eine Facebook-App, die Sie unter dem URL <https://developers.facebook.com/apps> erstellen können. Klicken Sie zunächst rechts oben auf den grünen Button + *Add a New App*. Es erscheint ein Overlay mit vier möglichen Plattformen (unter dem Link *advanced setup* stehen sogar noch weitere zur Verfügung):

- iOS (Integration in iOS),
- Android (Integration in Android),
- Facebook Canvas (Integration in Facebook),
- Website (separate Website mit Facebook-Integration).

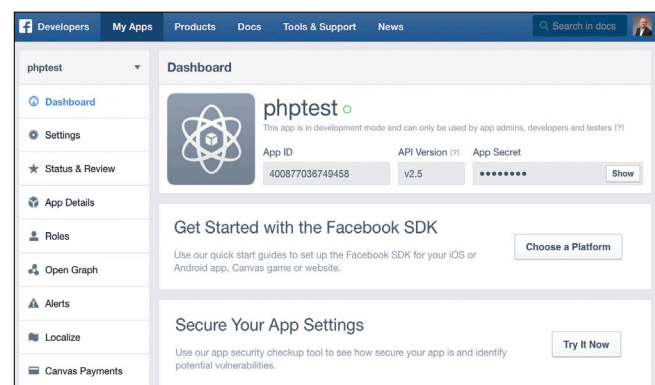
Wir entscheiden uns für den letzten Punkt *Website*, geben im darauffolgenden Screen den Namen *phptest* ein und klicken dann auf *Create New Facebook App ID*. Nun kann eine Kategorie gewählt werden. Wir nehmen hier *Apps für Seiten* und klicken auf *App-ID erstellen* (Bild 1).

Wichtig ist nur der folgende Code (und dort vor allem die App-ID {*app-id*}), denn darin sind die Daten enthalten, die wir gleich zur Einbindung brauchen. Speichern Sie diesen Code in einer Datei *index.html* ab, damit wir die Funktion der App testen können (Listing 1).

Als Nächstes muss der URL der (späteren) Website festgelegt werden. Dies ist insbesondere auch für das Testen notwendig.

Dann kopieren wir noch den folgenden Code zur Datei *index.html* hinzu und testen die Seite im Browser:

```
<div
  class="fb-like"
  data-share="true"
```



Einstellungen der Test-App (Bild 3)

```
data-width="450"
data-show-faces="true">
</div>
```

Erscheinen nun *Like*- und *Share*-Buttons, war die Einrichtung der App erfolgreich (Bild 2).

Sobald Sie wieder zum URL <https://developers.facebook.com/apps> gehen, sehen Sie die erstellte App in der Auflistung und können sie durch Klick auf das Icon konfigurieren. Wir benötigen hier noch das sogenannte App Secret, das Sie in der Konfiguration unter *App Secret* und *Show* erhalten (Bild 3).

Nun erweitern wir die Einbindung per Autoloader in der Datei *index.php* mit der Konfiguration. Hier sind vor allem die App-ID und das App Secret einzutragen. Der dritte Parameter ist die Open-Graph-API-Version, die verwendet werden soll (hier 2.5, die seit Oktober 2015 im Einsatz ist):

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
```

```
session_start();
```

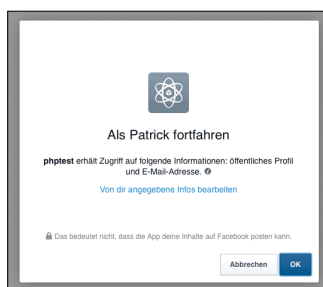
```
$fb = new Facebook\Facebook([
    'app_id' => {app-id},
    'app_secret' => '{app-secret}',
    'default_graph_version' => 'v2.5',
]);
```

Als erstes Beispiel wollen wir nun einen User bei Facebook autorisieren und anschließend seine Profildaten auslesen. Hierfür verwenden wir den sogenannten *getRedirectLoginHelper*, der eine Autorisierung über OAuth 2.0 ermöglicht (Listing 2).

Link zu Facebook

Zuerst wird über die Methode *getLoginUrl(callback-url, permissions)* am Ende des Listings ein Link zu Facebook generiert und diesem der URL mitgegeben, zu dem gesprungen werden soll, wenn das Login erfolgreich war.

Bei Klick auf den Link wird der User zu Facebook weitergeleitet, wo ein modaler Dialog den User auffordert, sich einzuloggen. Die Daten, die die App anfordert, werden in



Genehmigungsdialo für Informationen (Bild 4)

Listing 1: index.html – Basis

```
<script>
window.fbAsyncInit = function() {
    FB.init({
        appId      : '{app-id}',
        xfbml      : true,
        version     : 'v2.5'
    });
};

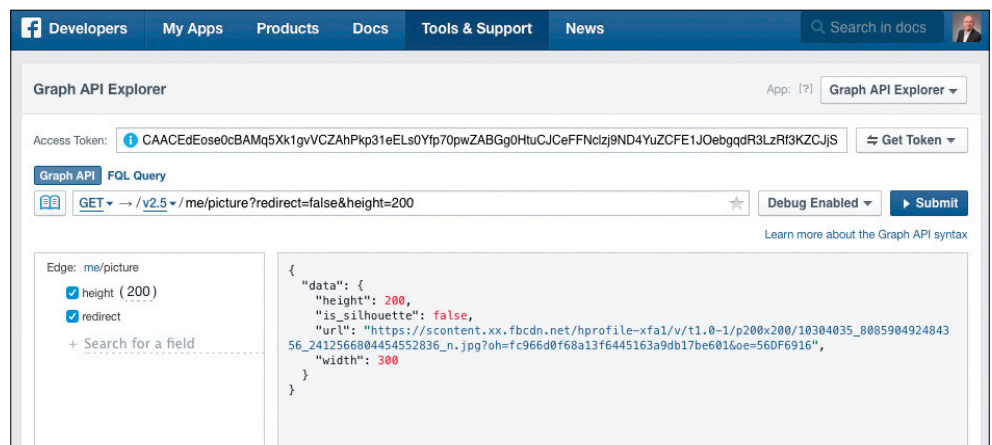
(function(d, s, id){
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) {return;}
    js = d.createElement(s); js.id = id;
    js.src = "//connect.facebook.net/en_US/sdk.js";
    fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));
</script>
```

diesem Dialog ebenfalls angezeigt (und sind vorher über die *permissions* festgelegt worden).

Wenn der User sich erfolgreich eingeloggt hat oder aber die Autorisierung aktiv ablehnt, wird er zu dem in *callback-url* hinterlegten URL zurückgeleitet (Bild 4).

Der Access-Token wird dabei in einer Session gespeichert und anfangs über die Methode *setDefaultAccessToken()* ermittelt. Anschließend wird der kurzfristige Token über den OAuth-Client *getOAuth2Client()* in einen langfristigen verwandelt.

Das Auslesen der Profil-Informationen übernimmt die API-Funktion *get()*, die mit dem Endpoint */me* angefragt wird, dem als Parameter diejenigen Felder übermittelt werden, die man erhalten will – hier Name, Vorname, Nachname und E-Mail. Testen kann man die Graph-Abfrage auch im sogenannten Graph API Explorer unter: <https://developers.facebook.com/tools/explorer> (Bild 5). Weitere Informationen, die man auslesen kann, zeigt Tabelle 2.



Graph API Explorer: Damit kann man die Graph-Abfrage testen (Bild 5)

Eine komplette Übersicht darüber finden Sie im Web unter <https://developers.facebook.com/docs/graph-api/reference/v2.5/user>.

Das User-Profil wird im Code-Beispiel lediglich mit `print_r()` ausgegeben, kann aber dann natürlich programmatisch weiterverwendet werden. Die Fehlerbehandlung ist zudem so aufgebaut, dass sowohl SDK-Fehler (*FacebookSDKException*) als auch Facebook-Graph-Fehler (*FacebookResponseException*) mittels `getMessage()` ermittelt und ausgegeben werden. Das ist Grund genug für die Entwickler, diesen Erfolg zu festigen und das schon etwas betagte Framework einer Generalüberholung zu unterziehen.

Als Nächstes wollen wir das Profilfoto des Users auslesen – dafür duplizieren wir die Datei und nennen das Duplikat *fb-*

photo.php (Listing 3). Hier wird über den Endpoint `/me/picture` das Bild ermittelt und über den Parameter *height* mit einer bestimmten Höhe als Link zurückgeliefert. Das Bild wird im unteren Teil anschließend ausgegeben und auf dem Server abgespeichert.

Timeline des Useres auslesen

Wir wollen nun die Timeline des Users auslesen. Auch hierfür legen wir ein Duplikat der vorherigen Datei an und nennen diese *fb-timeline.php*. Folgende Veränderung nehmen wir darin vor:

```
$permissions = ['user_posts'];
$posts_request = $fb->get('/me/posts?limit=50');
```

Listing 2: fb-login.php

```
<?php
session_start();
require_once __DIR__ . '/vendor/autoload.php';

$fb = new Facebook\Facebook([
    'app_id' => '400877036749458',
    'app_secret' => '8a414093d62cadae6b515c38b7702263',
    'default_graph_version' => 'v2.4',
]);

$helper = $fb->getRedirectLoginHelper();
$permissions = ['email'];
try {
    if (isset($_SESSION['facebook_access_token'])) {
        $accessToken =
            $_SESSION['facebook_access_token'];
    } else {
        $accessToken = $helper->getAccessToken();
    }
} catch (Facebook\Exceptions\
FacebookResponseException $e) {
    {
        echo 'Graph meldet einen Fehler: ' .
            $e->getMessage();
        exit;
    } catch (Facebook\Exceptions\FacebookSDKException $e) {
        echo 'Facebook SDK meldet einen Fehler: ' .
            $e->getMessage();
        exit;
    }
}

if (isset($accessToken)) {
    if (isset($_SESSION['facebook_access_token'])) {
        $fb->setDefaultAccessToken($_SESSION
            ['facebook_access_token']);
    } else {
        $_SESSION['facebook_access_token'] = (string)
            $accessToken;
    }

    $oAuth2Client = $fb->getOAuth2Client();
    $longLivedAccessToken =
        $oAuth2Client->getLongLivedAccessToken
            ($_SESSION['facebook_access_token']);
    $_SESSION['facebook_access_token'] =
        (string) $longLivedAccessToken;
    $fb->setDefaultAccessToken
        ($_SESSION['facebook_access_token']);
}

try {
    $profile_request = $fb->get
        ('/me?fields=name,first_name,last_name,email');
    $profile = $profile_request->getGraphNode()-
        >toArray();
} catch (Facebook\Exceptions\
FacebookResponseException $e) {
    echo 'Graph meldet einen Fehler: ' .
        $e->getMessage();
    session_destroy();
    header("Location: ./fb-login.php");
    exit;
} catch (Facebook\Exceptions\
FacebookSDKException $e) {
    echo 'Facebook SDK meldet einen Fehler: ' .
        $e->getMessage();
    exit;
}

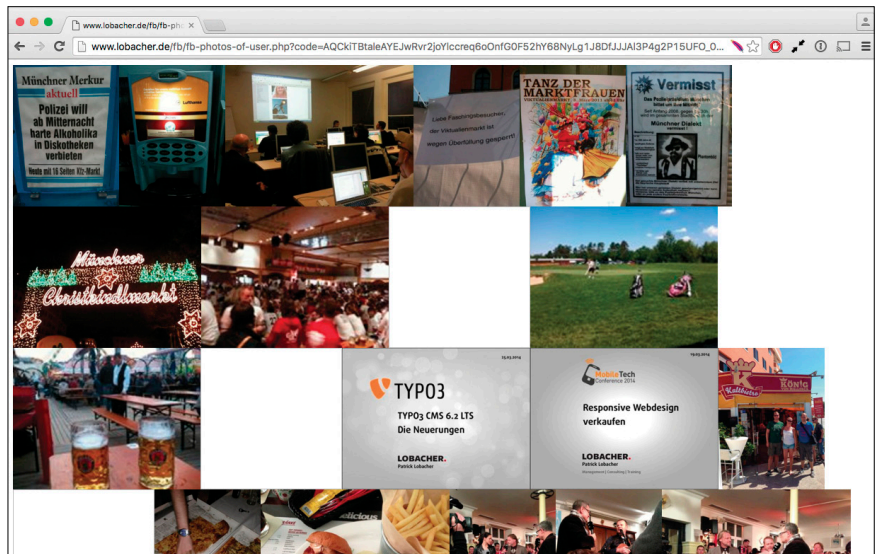
print_r($profile);

$_SESSION['facebook_access_token']
} else {
    $loginUrl = $helper->getLoginUrl
        ('http://www.lobacher.de/fb/fb-login.php',
        $permissions);
    echo '<a href="' . $loginUrl . '">Log in mit
        Facebook!</a>';
}
```

```
print_r($posts_request->
getGraphEdge()->toArray());
```

Um die Timeline auszulesen, ist es zunächst wichtig, dass diese auch in den Berechtigungen enthalten ist. Dafür muss der User aber die Berechtigungen erweitern, was nur geht, wenn er sich erneut einloggt.

Um dies zu forcieren, können Sie ein `session_destroy()` am Anfang des Skripts positionieren, dieses erneut aufrufen und anschließend wieder entfernen. Beim wiederholten Aufrufen werden nun die Berechtigungen erneut aufgerufen. Verantwortlich dafür ist die Zeile `$permissions` im Skript.



Ausgabe der Bilder des eingeloggten Users (Bild 6)

Posts ausgeben

Nun werden über den Endpoint `/me/posts` die Posts (hier limitiert auf 50) aufgerufen und weiter unten ausgegeben, indem die Methode `getGraphEdge()` auf die Ausgabe angewendet und dann das Resultat in ein Array umgewandelt wird. Die Fotos eines Users sind im Endpoint `/me/photos` gespeichert (Bild 6). Dort allerdings befinden sich nur die IDs der Fotos, die man nun gezielt per `<id>?fields=images` auslesen muss. Hier erhält

man nur alle URLs auf die Fotos samt zugehöriger Meta-Daten (Listing 4). Die Freundesliste lässt sich über den Endpoint `/me/taggable_friends` auslesen (Listing 5).

Bisher haben wir lediglich lesend auf die Facebook-Graph-Daten zugegriffen. Nun wollen wir auch schreibend aktiv werden. Sobald wir hier die Berechtigungen um `publish_actions` ergänzen, wird bei der Freigabe auch dediziert danach

Tabelle 2: Graph-Abfrage

Parameter	Abfrageergebnis
<code>age_range</code>	Minimales und/oder maximales Alter
<code>birthday</code>	Geburtsstag
<code>devices</code>	Geräte, die der User verwendet – nur iOS und Android
<code>currency</code>	Währungseinstellung des Users inklusive Umrechnungskurs
<code>education</code>	Ausbildungsinformationen
<code>favorite_athletes</code>	Favorisierte Sportler
<code>favorite_teams</code>	Favorisierte Sportteams
<code>gender</code>	Geschlecht
<code>hometown</code>	Heimatstadt
<code>inspirational_people</code>	Inspirierende Personen
<code>link</code>	Link zur Timeline
<code>location</code>	Aktuelle eingegebene Position
<code>political</code>	Politische Interessen
<code>religion</code>	Religiöse Interessen
<code>sports</code>	Sportliche Interessen
<code>quotes</code>	Favorisierte Quotes
<code>website</code>	Website

Listing 3: fb-photo.php

```
<?php
session_start();
require_once __DIR__ . '/vendor/autoload.php';

$fb = new Facebook\Facebook([...]);
...
if (isset($accessToken)) {
    try {
        $requestPicture = $fb->get(
            ('/me/picture?redirect=false&height=300');
        // Basis Infos ermitteln
        $requestProfile = $fb->get('/me');
        $picture = $requestPicture->getGraphUser();
        $profile = $requestProfile->getGraphUser();
    } catch (Facebook\Exceptions\
        FacebookResponseException $e) {
        ...
    }

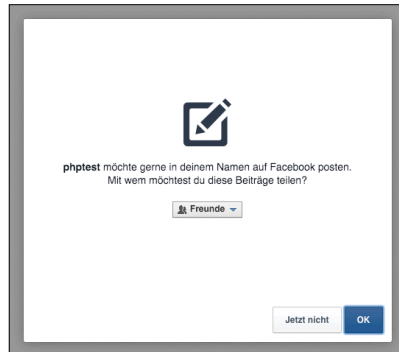
    echo "<img src='".$picture['url']."'>";
    // Bild auf dem Server speichern
    $img = __DIR__ . '/' . $profile['id'] . '.jpg';
    file_put_contents($img,
        file_get_contents($picture['url']));
    ...
}
```


gefragt, ob man will, dass die App auch im eigenen Namen auf Facebook posten darf (Bild 7). Dafür posten wir eine Testnachricht im Feed des Users. Das nachfolgende Listing zeigt die Umsetzung:

```
$permissions = ['email',
'publish_actions'];
...
$data = ['message' =>
'Test-Post für web & mobile
DEVELOPER Artikel...'];

$request = $fb->post('/me/feed',
$data);
$response = $request->getGraphNode();
...
print_r($response);
...
```

In diesem Beispiel verwenden wir als Methode auf dem Facebook-Objekt `post()`, welche es ermöglicht, die gewünschten



Posten: Dieser Genehmigungsdialog zum Posten erscheint vorab (Bild 7)

Daten zu schreiben. Der Endpoint für den Feed lautet `/me/feed` und als Nachricht muss ein Array angegeben werden, welches den Nachrichtentext, der gepostet werden soll, im Schlüssel `message` enthält (Bild 8).

Mit dem Wissen des vorherigen Beispiels können wir auch einen Link posten. Hier muss lediglich ein Link im Data-Array zusätzlich angegeben werden. Bild 9 zeigt das Ergebnis:

```
$permissions = ['email',
'publish_actions'];
...
$data = [
'message' => 'Test-Post mit Link für web & mobile
DEVELOPER Artikel...',
'link' => 'http://www.webundmobile.de'
];
$request = $fb->post('/me/feed', $data);
$response = $request->getGraphNode();
...
print_r($response);
...
```

Zusätzlich kann man im Data-Array noch einen Schlüssel `description` angeben, mit dem der Link zur Information des Benutzers umschrieben wird (URL ist hier *default*). ▶

Listing 4: fb-photos-of-user.php

```
$permissions = ['user_photos'];

$photos_request = $fb->get('/me/photos?
limit=20&type=uploaded');
$photos = $photos_request->getGraphNode();
...
all_photos = array();
if ($fb->next($photos)) {
    $photos_array = $photos->asArray();
    $all_photos = array_merge($photos_array,
    $all_photos);
    while ($photos = $fb->next($photos)) {
        $photos_array = $photos->asArray();
        $all_photos = array_merge($photos_array,
        $all_photos);
    }
} else {
    $photos_array = $photos->asArray();
    $all_photos = array_merge($photos_array,
    $all_photos);
}
foreach ($all_photos as $key) {
    $photo_request = $fb->get('/'.$key['id'].
    '?fields=images');
    $photo = $photo_request->getGraphNode()->
    asArray();
    echo '';
}
...
```

Listing 5: fb-friends-of-user.php

```
...
$permissions = ['user_friends'];
...
$requestFriends = $fb->get('/me/
taggable_friends?fields=name');
$friends = $requestFriends->getGraphNode();
...
if ($fb->next($friends)) {
    $allFriends = array();
    $friendsArray = $friends->asArray();
    $allFriends = array_merge($friendsArray,
    $allFriends);
    while ($friends = $fb->next($friends)) {
        $friendsArray = $friends->asArray();
        $allFriends = array_merge($friendsArray,
        $allFriends);
    }
} else {
    $allFriends = $friends->asArray();
    $totalFriends = count($allFriends);
}
print_r($allFriends);
...
```



Text-Post in der Timeline (Bild 8)

Nun wollen wir ein Bild in die Timeline posten. Hier wird als Quelle das Bild lokal in der Eigenschaft *source* referenziert und per Post-Request an den Endpoint */me/photos* geschickt. Zusätzlich wird über *message* noch ein Text mitgegeben (Bild 10):

```
$permissions = ['email', 'publish_actions'];
$data = [
    'source' => $fb->fileToUpload(__DIR__ .
        '/rocky.jpg'),
    'message' => 'Rocky ist der Beste!'
];

$request = $fb->post('/me/photos', $data);
$response = $request->getGraphNode();
print_r($response);
```

Als vorletztes Beispiel wollen wir nun auf einer Seite posten, die der User verwaltet (Bild 11). Dazu muss der Name der verwalteten Seite (hier zum Beispiel Pluswerk AG) bekannt sein und entsprechend abgefragt werden. Die Liste der Seiten wiederum erhält man über den Endpoint */me/accounts* (Listing 6).

Listing 6: fb-post-page.php

```
$permissions = ['manage_pages', 'publish_pages'];
$profile_request = $fb->get('/me');
$profile = $profile_request->getGraphNode()->
    asArray();
$pages = $fb->get('/me/accounts');
$pages = $pages->getGraphEdge()->asArray();
foreach ($pages as $key) {
    if ($key['name'] == 'Pluswerk AG') {
        $post = $fb->post(
            '/' . $key['id'] . '/feed',
            array('message' => 'Testnachricht für
                web & mobile DEVELOPER'),
            $key['access_token']);
        $post = $post->getGraphNode()->asArray();
        print_r($post);
    }
}
```



Link-Post in der Timeline (Bild 9)

Canvas-App

Um Canvas-Apps zu verwenden (also in Facebook als *iframe* integrierte Apps), muss der Server SSL verstehen, da man nur sichere URLs mittels HTTPS referenzieren darf.

Legen Sie nun eine neue App unter <https://developers.facebook.com> an, indem Sie bei *My Apps* auf *Add a New App* klicken und dann *Facebook Canvas* wählen. Als Namen für die App verwenden Sie *phpcanvastest*.

Wählen Sie anschließend die Option *Skip Basic Setup* und wechseln Sie direkt in die Konfiguration der App. Hierzu verwenden Sie die Option *Settings*. Beim Eintrag *App Domains* tragen Sie Ihre Domain ein und wählen unter Plattform *Facebook Canvas* aus. Als Namespace tragen Sie ebenfalls *phpcanvastest* ein.

Bei den Einstellungen zu *Secure Canvas URL* und *Mobile Site URL* tragen Sie das Verzeichnis auf Ihrem Server ein, das

Listing 7: fb-canvas.php

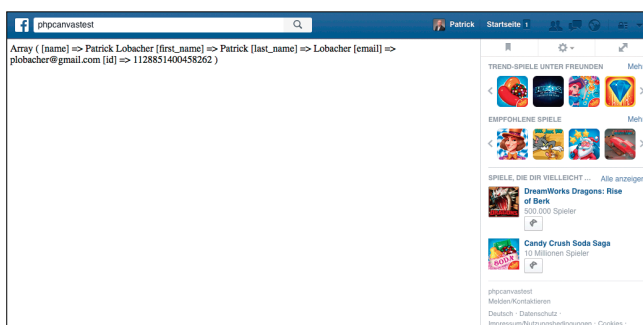
```
<?php
session_start();
require_once __DIR__ . '/vendor/autoload.php';
$fb = new Facebook\Facebook([...]);
$helper = $fb->getCanvasHelper();
$permissions = ['email'];
$profile_request = $fb->get('/me?fields=name,first_name,last_name,email');
$profile = $profile_request->getGraphNode()->
    asArray();
...
print_r($profile);
} else {
    $helper = $fb->getRedirectLoginHelper();
    $loginUrl = $helper->getLoginUrl('https://apps.facebook.com/phpcanvastest/', $permissions);
    echo "<script>window.top.location.href='".$loginUrl."'</script>";
}
```



Bild-Post in der Timeline (Bild 10)



Verwaltete Seite: Auch auf einer verwalteten Seite kann man einen Post platzieren (Bild 11)

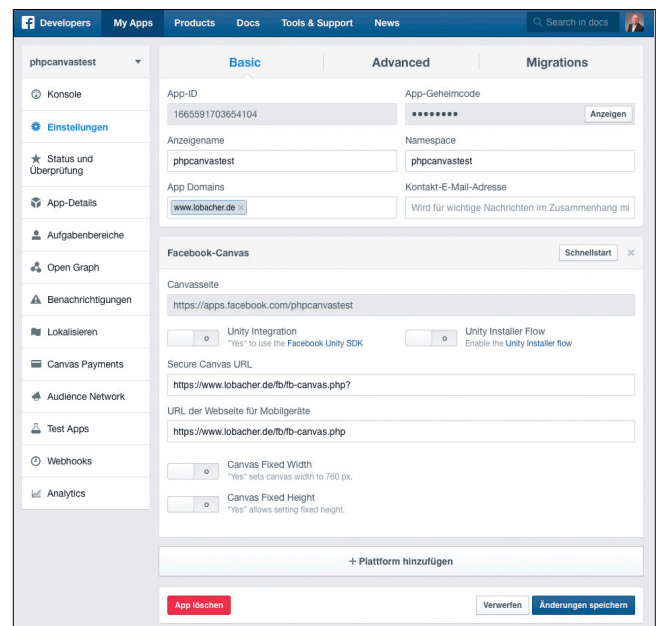


Ausgabe der Userdaten im Canvas (Bild 13)

die PHP-Datei enthält, die Sie innerhalb der Canvas anzeigen wollen (Bild 12). Nun können Sie die App-ID und das App-Secret in eine neue PHP-Datei kopieren, die Sie am besten von einer der vorherigen duplizieren (Listing 7).

Hier verwenden Sie als einzigen großen Unterschied nun den sogenannten `getCanvasHelper()`, um die Canvas-Funktionalität zu realisieren. Am Ende wird wiederum der Redirect zur Facebook-Seite realisiert.

Um die Funktionalität zu testen, rufen Sie den URL zum Skript auf (mittels HTTPS) und werden zu Facebook zurückgeleitet. Hier müssen Sie die Berechtigung gewähren. An-



Einstellungen der Canvas-Test-App (Bild 12)

schließend erhalten Sie die Ausgabe des `print_r()` innerhalb von Facebook (Bild 13).

Fazit

Wir haben mit den vorliegenden Beispielen nur an der Oberfläche der Möglichkeiten gekratzt, die das Graph API bietet. Letztlich kann nahezu jede Aktion, die innerhalb von Facebook möglich ist, auch programmatisch erstellt werden. ■

Links zum Thema

- Facebooks Developer-Bereich
<https://developers.facebook.com>
- Graph-API-Referenz
<https://developers.facebook.com/docs/graph-api>
- Facebook – Plattform-Überblick
<https://developers.facebook.com/docs/apps/changelog>
- Facebook SDK für PHP
<https://developers.facebook.com/docs/reference/php>
- Graph API Explorer
<https://developers.facebook.com/tools/explorer>



Patrick Lobacher

ist Digital Native, Entwickler, Berater, Trainer, Coach und Autor zahlreicher Fachbücher und Fachartikel. Er ist Vorstandsvorsitzender der Pluswerk AG, die an zehn Standorten mit über 150 Mitarbeitern digitale Kommunikationslösungen konzipiert, umsetzt und betreut.

PHP-BIBLIOTHEK SHELLWRAP

Bequemer Abstieg in die Shell

Eine kompakte Bibliothek vereinfacht die Nutzung der Linux-Shell mit Hilfe von PHP.

Auch wenn PHP unheimlich mächtig ist: Ab und zu ist es einfach notwendig oder sinnvoll, Shell-Kommandos aufzurufen. Das klappt dann prinzipiell mit Funktionen wie `exec()` oder `shell_exec()` auch ganz gut. Jenseits der einfachen Fälle wird es aber recht schnell unschön, wenn Themen wie Fehlererkennung oder das Escaping von Sonderzeichen anstehen.

Um dem Entwickler die Nutzung zu erleichtern, hat der GitHub-User James Hall (aka Mr Rio) die Klasse `ShellWrap` geschrieben, die eine konsistente Nutzung von Shell-Befehlen erlaubt und sich um den Kleinkram kümmert. Damit können Sie dann bequem aus Ihren Webanwendungen in die Niederungen der Shell absteigen oder die Kommandozeilen-Variante von PHP zur Programmierung von Skripts nutzen, die fallweise kurze Abstecher in die Shell machen (Bild 1).

Die Beispiele, mit denen wir Ihnen die Nutzung von ShellWrap zeigen, sind nur zur Illustration gedacht. Die meisten Aufgaben lassen sich besser in PHP lösen. Weil die konkreten Einsatzgebiete individuell verschieden sind, haben wir uns dafür entschieden, lieber kleine und nachvollziehbare Funktionen zu verwenden.

Ein erstes Beispiel

Haben Sie noch keine Composer-Abhängigkeit in Ihrem Projekt, installieren Sie die Library durch den Befehl:

```
composer require "mrrio/shellwrap @dev"
```

Ansonsten fügen Sie `mrrio/shellwrap @dev` der Liste der Abhängigkeiten in der Datei `composer.json` hinzu und führen die Zeile `composer update` aus. Mit dem Zusatz `@dev` erreichen Sie, dass auch die letzten Bugfixes berücksichtigt werden, die noch nicht den Einzug in eine eigene Version gefunden haben. Ansonsten gibt es nämlich ein Problem bei der Rückmeldung von Fehlern aus der Kommandozeile. Danach können Sie einen ersten Test starten:

```
require_once './vendor/autoload.php';  
echo MrRio\ShellWrap::pwd();
```

In der ersten Zeile wird der mit Composer gelieferte Autoloader aktiviert. Die zweite Zeile ruft den Wrapper mit dem Shell-Kommando `pwd` auf und gibt das aktuelle Verzeichnis zurück, also den Speicherort Ihres PHP-Skripts. Die Rück-

Examples

```
<?php  
require_once 'vendor/autoload.php';  
use MrRio\ShellWrap as sh;  
  
// List all files in current dir  
echo sh::ls();  
  
// Checkout a branch in git  
sh::git('checkout', 'master');  
  
// You can also pipe the output of one command, into another  
// This downloads example.com through cURL, follows location, then pipes through grep to  
// filter for 'html'  
echo sh::grep('html', sh::curl('http://example.com', array(  
    'location' => true  
)));  
  
// Touch a file to create it  
sh::touch('file.html');  
  
// Remove file  
sh::rm('file.html');  
  
// Remove file again (this fails, and throws an exception because the file doesn't exist)  
  
try {  
    sh::rm('file.html');  
} catch (ShellWrapException $e) {  
    echo 'Caught failing sh::rm() call';  
}  
  
// This throws an exception, as 'invalidoption' is not a valid argument  
try {  
    echo sh::ls(array('invalidoption' => true));  
} catch (ShellWrapException $e) {  
    echo 'Caught failing sh::ls() call';  
}  
  
// Commands can be written multiple ways  
sh::git('reset', array('hard' => true), 'HEAD');
```

Auf der GitHub-Seite des Autors finden Sie Beispielcode für alle wichtigen Aspekte von ShellWrap (Bild 1)

gabe der aufgerufenen Methode ist also immer die normale Textausgabe des verwendeten Befehls.

Eine praktische Vereinfachung erhalten Sie noch, wenn Sie den Wrapper per `use`-Statement unter dem Kürzel `sh` bekannt machen. Fügen Sie also Folgendes in Zeile 2 ein:

```
use MrRio\ShellWrap as sh;
```

Dann wird der Aufruf schön kurz und prägnant:

```
echo sh::pwd();
```

Weil von ShellWrapper statische Methoden angeboten werden, müssen Sie kein Objekt instanzieren und können direkt den Weg über die Klasse gehen und hinter die beiden Doppelpunkte das gewünschte Kommando schreiben. Die Klasse `ShellWrap` hat nun aber nicht etwa jeden bekannten Shell-

Befehl als Methode implementiert, sondern macht sich die magische Methode `__callStatic()` von PHP zunutze. Die wird immer dann aufgerufen, wenn eine Klasse die verwendete Methode nicht besitzt. Darin wird dann der jeweilige genannte Methodenname ausgewertet und in Folge das gleichnamige Shell-Kommandos aufgerufen.

Die Bibliothek kennt mehrere Methoden, um den Linux-Kommandos Argumente mit auf den Weg zu geben. Sehen Sie hier etwa ein Shell-Kommando, das ein Datum annimmt und in der Standard-Formatierung Ihres Systems ausgibt:

```
date --date '2015-12-31 23:59:59'
```

Die simpelste Methode, diesen Aufruf in PHP zu überführen, ist, die gewünschten Argumente einfach als String an die Methode zu übergeben. Um obige Zeile von PHP aus zu nutzen, könnten Sie also schreiben

```
echo sh::date("--date '2015-12-31 23:59:59'");
```

Verlangt Ihr Fall mehrere Parameter für ein Kommando, wird diese Methode unübersichtlich und die Verwendung von Anführungszeichen in der PHP-Strings ist ebenfalls recht fehlerträchtig. Dann verwenden Sie besser die andere Variante. Dabei packen Sie die Optionen und ihre Werte in ein Array:

```
echo sh::date(array(
    "date"=>"2015-12-31 23:59:59"
));
```

Neben der sauberen Trennung von Option und Wert hat diese Variante noch den Vorteil, dass die Bibliothek für Sie ein paar lästige Kleinigkeiten erledigt. So kümmert sie sich selbstständig um das Einschließen des Wertes in Anführungszeichen

und fügt vor dem Namen der Option die beiden Bindestriche ein. Wenn Sie lieber die kurzen Optionen nutzen, die nur aus einem einzigen Buchstaben oder einer Ziffer bestehen, so leitet ShellWrap diese automatisch mit einem einfachen Bindestrich ein. Die folgende Zeile klappt also genauso:

```
echo sh::date(array(
    "d"=>"2015-12-31 23:59:59"
));
```

Manche Optionen haben keinen Wert, sondern wirken als einfache Schalter. Beim Shell-Befehl `date` etwa bewirken Sie mit `-R`, dass als Format der Datumsausgabe die RFC 2822 beachtet wird. Das teilen Sie ShellWrap durch einen Wert `true` für den jeweiligen Parameter mit:

```
echo sh::date(array(
    "d"=>"2015-12-31 23:59:59",
    "R"=>true
));
```

Einfache Argumente, die im Shell-Befehl also nicht von einem Optionsnamen begleitet werden, fügen Sie im Aufruf von ShellWrap einfach als weitere Parameter ein. Angenommen, Sie möchten via CURL eine Datei von einem anderen Server abholen, der den Zugriff darauf per Basic Authentication abgesichert hat. Dann könnten Sie schreiben

```
$dat = sh::curl(array(
    'u' => 'user:passwort'
), 'http://xy.de/datei.csv');
```

Sie sehen also, dass die Nutzung eines Arrays zum Setzen der Argumente des Shell-Befehls die sauberere Lösung ist, vor ►

Sandkastenspiele

Wollen Sie mit ShellWrap einfach nur ein wenig herumexperimentieren, dann nutzen Sie den interaktiven Modus des Tools.

Dazu rufen Sie aus dem Verzeichnis, in das Sie die Erweiterung installiert haben, folgendes Kommando aus der Shell auf:

```
vendor/bin/shellwrap
```

Das Programm realisiert eine Read-Eval-Print-Loop (REPL). Sie geben also dort eine oder mehrere Zeilen ein, so wie Sie in PHP die Methode aufrufen würden, und bekommen als Rückmeldung den Wert angezeigt, den der Methodenaufruf zurückliefert.

Die einzige Besonderheit ist, dass Sie nicht wie in einem Skript `sh:: ...` verwenden können, sondern Sie nutzen stattdessen `$sh:: ...`. Um etwa das aktuelle Verzeichnis darstellen zu lassen, tippen Sie den Befehl `$sh::ls()`; ein.

Mit der Tastenkombination [Strg D] können Sie die Shell dann wieder verlassen.

```
root@vmdev:/var/www/doodle/shellwrap# vendor/bin/shellwrap
ShellWrap Interactive Shell

Mapped to $sh due to eval() limitations. Ctrl+D to exit.

[1] shellwrap> $sh::pwd();
/var/www/doodle/shellwrap
// 0
[2] shellwrap> $sh::ls('-l');
insgesamt 48
-rwxr--r-- 1 www-data www-data 172 Dez  3 13:06 callback.php
-rw-r--r-- 1 root      root      61 Dez  2 12:09 composer.json
-rw-r--r-- 1 root      root     3555 Dez  2 12:09 composer.lock
-rwxr--r-- 1 www-data www-data 278 Dez  2 16:06 error.php
-rw-r--r-- 1 root      root      18 Dez  3 13:41 log
drwxr-xr-x 3 www-data www-data 4096 Dez  2 14:12 mrrio
-rwxr--r-- 1 www-data www-data 614 Dez  3 09:03 params.php
-rwxr--r-- 1 www-data www-data 152 Dez  2 14:16 piping.php
-rwxr--r-- 1 www-data www-data  91 Dez  1 16:49 simple.php
-rwxr--r-- 1 www-data www-data 181 Dez  3 13:39 stream.php
-rwxr--r-- 1 www-data www-data 150 Dez  3 11:10 trim.php
drwxr-xr-x 6 root      root     4096 Dez  2 12:09 vendor
// 0
[3] shellwrap> $sh::printenv();
LESSOPEN=| /usr/bin/lesspipe %s
MAIL=/var/mail/root
```

Mit dem Shell-Modus erhalten Sie eine Spielwiese, wo Sie direkt ShellWrap-Funktionen eintippen und die Wirkungsweise überprüfen können

allem, wenn Sie die Optionen dynamisch per Skript zusammenstellen möchten oder es sich um viele Optionen handelt, die gesetzt werden müssen.

So verarbeiten Sie die Ausgaben korrekt

Sie müssen sich immer klar machen, dass die Rückgabewerte der Shell-Aufrufe keine »normalen« Strings sind, sondern es sich dabei wirklich um die Ausgaben der aufgerufenen Kommandos handelt. Insbesondere bedeutet das, dass auch die Rückgabewerte, die nur aus einer Zahl oder einem Wort bestehen, am Ende eine Zeilenschaltung aufweisen. Sie sollte es sich also zur Regel machen, die Ergebnisse per `trim()` zu rechtzustutzen.

Zusätzlich sollten Sie wissen, dass die Aufrufe ein ShellWrap-Objekt zurückgeben, das erst über die implizit aufgerufene Methode `__toString()` die Ausgabe in Textform zurückliefert. In der Regel erfolgt das automatisch, aber es kann nicht schaden, dies durch einen Cast sicherzustellen wie in:

```
$dat = (string) sh::curl(...);
```

Handelt es sich bei der Ausgabe um mehrere Zeilen, dann möchten Sie vielleicht nach dem Aufruf darauf geordnet zugreifen können. Dazu können Sie die Ausgabe zum Beispiel in ein Array umwandeln:

```
$arrFiles = explode("\n", (string) trim(sh::ls()));
```

Das `explode()` macht sich das Newline-Steuerzeichen am Ende jeder Zeile zunutze, um die Rückgabemeldung aufzusplitten. Das `trim()` ist hier notwendig, damit der letzte Zeilenvorschub nicht zu einem leeren Element am Ende des Arrays führt.

So verketteten Sie Shell-Aufrufe

Was zum großen Teil das Wesen der Arbeit in einer Linux-Shell ausmacht, das Aneinanderhängen von Kommandos, ist auch per ShellWrap möglich, wenn auch mit anderer Syntax.

So soll etwa die per CURL abgerufene Datei aus dem vorigen Beispiel vielleicht nicht vollständig verarbeitet werden, sondern nur die Zeilen, die die Kennung *KundeXY* beinhalten. In der Shell gelingt diese Filterung viel einfacher als in PHP, indem man an das Kommando für das Holen der Datei einfach noch `| grep KundeXY` anhängt.

In der Welt von ShellWrap baut man diese zusätzliche Filterung so ein:

```
$filtdat = sh::grep('KundeXY',
    sh::curl(array(
        'u' => 'user:passwort'),
        'http://xy.de/datei.csv'));
);
```

Statt das `grep`-Kommando hinten dranzuhängen, nutzen Sie also den Output von `sh::curl()` als Eingabe von `sh::grep()`.

ShellWrap bietet einen speziellen Modus, der in manchen Fällen sehr hilfreich ist. Dabei legen Sie als letzten Parameter eine Closure fest. Die wird bei jeder eingelesenen Ausgabezeile mit dem Inhalt der aktuellen Zeile als Parameter aufgerufen. Das ist praktisch, weil Sie oft eine Funktionalität einfacher auf dem Kontext einzelner Zeilen aufsetzen können, anstatt im ersten Schritt auf die Ausgabe zu warten, um diese dann zeilenweise zu beackern.

Diese Stream-Logik sehen Sie an folgendem Beispiel des ShellWrap-Autors:

```
sh::tail('-f log', function($in) {
    echo "\033[32m" . date('Y-m-d H:i:s');
    echo "\033[39m " . $in ;
});
```

Das Skriptfragment setzt das Kommando `tail` mit dem Parameter `-f` ab. Dieser bewirkt, dass die beispielhaft genannte Datei `log` ständig überwacht und jede neue Zeile ausgegeben wird. Die gibt dann die anonyme Funktion aus und reichert sie mit einem Timestamp und Escape-Sequenzen an, die im Terminal unterschiedliche Farben erhalten.

Wegen der Verwendung als Dauerläufer taugt das Skript also nur im Kommandozeilen-Modus von PHP. Zum Testen lassen Sie in einem Terminalfenster das PHP-Skript selbst laufen. Dann legen Sie sich in einem anderen Fenster die Datei `log` im selben Verzeichnis an und schieben per `echo` neue Zeilen in die Datei (Bild 2).

Tritt ein Fehler beim Ausführen eines Kommandos auf, wirft ShellWrap eine Exception. Hier ein einfaches Beispiel:

```
try {
    sh::rm('unbekannt.dat');
} catch (Exception $e) {
    echo "Fehler beim Löschen der Datei! ";
}
```

Gibt es die Datei nicht, schlägt das Löschkommando fehl und der `catch`-Zweig mit der Meldung wird aufgerufen.

```
<?php
require_once './vendor/autoload.php';
use MrRio\ShellWrap as sh;

sh::tail('-f log', function($in) {
    echo "\033[32m" . date('Y-m-d H:i:s') . "\033[39m " . $in;
});

root@vmdev: /var/www/doodle/shellwrap
root@vmdev: /var/www/doodle/shellwrap# touch log
root@vmdev: /var/www/doodle/shellwrap# php stream.php
2015-12-03 13:41:06 Test1
2015-12-03 13:41:11 Test2
2015-12-03 13:41:34 Test3

root@vmdev: /var/www/doodle/shellwrap
root@vmdev: /var/www/doodle/shellwrap# echo Test1 >> log
root@vmdev: /var/www/doodle/shellwrap# echo Test2 >> log
root@vmdev: /var/www/doodle/shellwrap# echo Test3 >> log
root@vmdev: /var/www/doodle/shellwrap#
```

Das Beispiel zeigt, wie durch den Einsatz einer anonymen Funktion ShellWrap in einem Stream-Modus genutzt werden kann (Bild 2)

Links zum Thema

- Homepage des Projekts auf GitHub
<https://github.com/MrRio/shellwrap>

Zur näheren Untersuchung des Problems gibt Ihnen das Exception-Objekt mit der Methode `getCode()` den Exit-Code des Shell-Kommandos, und Sie erhalten über `getMessage()` die Fehlermeldung. Diese entspricht dem, was der Befehl bei direkter Nutzung in der Konsole auf dem Fehlerkanal zurückliefern würde.

So können Sie also entweder dem Benutzer noch weitere Hinweise mitgeben oder je nach Exit-Code verschiedene Maßnahmen nach dem Auftreten des Problems ergreifen. Um dem Benutzer lediglich mehr Infos zu bieten, könnten Sie das Skript also erweitern und um die folgenden Code-Zeilen ergänzen:

```
try {
    sh::rm('unbekannt.dat');
} catch (Exception $e) {
    echo "Fehler beim Löschen der Datei!\n";
    echo "Exit-Code: " . $e->getCode() . "\n";
    echo "Info: " . $e->getMessage() . "\n";
}
```

Die Antwort würde dann so lauten:

```
Fehler beim Löschen der Datei!
Exit-Code: 1
Info: /bin/rm: cannot remove 'unbekannt.dat': No such
file or directory
```

Die statische Nutzung von ShellWrap ist in der Verwendung schön kompakt. Sie können die Bibliothek aber genauso wie eine normale Klasse verwenden, was manchmal unabdingbar ist.

Wie eine normale Klasse verwenden

Ein Beispiel: Sie möchten per Shell-Aufruf herausfinden, wohin der Apache-Server seine Logs schreibt. Die Information über das definierte Ausgabeverzeichnis steckt in der Umgebungsvariablen `APACHE_LOG_DIR`. Auf der Shell selbst würden Sie diese Information so ausgeben:

```
echo $APACHE_LOG_DIR
```

Der erste Versuch, dieses Kommando in PHP zu realisieren, sähe vielleicht so aus:

```
$logdir = sh::echo('$APACHE_LOG_DIR');
```

Doch hier funkt PHP selbst dazwischen und bricht die Ausführung ab. Denn `echo` ist ein reserviertes Wort und damit als Methodenname verboten.

Darum gehen Sie einen kleinen Umweg und legen eine Instanz von ShellWrap an:

```
$sh = new sh();
echo $sh('echo', '$APACHE_LOG_DIR');
```

Der gewünschte Shell-Befehl wird hierbei also als erster Parameter angegeben. Dabei wirkt eine andere magische Methode `__invoke()`, die dann aufgerufen wird, wenn ein Objekt wie eine Funktion genutzt wird.

Ein anderer Trick wird notwendig, wenn sich das gewünschte Programm oder Kommando an einem unüblichen Ort befindet. Normalerweise sucht sich ShellWrap die passende ausführbare Datei unter Zuhilfenahme des Shell-Befehls `which`, um beim Aufruf den Pfad mit angeben zu können. Wenn Sie nun ein Shell-Programm aufrufen wollen, das nicht in einem der in `PATH` genannten Verzeichnisse liegt, müssen Sie ShellWrap den Pfad zu diesem Programm eigens angeben. Das klappt von der Syntax her natürlich nicht mit dem bequemen statischen Aufruf, da hier ja keine Slashes erlaubt sind. Stattdessen schreiben Sie zum Beispiel:

```
$sh = new sh();
$ausgabe = $sh('/irgendein/dir/kommando');
```

Dafür, dass der Linux-User, mit dem Ihr PHP arbeitet, die jeweilige Datei auch ausführen darf, sorgt dieses Kommando aber nicht. Sie müssen ihm also trotzdem gegebenenfalls die entsprechenden Rechte einräumen.

Eine Warnung vor Missbrauch

Auch wenn der Linux-User, unter dem PHP läuft, keine weitreichenden Rechte hat, ist ein Shell-Aufruf immer ein potenzielles Einfalltor für Angriffe. Sie sollten es darum tunlichst vermeiden, User-Input in Ihre Shell-Aufrufe aufzunehmen. Auch Filtern bietet keinen hundertprozentigen Schutz. Für viele der Filter-Mechanismen gibt es trickreiche Aushebungen, um dennoch Schadcode einzuschleusen. Es genügt also beispielsweise nicht, die Argumente mit der PHP-Funktion `escapeshellarg()` zu umschließen, um sichere Skripts zu erhalten. Bestenfalls können Sie es erlauben, dass der User numerische Werte vorgeben darf. Denn die können Sie durch dazu passende Casts oder Funktionen wie `intval()` so entschärfen, dass sie keine Gefahr mehr darstellen. Strings dagegen sind potenziell böse und sollten nicht vom Input übernommen werden. ■



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.

GRAFIK FÜR ENTWICKLER

Gestochen scharf

Bildschärfe entspricht nicht dem subjektiven Schärfeeindruck – dieser kann durch geschickte Nachbearbeitung erzielt werden.

Unschärfe Kanten im Bild entstehen entweder bereits bei der Aufnahme oder durch eine falsche Bearbeitung des Fotos. Neben dem unschönen, verwaschenen Bildeindruck verbergen solche Fotos zudem oft wesentliche Teile der Bildinformationen: So gehen feine Linien und Zeichnungen in den Flächen verloren und das beste Motiv verliert so schnell an Aussagekraft und künstlerischem Wert.

Ob im Druck oder auf einem digitalen Ausgabegerät – unscharfe Bilder wirken schnell unprofessionell. Ein geeignetes Bild sollte also nicht nur ein spannendes Motiv zeigen, es sollte neben den korrekten Farben und einem stimmigen Kontrast auch die optimale Schärfe aufweisen (Bild 1).

Bildschärfe

Ob ein Foto tatsächlich scharf ist, ist oft nicht so leicht zu beurteilen. Betrachtet man den gerade geschossenen Schnappschuss auf dem Display des Smartphones, mag er einen makellosen Schärfeeindruck vermitteln. Erst auf einem großen Monitor zeigen sich dann die Schwächen.

Physikalische Bildschärfe wird mit Hilfe von Linien pro Millimeter gemessen, die noch voneinander unterschieden werden können. Der dabei ermittelte Zahlenwert wird jedoch zur Bewertung von Objektiven, Kameras oder Druckern verwendet und kann nicht als eindeutiges Maß für die Beurteilung des fotografischen Bildeindrucks dienen.

Scharfe Vektoren

Eine eindeutige Definition für Bildschärfe gibt es also nicht – zumindest nicht bei Bildern, die sich aus einzelnen Bildpunkten zusammensetzen. Wären die Farb- und Helligkeitswerte



Von der Vektorgrafik über ein Photoshop-Smartobjekt bis hin zum GIF: Lediglich die Vektorgrafik liefert tatsächlich scharfe Kanten (Bild 2)

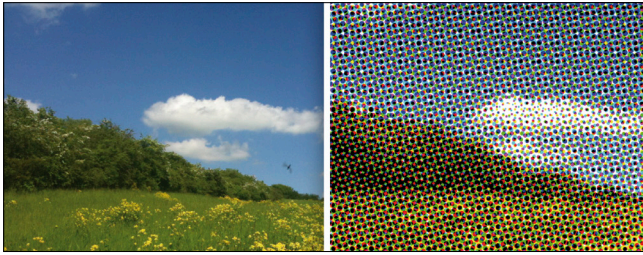


Unschärfe: Die scharfe Variante des gleichen Motivs wirkt deutlich plastischer (Bild 1)

der Bildpunkte in einem Foto an Motivkanten exakt voneinander getrennt, würden sich entlang der Kante kleinste Stufen bilden, die, je nach Bildauflösung, mehr oder weniger stark zu erkennen sind. Diese sogenannte Treppchenbildung bei Pixelbildern wird in der Regel durch Zwischenwerte ausgeglichen, um den Eindruck von möglichst glatten Kanten zu erzeugen. Anders sieht es bei einer Vektorgrafik aus: Hier begrenzen Pfade, die durch Ankerpunkte definiert werden, einzelne Flächen. Eine Vektorgrafik ist somit nicht aus einzelnen Bildpunkten aufgebaut und kann in der Größe frei transformiert werden. Ein Beispiel dazu ist in Bild 2 zu sehen: Bei der linken Grafik handelt es sich um die originale Vektordatei. Besonders in den roten Rändern der Buchstaben sind die einzelnen Ankerpunkte (türkis) gut zu erkennen.

In der Mitte wurde die Vektorgrafik in Photoshop CC 2015 als Smartobjekt platziert. Beim Smartobjekt behält Photoshop die originalen Daten, sodass nach einer Verkleinerung des Objekts jederzeit ohne Qualitätsverlust auf den Originalzustand zurückgegriffen werden kann. Dennoch werden letztendlich die Vektorinformationen in einzelne Bildpunkte umgewandelt; eine Vergrößerung führt hier zu mehr oder weniger unscharfen Kanten – abhängig vom Vergrößerungsfaktor.

Rechts: Nun soll die Grafik als GIF ausgegeben werden – es handelt sich ja um glatte Flächen, die keine Verläufe zeigen. GIF arbeitet mit indizierten Farben, stellt also höchstens 256 Farbwerte zur Verfügung. Werden, wie der obere Bild-



Auflösung: Ob ein Bild scharf ist, hängt auch von der Auflösung ab (Bild 3)

ausschnitt zeigt, exakt die drei Farbwerte Rot, Schwarz und Weiß gewählt, die die Grafik beinhaltet, kommt es zur gefürchteten Treppchenwirkung. Bei 256 Farben hingegen erscheinen die Kanten wesentlich glatter, da hier verschiedene Tonwerte zwischen Weiß und Rot die Lücken füllen.

Subjektiver Schärfeeindruck

Soll die Qualität eines Fotos beurteilt werden, spielt der subjektive Schärfeeindruck eine wichtige Rolle. Denn ob ein Bild scharf ist, hängt stark von der subjektiven Wahrnehmung des Betrachters ab, die sehr unterschiedlich sein kann.

Hier sei zunächst das Auflösungsvermögen des menschlichen Auges im Zusammenhang mit der Bildauflösung genannt: Je größer der Abstand zum Motiv, desto weniger Details sind zu erkennen. Daher kann etwa ein Plakat mit einer viel geringeren Auflösung gedruckt werden als ein Bild in einem Magazin. Beim Plakat sind aus der entsprechenden Entfernung einzelne Druckpunkte nicht mehr wahrzunehmen; betrachtet man jedoch einen Bildausschnitt aus der Nähe, lösen sich die Flächen in viele kleine Rasterpunkte auf. Beim Druckraster eines hochwertigen Magazins sind die einzelnen Bildpunkte mit bloßem Auge auch aus der Nähe nicht zu erkennen, hier arbeitet der professionelle Druck mit einer Auflösung von 300 ppi. Beim großflächigen Plakat hingegen beträgt der Betrachtungsabstand etwa fünf Meter; die Druckerauflösung wird in diesem Fall mit 20 ppi angesetzt.

Auf Bild 3 ist beispielweise links ein Foto zu sehen, auf dem feine Details, etwa die Blumen im Vordergrund, gestochen



Kontrast/Bildschärfe: Links fehlt der Kontrast, das Bild wirkt flau; rechts entsteht durch die Nachbearbeitung der Farben und des Kontrasts eine subjektive Bildschärfe (Bild 4)

scharf zu sehen sind. Rechts daneben wurde derselbe Bildausschnitt in Photoshop per Filter zu einem Farbraster umgewandelt. Zwar sind die größeren Flächen noch halbwegs zu erkennen, Feinheiten gehen jedoch gänzlich verloren – das Bild als Ganzes scheint unscharf zu sein, obwohl die einzelnen Punkte des Farbrasters für sich genommen gestochen scharf sind.

Schärfe durch Kontrast

Abgesehen von der passenden Auflösung erscheint ein Bild dann scharf, wenn die abgebildeten Kanten einen hohen Kontrast zeigen. Erst durch diese sogenannte Kantenschärfe sind auch kleinere Details gut zu erkennen. In der Fotografie wird die Schärfe der Konturen durch die Lichtöffnung bestimmt. Je kleiner die Öffnung, desto schärfer das Bild. Eine Kamera obscura verdeutlicht diesen Effekt.

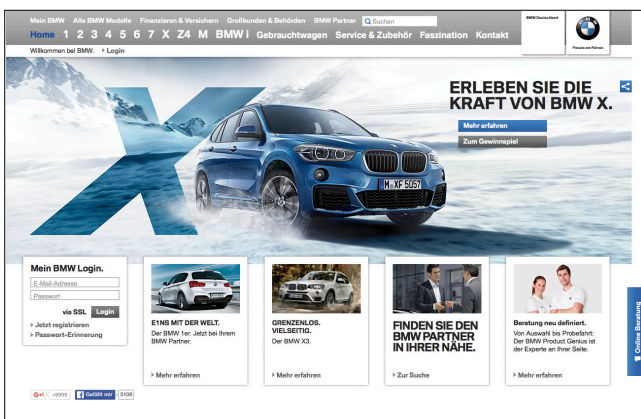
Ein gutes Beispiel dafür, wie mangelnder Kontrast die Kantenschärfe beeinflusst, zeigt Bild 4: Im linken Bildausschnitt fehlt der Kontrast, das Bild wirkt dadurch unscharf, obwohl alle notwendigen Bildinformationen gegeben sind. Durch die Nachbearbeitung von Farbe und Kontrast werden im gleichen Ausschnitt rechts feine Linien im Motiv besser sichtbar und eine subjektive Bildschärfe entsteht.

Künstlerische Unschärfe

Nicht immer muss jedes Bilddetail gestochen scharf erscheinen. Unschärfe Bereiche können ein scharfgezeichnetes Motiv oder Teile davon hervorheben, um so die Bildaussage zu unterstützen. Wehende Haare, die durch die Aufnahmetechnik Bewegungsunschärfe zeigen, bringen so beispielsweise Bewegung ins Bild und heben das Gesicht besonders hervor.

Dieser Kunsttrick wird auch auf einigen Webseiten sinnvoll eingesetzt. So bildet etwa die Startseite von BMW das Modell X5 gestochen scharf ab. Lediglich die Felgen, die Winterlandschaft im Hintergrund und der von den Reifen aufgewirbelte Schnee zeigen eine deutliche Unschärfe. Wären alle Bildbereiche scharfgezeichnet, würde das Bild wie eingefroren wirken. Erst durch die an den richtigen Stellen eingesetzte Bewegungsunschärfe entsteht die gewünschte Dynamik (Bild 5).

Wer seinen Fotos künstlerische Unschärfe hinzufügen möchte, findet verschiedene Apps, die unterschiedliche ►



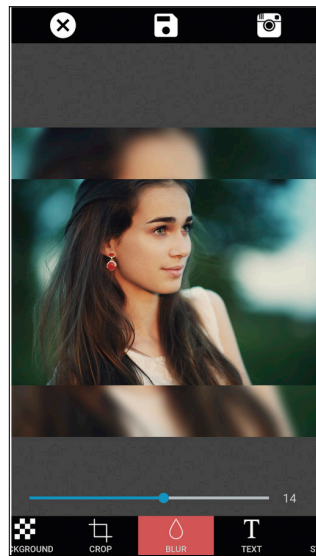
BMW setzt Bewegungsunschärfe bewusst ein, um Dynamik zu erzeugen (Bild 5)

Scharf- und Weichzeichner-Effekte bieten. So etwa die Google-App Photo Editor Collage Maker von Lyrebird Studio, die neben verschiedenen Collage-Werkzeugen einen Weichzeichner für den Hintergrund bietet (Bild 6). Ähnlich funktioniert der Blur-Filter der iPhone-App Photogene, mit dem per Fingerwisch Unschärfe in das Bild gepinselt werden kann (Bild 7).

Schärfemethoden

Jegliche Bildbearbeitung ist im Grunde genommen lediglich ein Trick, bei dem verschiedene Pixel in ihrer Farbe und dem Tonwert so verändert werden, dass dadurch gewisse Dinge besser in den Vordergrund treten. So erhöht ein Scharfzeichnungsfilter den Kontrast entlang den Bildkanten. Dabei erkennt er jedoch tatsächlich keine Kanten in Bildern, sondern sucht nach Pixeln, die sich um einen individuell eingestellten Schwellenwert von den benachbarten Pixeln unterscheiden. Der Kontrast dieser Pixel wird dann um einen mehr oder weniger bestimmten Wert erhöht.

Nun zeigt jedes Bild unterschiedliche Kontraste und Farbwerte. Zudem kann das Foto eine geringe oder eine hohe Auflösung mitbringen. Auch verschiedene Bildgrößen erfordern unterschiedliche Einstellungen eines Filters. Um also ein gutes Ergebnis zu erzielen, sollten Filter mehrere Einstellungen liefern, die der Anwender individuell justieren kann. Zwar bieten mittlerweile einige Bildbearbeitungsprogramme recht gute Schärfefilter, die sich per Schieberegler einstellen lassen. Adobe Photoshop CC 2015.1 ermöglicht jedoch verlustfreies Arbeiten – wird ein Filter angewandt, kann diese Einstellung auch nachträglich geändert, oder sogar rückgängig gemacht werden. Zuständig ist hier das Bedienfeld Kor-



Lyrebird Studio bietet einen Weichzeichner für eigene Fotos (Bild 6)

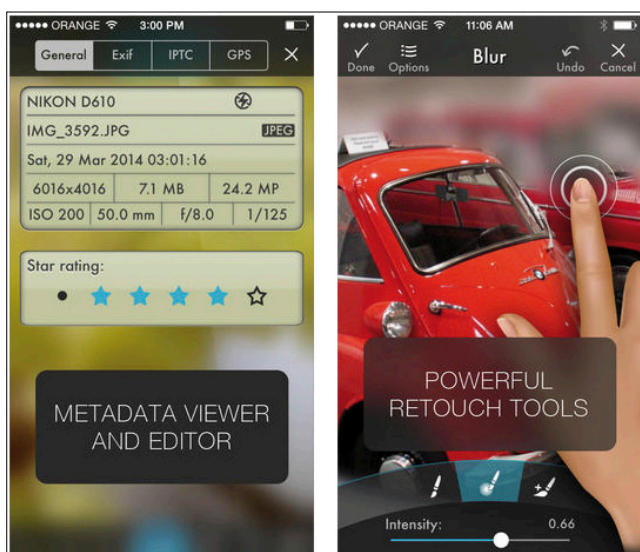
rekturen: Jede Korrektur legt das Programm als eigene Ebene über das Bild und belässt so die Bildpunkte des Originals. Zudem erscheint neben der Korrektur eine Maske, die es ermöglicht, verschiedene Bildbereiche vor der Korrektur zu schützen (Bild 8).

Leider bietet Adobe Korrektorebenen nur zur Anpassung von Farben und Kontrast. Soll die Schärfe bearbeitet werden, sind die Einstellungen dazu im Menü *Filter* zu finden – und ändern bei der Anwendung die Pixel im Bild direkt. Um auch hier verlustfrei zu arbeiten, muss das Bild zunächst für Smartfilter konvertiert werden, eine Funktion, die ebenfalls im *Filter*-Menü zu finden ist. Alle Filter aus dem Menü werden nun bei deren Anwendung auf das Bild automatisch zu Smartfiltern: Im Ebenenbedienfeld sind die Korrekturen nun der eigentlichen Bilderebene untergeordnet und liefern zudem eine bearbeitbare Maske.

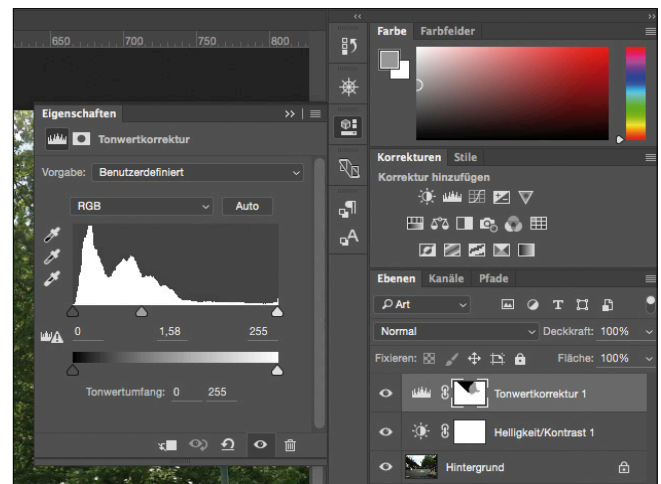
Photoshops Scharfzeichner

Insgesamt sechs verschiedene Filter sind unter den Scharfzeichnungsfiltern zu finden (Bild 9). Dabei sind nicht alle gleichermaßen für die professionelle Bildbearbeitung geeignet: Die drei Filter *Konturen scharfzeichnen*, *Scharfzeichnen* und *Stärker scharfzeichnen* ändern Pixel im Bild, ohne dem Anwender zuvor verschiedene Einstellungsmöglichkeiten zu bieten. Das mag zwar bei einigen Motiven zu einem verbesserten Ergebnis führen, generell ist von diesen Filtern jedoch aus den bereits genannten Gründen abzuraten.

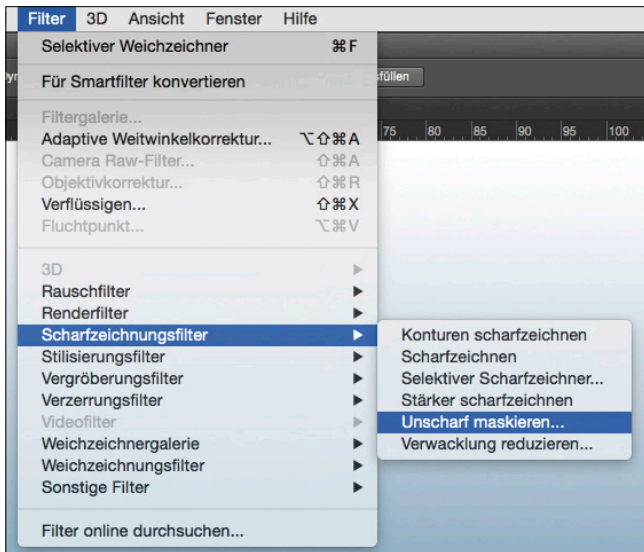
Die professionelle Methode zum Schärfen von Bildern ist das *Unschärf maskieren*. Drei Parameter stehen hier zur Schärfung bereit: Die Stärke bestimmt den Kontrast der Bildpunkte entlang der Kanten, um den Schärfeeffekt zu erzielen. Dieser Wert sollte nur in Ausnahmefällen höher als etwa 120 liegen. Der Radius definiert die Breite des Kantenbe-



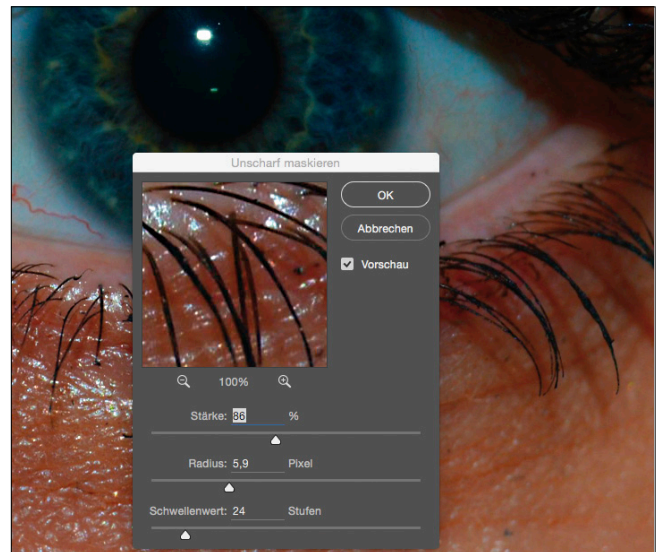
Auch die iPhone-App Photogene ermöglicht künstlerische Unschärfe (Bild 7)



Photoshop CC 2015 arbeitet dank Einstellungsebenen verlustfrei (Bild 8)



Scharfzeichner: Photoshop bietet sechs verschiedene Scharfzeichner (Bild 9)

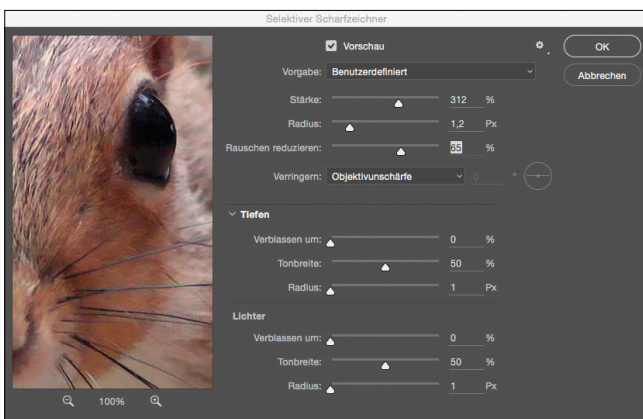


Unscharf maskieren ist der bekannteste professionelle Scharfzeichner (Bild 10)

reichs. Mit dem Schwellenwert gelingt eine Unterscheidung der scharf gezeichneten Pixel von den umliegenden flächigen Bildstellen (Bild 10).

Ebenfalls sehr gut einstellbar ist der *Selektive Scharfzeichner*. Dessen Dialog zeigt in der aktuellen Version Photoshop CC 2015.1 eine neue Bedienoberfläche, die Funktionalität ist jedoch wie gehabt: Stärke und Radius lassen sich hier wie bei dem Filter *Unscharf maskieren* einstellen. Hinzu kommt hier auch die Möglichkeit, das Rauschen zu reduzieren. Scharfgezeichnete Kanten werden dabei nicht weichgezeichnet. Zudem lassen sich hier die Tiefen und Lichter gesondert einstellen (Bild 11).

Für gut beleuchtete Bilder, die möglichst wenig Bildrauschen zeigen, aber dennoch verwackelt sind, bietet Photoshop den Filter *Verwacklung reduzieren*. Dabei wird der Bereich analysiert, in dem außerhalb der Motivkanten Farbrückstände durch die Verwacklung zu sehen sind. Meist zeigen solche Bilder jedoch auch mehr oder weniger starkes Bildrauschen, sodass dieser Filter eher selten zum Einsatz kommt.



Der selektive Scharfzeichner ermöglicht eine gesonderte Schärfung von hellen und dunklen Bereichen (Bild 11)

Generell sollten bei allen Filtern zum Scharfzeichnen nur entsprechend niedrige Werte gewählt werden. Zu hohe Werte führen häufig zu Artefakten an den Kanten, etwa Farbsäume durch das Überscharfen. Eine Kontrolle aller Bildbereiche in einem hohen Zoom-Faktor ist also anzuraten.

Fazit

Zwar bietet Photoshop leistungsstarke Filter zur Nachbearbeitung, zaubern kann das Programm jedoch nicht. So lassen sich nur die Bilder halbwegs zufriedenstellend bearbeiten, die auch vor den Korrekturen annähernd stimmig waren.

Vor dem Schärfen sollten zudem alle anderen Bildkorrekturen erledigt worden sein. Gerade ein stimmiger Kontrast und die passenden Farben lassen das Bild oft knackiger erscheinen, wodurch dann nur weniger Schärfen vonnöten ist.

Denn das Schärfen ist die schwierigste Disziplin: Sind die Kanten des Motivs völlig verschwommen, kann auch der beste Filter sie nicht ausfindig machen. Dennoch lohnt sich auch bei guten Fotos ein leichtes Schärfen durchaus. Die Einstellungen sollten dabei so gewählt werden, dass die Änderungen in der Vergrößerung gerade noch zu sehen sind. Eine Kontrolle des kompletten Bildes am Ende ist unabdingbar, da sich gerne im Hintergrund überschärfte Pixel einschleichen. Dank der Smartfilter-Technik lassen sich jedoch auch solche Schnitzer noch im Nachhinein ausbessern. ■



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 und verwirklichte mehrere Buchprojekte.



Foto: shutterstock@janet Faye Hastings

EINPLATINENCOMPUTER ORANGE PI

Raspberry-Konkurrent

Aus China kommen mittlerweile Raspberry-Konkurrenten, die für weniger Geld mehr Hardwareleistung bieten.

Wir konzentrieren uns in diesem Artikel auf den für rund 15 US-Dollar erhältlichen Orange Pi, der die größte Verbreitung genießt und deswegen am meisten Community-Support abbekommt.

Shenzhen Xunlong bietet auch andere Varianten der Platine an, die sich im Bereich der verbauten Hardware unterscheiden. Der normale Orange Pi bringt beispielsweise ein WiFi-Modul und einen VGA-Ausgang mit, muss dafür aber mit einem zweikernigen Prozessor auskommen (**Bild 1**).

Vierkernprozessor an Bord

Im Vergleich zum Raspberry Pi 2 tritt der mit bis zu 1,6 GHz wesentlich höher getaktete Vierkernprozessor aus dem Hause AllWinner in den Vordergrund. Interessanterweise kann er diese Taktrate nicht permanent halten. In der Praxis arbeitet die CPU meist mit rund 1,2 GHz (siehe http://linux-sunxi.org/Orange_Pi_PC#CPU_clock_speed_limit).

Als weitere Besonderheiten spendiert Shenzhen Xunlong dem Computer ein Mikrofon und eine Infrarotschnittstelle: Fernbedienungen finden auf diese Weise schnell und unbürokratisch Anschluss an das Computersystem.

Vom Raspberry Pi umsteigende Nutzer müssen sich mit einer kleinen Besonderheit arrangieren: Die Micro-USB-Buch-

se des Orange Pi dient nur zum Anschluss von USB-OTG-Peripherie. Für die Stromversorgung ist ein externes Netzteil zwingend erforderlich. Wer Telefon-Ladegeräte anschließt, riskiert massive Hardwareschäden.

Einkauf in Einzelteilen

Shenzhen Xunlong bietet die gesamte Orange Pi-Serie nur per AliExpress an. Das für Endkunden vorgesehene Portal ist ein Ableger der Großhandelsplattform Alibaba. Dort akzeptiert man alle Kreditkarten außer AMEX.

Achten Sie darauf, dass das in **Bild 2** gezeigte und unter www.aliexpress.com/item/Orange-Pi-PC-ubuntu-linux-and-android-mini-PC-Beyond-and-Compatible-with-Raspberry-Pi-2/32448079125.html aufrufbare Listing nur den eigentlichen Prozessrechner umfasst. Das Netzteil und die für das Betriebssystem benötigte Speicherkarte müssen separat erworben werden.

AliExpress funktioniert – allen Vorurteilen zum Trotz – in der Praxis so gut wie immer ohne Probleme. Der Orange Pi des Autors war in rund 14 Tagen in der Slowakei.

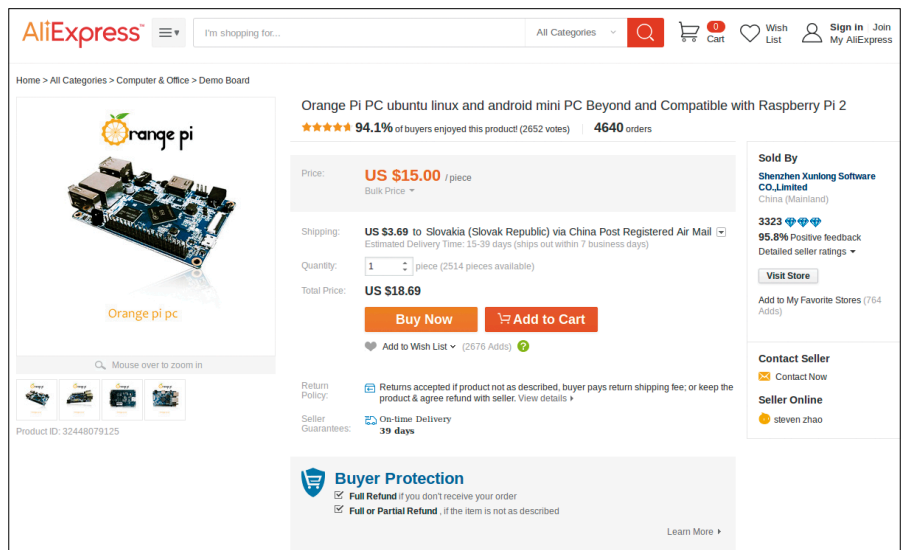
Achten Sie beim Bestellen größerer Mengen darauf, dass die Anbieter solche Orders gern in einem Paket versenden. Dieses für das in China sitzende Unternehmen kostensparen-

de Vorgehen ist für Sie insofern suboptimal, als der Zoll in diesem Fall den Gesamtwert des Pakets verrechnet. Bei zwei separat versendeten Orange-Pi-PCs fällt durch die Freibetragsregelungen kein Zoll an, während ein Paket mit zwei Einplatinencomputern Kosten verursacht.

Diverse Betriebssysteme

Der Hersteller verspricht, dass der Orange Pi mit Android 4.4, Debian, Ubuntu und dem von Raspberry Pi und Co. bekannten Raspbian-Betriebssystem kompatibel ist. Wer die unter www.orangepi.org/downloadresources befindliche Downloadseite von Shenzhen Xunlong ansteuert, erlebt sein blaues Wunder – die Betriebssystemimages werden zum Zeitpunkt der Drucklegung ausschließlich per Baidu ausgeliefert, das in **Bild 3** gezeigte Google-Drive-Logo ist somit reine Makulatur.

Wir wollen für unsere ersten Schritte das von Raspberry Pi und Co. bekannte Raspbian verwenden. Der Xgadget-Blog



Bestellseite: Der Orange-Pi-PC aus Sicht von AliExpress (**Bild 2**)

hat das Betriebssystem auf einigen Freehostern hochgeladen. Wer des Chinesischen nicht mächtig ist und etwas Wartezeit nicht scheut, findet unter <https://www.xgadget.de/app-software/orange-pi-pc-raspbian-download> ebenfalls eine Downloadmöglichkeit.

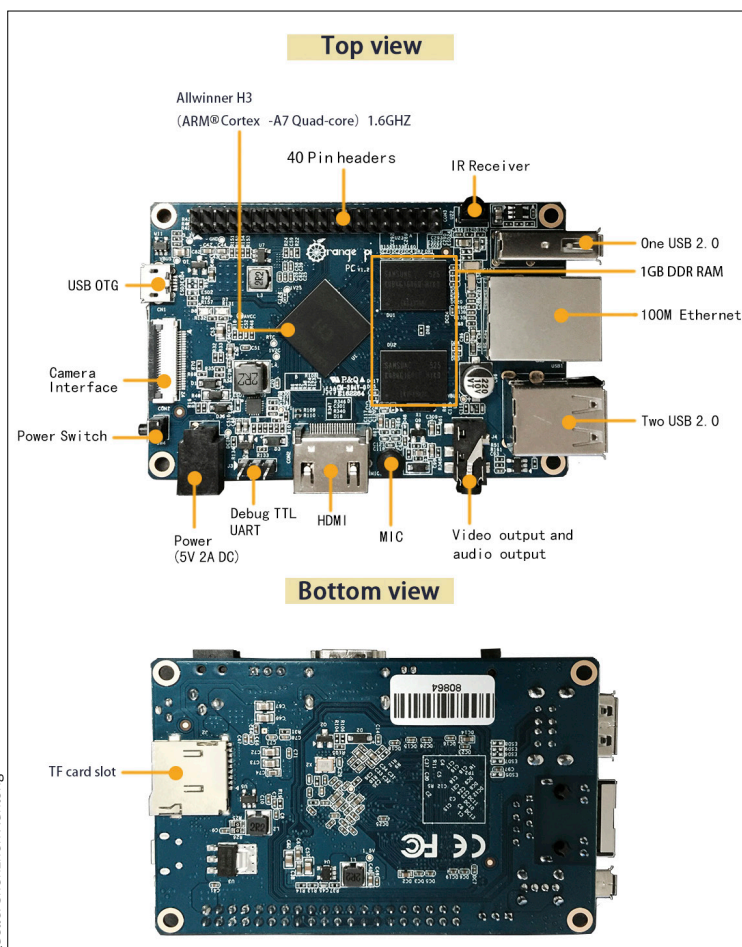
Je nach Hoster dauert das Herunterladen der rund 900 MByte großen Datei bis zu fünf Stunden. Die IMG.XZ-Datei lässt sich unter Ubuntu 14.04 per Doppelklick öffnen. Das Betriebssystem blendet daraufhin den in **Bild 4** gezeigten Deployment-Assistenten ein, in dem Sie die mit dem Image zu bevölkernde Speicherkarte auswählen. Der Deployment-Prozess nimmt einige Zeit in Anspruch.

Speicherkarte

Stecken Sie die Speicherkarte sodann in den auf der Unterseite des Prozessrechners befindlichen Slot und verbinden Sie die Planare mit Maus, Monitor Tastatur, Netzwerk und Stromversorgung. Falls der Prozessrechner darauf nicht mit dem Bootvorgang beginnt, können Sie ihn durch Drücken der Reset-Taste an seine Aufgabe erinnern. Achten Sie darauf, dass der Speicherkartenslot mit der von Smartphones bekannten Mechanik ausgestattet ist: Zum Entfernen der Karte müssen Sie diese in den Rechner hineindrücken.

Das Betriebssystem des Einplatinencomputers nimmt Tastaturen nur dann an, wenn sie schon beim Einschalten verbunden sind. Für das Einloggen nutzen Sie die Passwortkombination `root/orangepi`. Leider zeigt sich der Prozessrechner bei der Tastaturauswahl wählerisch: Häufig werden angeschlossene USB-Tastaturen schlichtweg nicht erkannt.

In diesem Fall müssen Sie sich per SSH einloggen: Entfernen Sie die Speicherkarte aus Ihrem Prozessrechner und editieren Sie die Datei `/etc/network/interfaces` nach folgendem Schema: ►



Orange Pi: Eine preiswerte Konkurrenz zum Raspberry Pi (**Bild 1**)

Quelle: Shenzhen Xunlong

```

auto lo iface lo inet loopback
iface eth0
inet static address 10.42.0.2 netmask
255.255.255.0 broadcast 10.42.0.255
gateway 10.42.0.1 allow-hotplug wlan0
...

```

Die hier beschriebenen Änderungen sorgen dafür, dass die Ethernet-Schnittstelle nach dem Start mit einer statischen IP-Adresse arbeitet. Dies erspart Ihnen das Hantieren mit Router beziehungsweise DHCP-Server.

Im nächsten Schritt können Sie sich per SSH mit dem Orange Pi verbinden. Achten Sie dabei darauf, als Nutzerkonto *root* zu verwenden (Passwort *orangeipi*). Wer sich mit *orangeipi/orangeipi* einloggt, wird mit sofortigem Verbindungsabbruch bestraft.

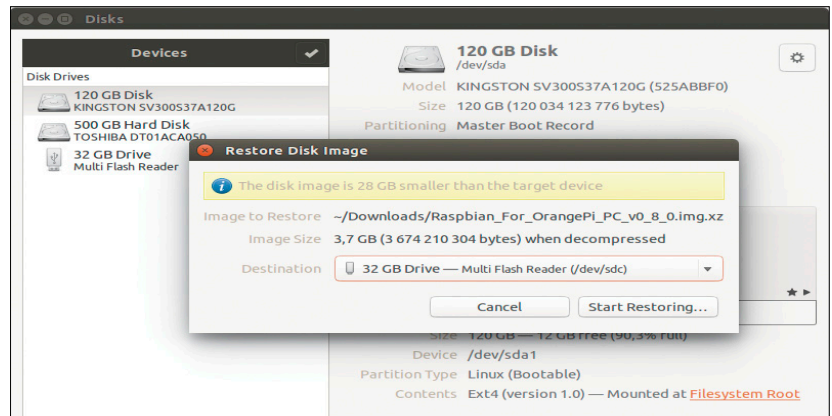
Das *lsusb*-Kommando liefert sodann weitere Informationen über die mit dem Gerät verbundene USB-Peripherie. Wer Tastatur und Maus zwecks Verhinderung von Problemen mit der Stromversorgung über einen aktiven Hub mit dem Orange Pi verbindet, wird mit einer an folgendes Schema angelehnten Ausgabe belohnt:

```

root@orangeipi:~# lsusb
Bus 003 Device 002: ID 1a40:0201 Terminus Technology
Inc. FE 2.1 7-port Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0
root hub
...
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1
root hub
...
Bus 003 Device 005: ID 046a:0023 Cherry GmbH CyMotion
Master Linux Keyboard
Bus 003 Device 004: ID 046d:c062 Logitech, Inc. LS1
Laser Mouse, corded

```

Führen Sie im nächsten Schritt – hierzu ist eine Internetverbindung zwingend erforderlich – die Befehlsfolge *apt-get update / apt-get upgrade* aus, um das Betriebssystem des Pro-



Die Warnung über fehlenden Platz auf der SD-Karte kann ignoriert werden (Bild 4)

zessrechners auf den aktuellsten Stand zu bringen. Ein Reboot lässt sich von der Kommandozeile aus durch Eingabe von *shutdown -r now* bewerkstelligen.

Wenn die Tastatur nach dem obligatorischen Neustart immer noch nicht funktioniert, so sollten Sie eine andere erwerben. Dabei ist es ratsam, den Einplatinencomputer samt Netzgerät in den Laden mitzubringen. Eine funktionierende Tastatur erkennen Sie daran, dass ihre CapsLock-Lampe im Rahmen des Starts zumindest kurz aufleuchtet. Als Alternative bietet sich die Nutzung der NumLock-Taste an, die auch bei hochgefahrenem Betriebssystem funktioniert.

Dieser Artikel entstand unter Verwendung einer LogiTech K120. Tastaturen des deutschen Traditionsherstellers Cherry wurden indes nicht erkannt.

Festplatte expandieren

Nach der Extraktion des Images weist die SD-Karte eine rund 4 GByte große Datenpartition auf. Der große Rest des Speichers liegt brach. Das in Raspbian enthaltene Repartitionierungs-Tool ist mit dem Vergrößern der Partition überfordert, weshalb etwas Handarbeit notwendig ist.

Das von Shenzhen Xunlong ausgelieferte Image enthält eine ungerade Partitionstabelle, die mit *gparted* nicht bearbeitet werden kann. Als Alternative bietet sich das etwas schwieriger zu bedienende *fdisk* an, das im Rahmen des Starts mit

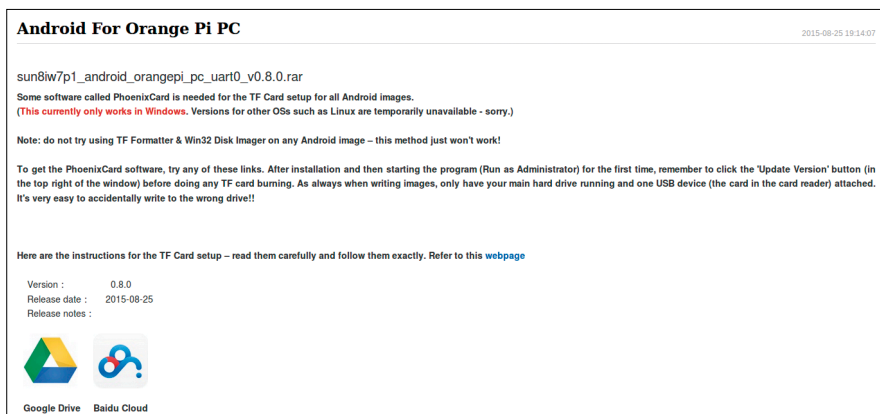
der Geräte-ID des zu verwendenden Festwertspeichers rechnet. Die Eingabe von *p* sorgt sodann dafür, dass die gerade aktuelle Partitionstabelle in die Konsole geschrieben wird:

```

tamhan@TAMHAN14:~/Desktop$ sudo
fdisk /dev/sdd Command (m for
help): p

```

Unser Image hat die SD-Karte mit zwei Partitionen versehen: Neben der kleinen Bootpartition *sdd1* gibt es mit *sdd2* eine größere Datenpartition. Diese wird im nächsten Schritt mittels *d*-Befehl gelöscht:



Google Drive wird nur auf dem Papier unterstützt (Bild 3)

```
Command (m for help): d
Partition number (1-4): 2
```

fdisk beschränkt sich auf die Partitionstabelle und tangiert die eigentlichen Daten nicht. Aus diesem Grund können wir im nächsten Schritt eine neue primäre Partition anlegen.

Achten Sie hierbei darauf, als *First sector* den im vorigen Schritt ausgegebenen Startsektor der alten Datenpartition anzugeben. Die Abfrage nach dem letzten Sektor hingegen wird mit der Return-Taste quittiert, um *fdisk* zur Ausnutzung des gesamten am IS-Medium verfügbaren Speichers zu animieren:

```
Command (m for help): n
Partition type:
p   primary (1 primary, 0 extended, 3 free)
e   extended
Select (default p): p
Partition number (1-4, default 2): 2
First sector (2048-61896703, default 2048): 124928
Last sector, +sectors or +size{K,M,G} (124928-61896703,
default 61896703):
Using default value 61896703
```

Bis jetzt liegen die Änderungen nur im Arbeitsspeicher des Hosts. Das Eingeben von *w* weist *fdisk* dazu an, die Informationen auf die Platte zu schreiben.

Partitionstabelle

Da der Kernel des Hostbetriebssystems seine Partitionstabelle nicht selbsttätig aktualisiert, müssen Sie die SD-Karte an dieser Stelle abstecken und einige Sekunden später wieder mit dem Rechner verbinden.

Achten Sie hierbei darauf, dass sich der Partitionsname mitunter verändert.

Tabelle 1: Benchmark-Ergebnisse (1)

	Orange Pi	Raspberry Pi 2
Number of threads	1	1
Total Time (sec.)	2.225.031	2.966.546
Total Number of Events	10.000	10.000
Total Time Taken by Event Execution (sec.)	2.224.926	2.966.390

Tabelle 1: Benchmark-Ergebnisse (2)

	Orange Pi	Raspberry Pi 2
Number of threads	4	4
Total Time (sec.)	632.279	745.818
Total Number of Events	10.000	10.000
Total Time Taken by Event Execution (sec.)	2.528.518	2.982.414

Zu guter Letzt muss das EXT-Dateisystem über die Vergrößerung seiner unterliegenden Partition informiert werden. Der neue Speicherplatz steht erst dann zur Verfügung, wenn *resize2fs* erfolgreich durchlief.

Entfernen Sie die Speicherkarte sodann per *umount* aus dem Dateisystem der Workstation, um sie wieder in den Orange Pi zu befördern. Der Lohn der Mühen ist eine Raspbian-Installation, die die vollen 32 GByte der Speicherkarte ausnutzt. Zum Installieren größerer Programme ist dies zwingend erforderlich, da das Betriebssystem selbst rund 3,2 GByte der nur 3,6 GByte großen Partition belegt.

Eine Frage des Einschalters

Der in der Einleitung erwähnte Einschaltknopf wirkt überflüssig, da der Orange Pi sofort startet, wenn man ihn mit Strom versorgt. Dieses auf den ersten Blick widersinnige Verfahren ist ein Entgegenkommen an die Prozessrechnerfraktion. Wer den Orange Pi zur Steuerung benutzt, freut sich darüber, wenn er nach einem Stromausfall automatisch neu startet.

Für normale User ist der Knopf immer dann relevant, wenn der Einplatinencomputer ordnungsgemäß durch Eingabe von *shutdown -h now* heruntergefahren wurde. In diesem Fall startet er erst dann neu, wenn der Button gedrückt wird. Zum Wechsel der MicroSD-Karte müssen Sie das Netzgerät also nicht mehr abstecken.

Benchmark-Ergebnisse

Eben Upton setzte im Raspberry Pi 2 auf eine evolutionär weiterentwickelte Variante eines Prozessors, der schon beim Erscheinen der ersten Generation veraltet war.

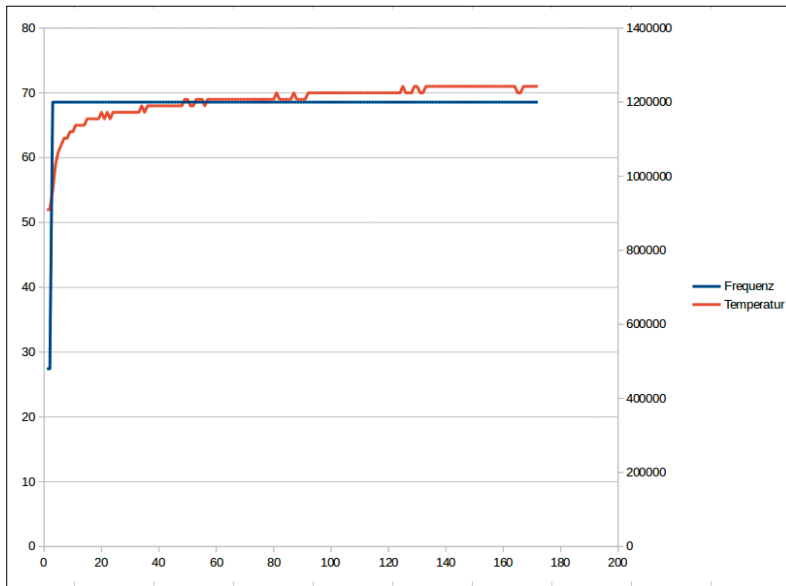
Der AllWinner H3 ist im Vergleich dazu wesentlich leistungsfähiger. Das chinesische Unternehmen konnte seinen Vierkernprozessor auch im ein oder anderen Mittelklasse-Smartphone unterbringen.

Als ersten Geschwindigkeitstest wollen wir den im Raspberry-Pi-Bereich weit verbreiteten Sysbench einspannen. Er lässt sich sowohl auf dem RPi 2 als auch auf dem Orange Pi durch Eingabe von *sudo apt-get install sysbench* herunterladen. Die einkernige Ausführung eines Testlaufs lässt sich durch Eingabe von *sysbench --num-threads=1 --test=cpu run* anwerfen und führt zu den in **Tabelle 1** dargestellten Ergebnissen.

Im Vergleich zu den 296 Sekunden des Raspberry Pi 2 fällt hier eine wesentliche Steigerung der Performance auf: Applikationen, die mit einem einzelnen Thread auskommen müssen, arbeiten am Orange Pi um mehr als ein Drittel schneller.

Im Mehrkernbetrieb – beide Einplatinencomputer nutzen bei Eingabe von *sysbench --num-threads=4 --test=cpu run* alle vier Prozessorkerne aus – ist der Orange Pi rund zehn Sekunden schneller als der Raspberry Pi 2. Die detaillierten Ergebnisse der Testläufe sind in **Tabelle 2** gezeigt. Für uns ist die Frage interessant, warum der AllWinner-SoC seine wesentlich höhere Einkern-Performance in diesem Benchmark nicht umsetzen kann.

Angelsächsische Quellen berichten von thermischen Problemen: Der OrangePi soll sich bei starker Prozessorbela-



Die Temperatur des Prozessors steigt unter Last rapide an (Bild 5)

tung aus Sicherheitsgründen ausbremsen, wenn die Platine nicht mit einem zusätzlichen Kühlkörper ausgestattet wird. Ein Griff reicht aus, um sich von der dort anfallenden Wärme zu überzeugen.

Sensoren des Prozessors

AllWinner erschwert die Analyse dieser Situation insofern, als das Unternehmen keinerlei Informationen über die im Prozessor befindlichen Sensoren herausrückt. Nutzer haben in Eigenregie die drei nachfolgenden Messpunkte ermittelt, die mittels *cat* ihre gerade aktuellen Werte in die Kommandozeile schreiben:

- `/sys/devices/virtual/thermal/thermal_zone0/temp`: Temperatur in einem nicht näher bestimmten Teil des SoC,
- `/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq`: Taktrate des Prozessors,
- `/sys/devices/platform/sunxi-ddrfreq/devfreq/sunxi-ddrfreq/cur_freq`: Taktrate des Arbeitsspeichers.

Ein einfacher Weg zum Testen der Performance ist, den Einplatinencomputer unter Last dazu zu animieren, regelmäßig Informationen zu CPU-Frequenz und Temperatur zu emittieren. Dazu eignet sich folgendes Bash-Skript, das die Ergebnisse der beiden *cat*-Befehle einmal pro Sekunde an zwei Textdateien anhängt:

```
cat /sys/devices/virtual/thermal/thermal_zone0/temp >>
cputemp.txt
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_
freq >> cpufreq.txt
```

Die periodische Abarbeitung der Shell-Datei erfolgt sodann durch eine Befehlskombination.

Chmod markiert die Datei als ausführbar, während *watch* den als Parameter übergebenen Befehl im per *-n* angegebenen Intervall anstößt:

```
root@orange-pi:~/Desktop# chmod +x runtest.sh
root@orange-pi:~/Desktop# watch -n 1
./runtest.sh
```

Zur Generierung der CPU-Last greifen wir auf eine Besonderheit von *sysbench* zurück. Das Produkt kann auf Wunsch mit einer Obergrenze ausgestattet werden: Es werden so lange Primzahlen berechnet, bis die gewünschte Zahl erreicht ist. Wer hier einen ausreichend hohen Wert übergibt, beschäftigt auch leistungsstarke Computer für einige Zeit:

```
sysbench --test=cpu --cpu-max-prime=50000
--num-threads=4 run
```

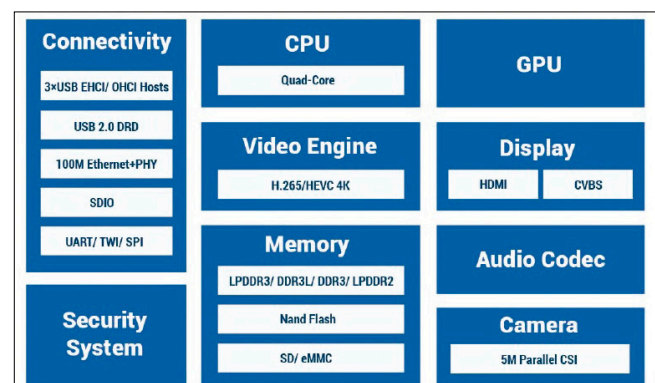
Für die parallele Ausführung von Testskript und Benchmark greifen wir auf eine kleine Besonderheit von Linux zurück. Das Drücken von [Strg Alt F1] bis [Strg Alt F4] erlaubt das Wechseln zwischen vier parallelen Konsolen, die je

einen eigenen Auftrag bearbeiten können. Loggen Sie sich in jedem der beiden Fenster ein und führen Sie die jeweiligen Befehle aus. Transferieren Sie die Daten nach getaner Arbeit auf eine Workstation, um sie mit Excel oder einer ähnlichen Tabellenverarbeitung aufzuarbeiten. Bild 5 zeigt das Ergebnis im Labor des Autors.

Auf AliExpress und bei den einschlägigen Anbietern von Zubehör für Übertakter findet sich eine Vielzahl verschiedener Kühlkörper, die teilweise sogar mit kleinen Ventilatoren ausgestattet sind. Wer nicht aufpasst, zahlt für den Kühlkörper mehr als für den Einplatinencomputer. Aus Sicht des Autors ist diese Investition nur dann sinnvoll, wenn Sie den Orange Pi im Cluster betreiben oder für eine bestimmte Anwendung etwas Zusatzleistung benötigen.

Grafik: Fehlanzeige

Der im Raspberry Pi verbaute VideoCore mag quelloffen sein, wirklich schnell ist die betagte GPU nicht. Da der Nachfolger auf denselben Grafikbeschleuniger setzt, sind die im Internet gern berichteten Performancesteigerungen größtenteils auf die schnellere CPU zurückzuführen. Der AllWinner H3 spielt



Der Netzwerkchip ist ein unabhängiger Teil des SoC (Bild 6)

im Vergleich dazu in einer komplett anderen Klasse, weshalb grafikaffine Benutzer zumindest in der Theorie mit wesentlich besseren Leistungsdaten rechnen dürfen.

Da *glxgears* vom Entwickler explizit nicht als Benchmark zugelassen wurde, nutzen wir stattdessen die quelloffene QuakeIII-Alternativen Nexuiz und OpenArena. Beide lassen sich durch Eingabe von *apt-get install <appname>* herunterladen. Rechnen Sie damit, dass der Download der rund 500 MByte einige Minuten in Anspruch nimmt.

Leider waren beide Produkte nicht zur Nutzung von OpenGL zu animieren: Die Treiberumgebung der zum Zeitpunkt der Drucklegung aktuellen Version von Raspbian für Orange Pi weist massive Probleme im Bereich der OpenGL-Grafikbeschleunigung auf und disqualifiziert den Orange Pi somit für den Einsatz als Mediacenter.

Netzwerkende Spielereien

LAN und USB waren zwei Schwachpunkte des RPi 1. Da der im RPi 2 verwendete Broadcom-Prozessor dasselbe grundlegende Schnittstellendesign mitbrachte, entstanden die Verbesserungen nur durch die gesteigerte Taktrate.

Laut dem in Bild 6 gezeigten Blockschaltbild verfügt der Allwinner-Prozessor des Orange Pi über einen dedizierten Netzwerchip, der sich die Bandbreite nicht mit dem USB-Bus teilen muss.

Wie im Fall des Raspberry Pi 2 beginnen wir auch hier mit einem Test per *iperf*. Als Gegenstelle diene hierbei eine AMD-Workstation mit einem Gigabit-Ethernet-Port, die direkt mit dem zu testenden Prozessrechner verbunden wurde. Die Eingabe von *iperf -s* sorgt auf Serverseite dafür, dass die Arbeitsstation eingehende Pakete entgegennimmt und weiterleitet.

Am Prozessrechner folgt im nächsten Schritt das Anwerfen des Testclients. Achten Sie darauf, die IP-Adresse an die Gegebenheiten Ihres lokalen Netzwerks anzupassen. Im Vergleich zum Raspberry Pi 2 fällt hier eine Geschwindigkeitssteigerung um rund 2 MBit/sec auf. Diese an sich im Rahmen der Messgenauigkeit liegende Differenz blieb allerdings auch nach mehreren Reboots und in längeren Testläufen konstant.

Als nächste Aufgabe wenden wir uns einem USB-Stick zu, der in den alleinstehenden USB-Steckplatz neben der Netzbuchse wanderte. Raspbian erkennt das Speichermedium nicht automatisch. Vor dem Zugriff muss ein Mountpunkt angelegt werden. Das Einhängen der betreffenden Partition erfolgt sodann manuell per *mount*-Kommando:

```
root@orangepi:/dev# mkdir /media/usbstick
root@orangepi:/dev# mount /dev/sda1 /media/usbstick
```

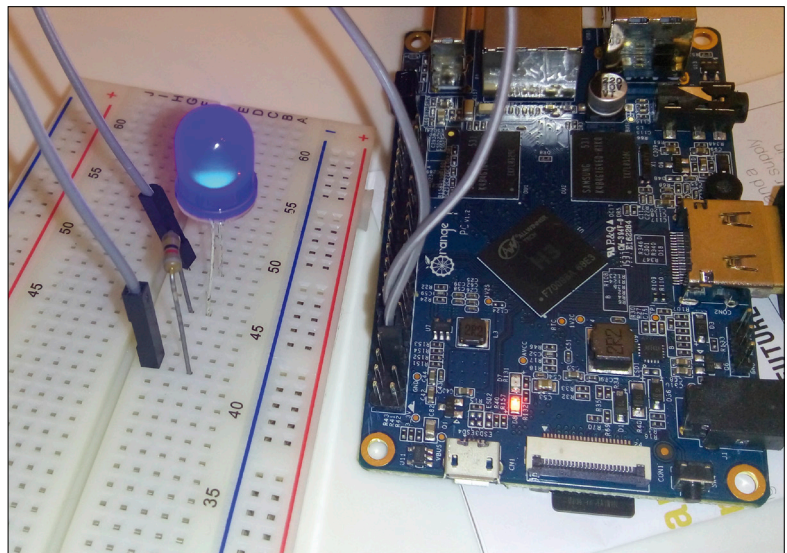
Ein kurzer Lesetest mit *hdparm* ergab die folgenden Werte:

```
root@orangepi:/dev# hdparm -t /dev/sda
/dev/sda:
```

```
Timing buffered disk reads: 36 MB in 3.14 seconds =
11.46 MB/sec
root@orangepi:/dev#
```

Shenzhen Xunlong empfiehlt in der Dokumentation für den Orange Pi explizit die Nutzung eines Netzgeräts, das zwei Ampere liefern muss. Damit sollte der Prozessrechner eine kleine USB-Festplatte mit Energie versorgen können. Mit einem vergleichsweise stromsparenden Modell aus dem Hause LaCie funktionierte dies sogar ohne Y-Kabel problemlos. Eine USB-3.0-Festplatte aus dem Hause Western Digital versagte hingegen den Dienst.

Im Kollisionstest wurden der Netzwerctest und der Festplattenbenchmark parallel ausgeführt. Ähnlich wie beim Raspberry Pi 2 kam es auch hier zu keinerlei Performance-

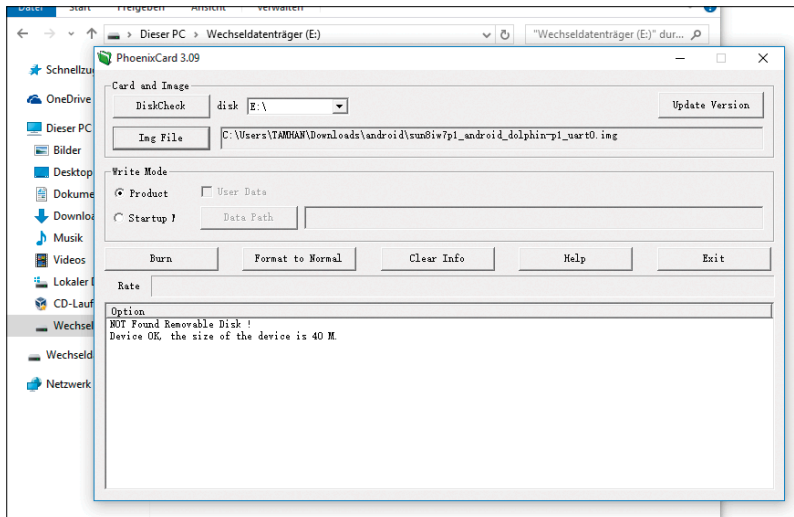


LED: Mit einer kleinen LED machen Tests mehr Freude (Bild 7)

einbußen. Interessant ist, dass die Netzwerkbandbreite des Orange Pi auch im unbelasteten Zustand permanent schwankt, wie das folgende Protokoll zeigt:

```
[ 3] 89.0-90.0 sec 12.0 MBytes 101 Mbits/sec
[ 3] 90.0-91.0 sec 11.4 MBytes 95.4 Mbits/sec
[ 3] 91.0-92.0 sec 11.5 MBytes 96.5 Mbits/sec
[ 3] 92.0-93.0 sec 11.1 MBytes 93.3 Mbits/sec
[ 3] 93.0-94.0 sec 11.0 MBytes 92.3 Mbits/sec
[ 3] 94.0-95.0 sec 11.0 MBytes 92.3 Mbits/sec
[ 3] 95.0-96.0 sec 11.1 MBytes 93.3 Mbits/sec
[ 3] 96.0-97.0 sec 11.1 MBytes 93.3 Mbits/sec
[ 3] 97.0-98.0 sec 11.0 MBytes 92.3 Mbits/sec
[ 3] 98.0-99.0 sec 11.0 MBytes 92.3 Mbits/sec
[ 3] 99.0-100.0 sec 12.4 MBytes 104 Mbits/sec
```

Der Orange Pi wird von seinem Anbieter als mit dem Raspberry Pi kompatibel beschrieben. Er unterscheidet sich schon auf den ersten Blick durch seine 40-Pin-Buchsenleiste von anderen Produkten aus dem Hause Shenzhen Xunlong. ►



PhoenixCard ist eine Spezialanwendung zur Erzeugung bootfähiger Android-Images (Bild 8)

Angesichts der restriktiven Informationspolitik von AllWinner ist es nicht einfach, die diversen GPIO-Bibliotheken auf den Orange Pi zu portieren. Zhao Lei hat die Wiring-Bibliothek auf den Orange Pi portiert. Die Installation erfolgt so:

```
root@orangePi:~# mkdir orangepiwiring
root@orangePi:~# cd orangepiwiring/
root@orangePi:~/orangepiwiring# git clone
https://github.com/zhaoLei/WiringOP.git -b h3
root@orangePi:~/orangepiwiring# cd WiringOP/
root@orangePi:~/orangepiwiring/WiringOP# chmod +x
./build
root@orangePi:~/orangepiwiring/WiringOP# sudo ./build
```

Ein nützliches Werkzeug ist *gpio readall*. Es gibt eine Portmap aus, die die unter <https://drive.google.com/file/d/0BwQT0FP7IkTbYmFleURSrkFNdlE/view> bereitstehenden Informationen ergänzt. Nun kann eine einfache Schaltung realisiert werden, die eine LED zum Blinken bringt. Dazu wird diese zwischen den Pins drei und sechs geschaltet. Im nächsten Schritt ist eine *.c*-Datei notwendig:

```
#include <stdio.h>
#include <wiringPi.h>
#define LED 6

int main (void) {
    wiringPiSetup ();
    pinMode (LED, OUTPUT);
    for (;;) {
        digitalWrite (LED, HIGH); // On
        delay (500);               // mS
        digitalWrite (LED, LOW);  // Off
        delay (500);
    }
    return 0; }
```

Nach dem Einbinden der beiden für die Programmausführung notwendigen Header deklarieren wir die ID des zu ver-

wendenden Pins. Der dabei zu verwendende Integer lässt sich in *gpio readall* aus der Spalte *wPi* ablesen: Ist ein Pin mit der ID -1 versehen, so lässt er sich über diese Bibliothek nicht ansprechen.

main() erinnert vom Aufbau her stark an Prozessrechenprogramme für Massimo Banzis Arduino-Plattform. Die Wiring-Implementierung für ARM-basierte Einplatinencomputer unterscheidet sich insofern vom großen Vorbild, als die Methode *wiringPiSetup* im Rahmen der Initialisierung aufgerufen werden muss.

Die eigentliche Blinklogik ist dann in einer gewöhnlichen Endlosschleife untergebracht, die den Pin unter Nutzung der *digitalWrite*-Funktion regelmäßig ein- und ausschaltet:

Für die Kompilation ist die *wiringPi*-Bibliothek zwingend erforderlich, die sich durch folgendes Kommando an GCC weiterreichen lässt:

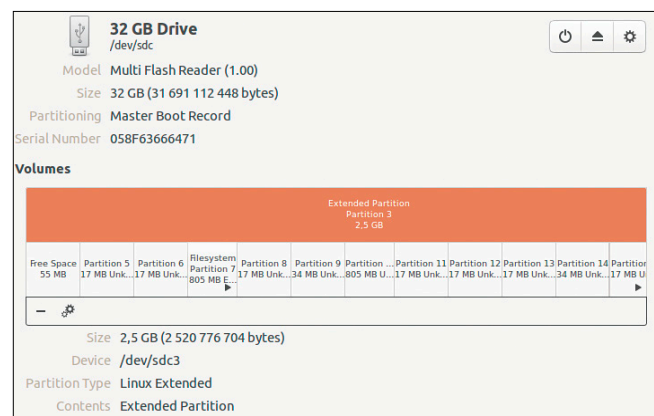
```
root@orangePi:~/orangepiwiring# gcc blink.cpp -lwiringPi
```

Führen Sie das Programm sodann durch Eingabe von *./a.out* aus, um die LED – wie in Bild 7 gezeigt – zum Blinken zu bringen.

Android für Orange Pi

Android für Raspberry Pi ist eine lange und leidige Geschichte. Für die erste Version wurde das Betriebssystem zwar von der Raspberry-Pi-Foundation versprochen, aber nie ausgeliefert.

Auf der offiziellen Download- und Ressourcenseite der Orange-Pi-Serie finden sich Android-Images. Die für die folgenden Schritte verwendete Variante wurde unter <https://www.xgadget.de/app-software/orange-pi-pc-image-download> heruntergeladen. Die Datei *sun8iw7p1_android_dolphin-p1_uart0.img* lässt sich nicht per GUI auf die Speicherkarte schreiben. Das Deployment der IMG-Datei muss mit einem nur unter Windows lauffähigen Spezialprogramm namens PhoenixCard erfolgen. Laden Sie es im ersten Schritt



Android besteht aus mehreren Partitionen (Bild 9)

unter www.haoyuelectronics.com/service/A10-A20/tools herunter, und extrahieren Sie die Datei an einen beliebigen Ort. Führen Sie im nächsten Schritt *PhoenixCard.exe* aus und konfigurieren Sie das Programm wie in **Bild 8** gezeigt.

Führen Sie im ersten Schritt mit aktivierter Option *Product* das Kommando *Format to Normal* aus, um die Speicherkarte in einen definierten Zustand zu bringen. Wählen Sie danach die Option *StartUp* aus und aktivieren Sie den eigentlichen Schreibprozess durch Anklicken von *Burn*. Nach der Anzeige von *Burn End* kann Windows heruntergefahren werden. Die Disks-Anwendung von Ubuntu stellt eine erfolgreich gebrannte Android-Speicherkarte wie in **Bild 9** gezeigt dar.

Der eigentliche Start des auf Android 4.4.2 basierenden Images nimmt einige Minuten in Anspruch. In den ersten 30 Sekunden des Bootvorgangs ist am Bildschirm nichts zu sehen. Nach dem erfolgreichen Start präsentiert die Android-Variante ein chinesisches Benutzerinterface, in dem sich der Play Store über das in **Bild 10** gezeigte Menü aufrufen lässt.

Wer – wie der Autor dieser Zeilen – des Chinesischen nicht mächtig ist, wechselt stattdessen in die Einstellungen. Das Anklicken des obersten Eintrags in der auf der rechten Seite eingeblendeten Optionsliste erlaubt das Öffnen des Sprachauswahldialogs, der sowohl Deutsch als auch Englisch anbietet (**Bild 11**).

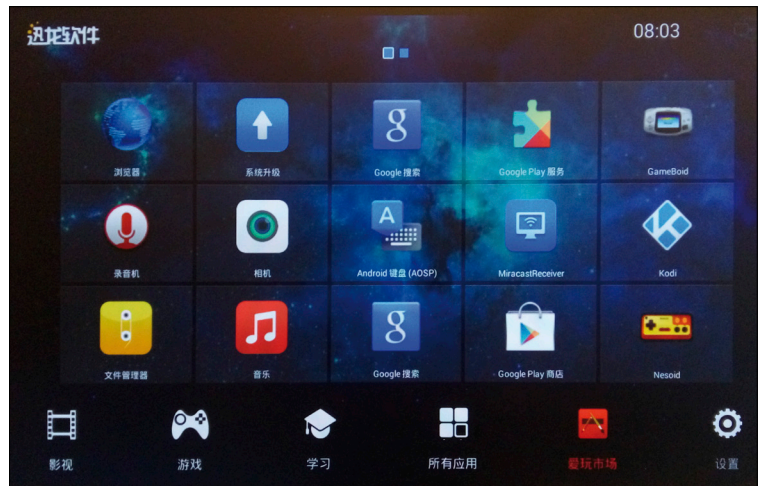
Aus technischer Sicht ist die Android-Portierung das stabilste der drei für den Orange Pi angebotenen Betriebssysteme: Die Grafikbeschleunigung funktioniert, das Benutzer-Interface wird ohne jegliches Ruckeln gerendert. Leider lässt sich Ethernet im uns vorliegenden Build nicht aktivieren. Die als Gegenstelle dienende Workstation erkennt auch dann keinen Zugriff von Seiten des Einplatinencomputers, wenn Ethernet im Menü explizit aktiviert wird.

Daraus ergibt sich eine wesentliche Einschränkung: Wer mit dem Orange Pi unter Android ins Internet möchte, muss kompatible Netzwerkhardware in Form eines USB-Sticks anschließen. Als Alternative dazu berichten einige Nutzer unter www.orangepi.org/orangepibbsen/forum.php?mod=viewthread&tid=514 von Erfolgen mit sehr schnellen Speicherkarten.

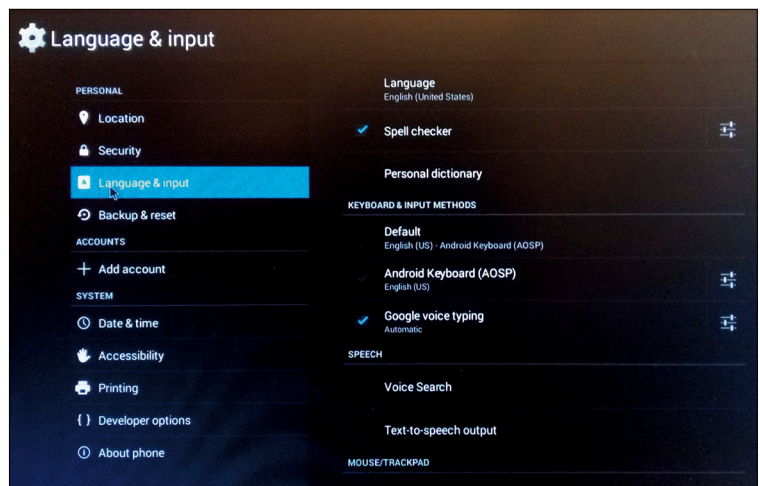
Ubuntu für jedermann

Am Raspberry Pi 2 unterstützt die Foundation nur eine Schmalpurversion der weit verbreiteten Linux-Distribution. Das Image von 14.04 ist ein von der Community gepflegter Fork. Beim Orange Pi hingegen sieht die Situation insofern besser aus, als Shenzhen Xunlong eine Variante von Lubuntu anbietet.

Die IMG-Datei *Lubuntu_1404_For_OrangePiPC_v0_8_0_0.img* ist eine normale Imagedatei, die sich unter Unix aus der Kommandozeile heraus auf eine Speicherkarte brennen lässt. Die dazu notwendige Eruiierung des Mountpunkts erfolgt



Der Play Store ist gut versteckt (**Bild 10**)



Sprache: Wer sich durch die Symbole hangelt, wandelt das chinesische Betriebssystem mit minimalem Aufwand in ein englisches um (**Bild 11**)

durch Eingabe von *df -h*. Falls Sie die vorher für Raspbian verwendete SD-Karte hier weiterverwenden, sollte die Ausgabe analog zum im Listing Gezeigten ausfallen:

```
tamhan@TAMHAN14:~/Desktop$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       110G  102G   3,3G   97% /
...
/dev/sdb5       202G  175G   17G   92% /media/tamhan/dc8d-
c1e2-72ee-4a07-bc80-2dfabb108545
/dev/sdb1       243G  166G   78G   69% /media/tamhan/windrive
/dev/sdc1        41M   25M   17M   60% /media/tamhan/5A69-D2F4
/dev/sdc2        29G   5,5G   23G   20% /media/tamhan/5d18be51-
3217-4679-9c72-a54e0fc53d6b
```

```
tamhan@TAMHAN14:~/Desktop$
```

Unsere SD-Karte liegt auf */dev/sdc*. Dies ist unschwer daran zu erkennen, dass nur dieses Gerät die kleine Boot- und ►

Links zum Thema

- Downloadseite von Shenzhen Zhulong
www.orange-pi.org/downloadresources
- Die im Artikel verwendeten Images
www.tamoggemon.com/test/NMGOOrangePi/index.htm
- Android-Images
<https://www.xgadget.de/app-software/orange-pi-pc-image-download>

die große Datenpartition aufweist. Mit diesem Wissen können wir uns an das eigentliche Deployment des Images heranwagen, das wie gewohnt per *dd* erfolgt:

```
tamhan@TAMHAN14:~/Desktop/NMGOOrangePi$ sudo dd bs=4M
if=Lubuntu_1404_For_OrangePiPC_v0_8_0_.img of=/dev/sdc

[sudo] password for tamhan:
136+1 records in
136+1 records out
570442752 bytes (570 MB) copied, 595,569 s, 958 kB/s
```

if ist hierbei die Datei, aus der die zu schreibenden Informationen stammen. Das Zielmedium wird durch den Parameter *of* beschrieben. Achten Sie darauf, hier nicht versehentlich eine wichtige Partition der Host-Workstation anzugeben.

Nach dem Einstecken der Speicherkarte startet der Orange Pi automatisch Ubuntu. Als Benutzername und Passwort dienen hier *orange-pi/orange-pi*. Nach dem Anmelden fällt das Vorhandensein eines modernen Browsers auf: Nutzer von Lubuntu können das Netz mit Chromium durchforsten und müssen sich nicht mit Midori ablagen.

Leider ist das System im praktischen Einsatz nicht sonderlich befriedigend. Wer ein YouTube-Video öffnet, erlebt auch bei der niedrigstmöglichen Auflösung eine wahre Ruckelorgie. Auch unter Ubuntu scheinen die Betriebssystementwickler den Grafikchip des Einplatinencomputers nicht wirklich unter Kontrolle zu haben.

Sowohl OpenArena als auch Nexuiz lassen sich unter Ubuntu problemlos installieren und starten auch – leider führt die fehlende Hardwarebeschleunigung auch unter diesem Betriebssystem zu immensen Ruckeln, wodurch die Bedienung des Spiels so gut wie unmöglich wird.

Grundlagen des Clusterings

Ob des im vorigen Abschnitt thematisierten Fehlens von Grafikbeschleunigung taugt der Orange Pi nur leidlich als Desktopersatz oder Mediencenter. Während das Abspielen von Musik keine großen Probleme macht, artet die Bedienung des Desktops stellenweise in ein wahres Geduldsspiel aus.

Dank der sehr leistungsfähigen CPU und des geringen Preises dürfte der Orange Pi insbesondere bei Betreibern von Clustern aus Raspberry Pis Begehrlichkeiten wecken. Die neue Planare ist größtenteils ähnlich, bietet aber rund 15

Prozent mehr Rechenleistung zu einem wesentlich geringeren Preis.

Aus Einplatinencomputern zusammengesetzte Cluster mögen auf den ersten Blick witzig beziehungsweise sinnfrei erscheinen, funktionieren in der Praxis aber bestens. Das Unternehmen des Autors betrieb seine geografisch verteilten Raspberry Pis fast zwei Jahre lang mit immensem Erfolg. Auf der Habenseite steht der geringe Platzbedarf und das einfache Deployment.

Die wichtigste Regel beim Clustering ist die Minimierung der Internodalkommunikation: Wenn die Einplatinencomputer über ihre vergleichsweise bandbreitenarmen Ethernet-Anschlüsse Daten austauschen müssen, so ist das Erreichen einer attraktiven Gesamtleistung unmöglich.

Als Ideallösung präsentiert sich an dieser Stelle die Nutzung von zustandslosen Protokollen und der Verzicht von Datenhaltung am Server. Sind die Requests in sich geschlossen, so können sie vom jeweiligen EPC ohne weitere Kommunikation bewerkstelligt werden. Ist dies nicht möglich, so bietet sich alternativ die dynamische Zuweisung an: Will ein Client arbeiten lassen, so sucht er sich einen Server aus. Dieser lädt die im Zentralspeicher befindlichen Daten in einen lokalen Cache, wo er sie sodann für weitere Berechnungen einspannt.

Für die eigentliche Lastverteilung bietet sich die Nutzung eines numerisch stabilen Zufallsgenerators an. Verschiedenen leistungsstarken Clients lassen sich durch das Zuweisen von verschiedenen breiten Teilen des Zahlenstrahls gerecht in den Gesamtprozess einbinden.

Wenn mehrere Orange Pi in einem beengten Raum arbeiten sollen, empfiehlt sich die Investition in Kühlkörper auf jeden Fall. Kleine Aluminiumrippen und ein sorgfältig geplanter Luftweg tragen in diesem Fall zu einer wesentlichen Verlängerung der Gesamtlebensdauer bei.

Fazit

Produkte von Shenzhen Xunlong weisen im Vergleich zu den Angeboten der gut etablierten Raspberry Pi Foundation die ein oder andere Kinderkrankheit auf. Wer nicht basteln möchte und 3D-Beschleunigung benötigt, ist beim Raspberry Pi 2 besser aufgehoben. Das Fehlen der Grafikbeschleunigung disqualifiziert den chinesischen Einplatinencomputer für Multimedia- und Desktopanwendungen.

Der Orange Pi besticht im Vergleich zu seinem etablierten Konkurrenten durch höhere CPU-Rechenleistung und einen geringeren Preis. ■



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

Impressum

Verlag

Neue Mediengesellschaft Ulm mbH
Bayerstraße 16a,
80335 München
Telefon: (089) 74117-0,
Fax: (089) 74117-101
(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,
Sebastian Eschweiler, Jens Geyer,
Thomas Hafen, Tam Hanna,
Johannes Hoppe, Anna Kobylinska,
Bernhard Lauer, Patrick Lobacher,
Filipe Martins, Florence Maurice,
Stefan Moch, Michael Rohrich,
Markus Schraudolph,
Katharina Sckommodau,
Thomas Sillmann, Gregor Woiwode

Art Directorin

Maria-Luise Sailer

Grafik & Bildredaktion

Alfred Agatz, Dagmar Breitenbauch,
Catharina Burmester, Hedi Hefele,
Manuela Keller, Simone Köhnke,
Cornelia Pflanzner, Petra Reichensperner,
Ilka Rüther, Christian Schumacher,
Nicole Üblacker, Mathias Vietmeier

Anzeigenberatung

Jens Schmidtman, Anzeigenleiter
Klaus Ahlering, Senior Sales Manager
Telefon: (089) 74117-125
Fax: (089) 74117-269
E-Mail Anzeigenberatung: sales@nmg.de

Anzeigendisposition

Dr. Jürgen Bossmann
Telefon: (089) 74117-281
Fax: (089) 74117-269
E-Mail: sales@nmg.de

Leitung Herstellung/Vertrieb

Thomas Heydn
Telefon: (089) 74117-111
E-Mail: thomas.heydn@nmg.de

Leserservice

Hotline: (089) 74117-205
Fax: (089) 74117-101
E-Mail: leserservice@nmg.de

Kooperationen

Denis Motzko
Telefon: (089) 74117-116
E-Mail: kooperationen@nmg.de

Druck

L.N. Schaffrath Druckmedien
Marktweg 42-50
47608 Geldern

CD-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.

Veröffentlichung gemäß**Art. 8 Abs. 3 BayPrG:**

Alleiniger Gesellschafter der Neuen Me-
diengesellschaft Ulm mbH ist die Neue
Medien Ulm Holding GmbH, die wiederum
eine 100-Prozent-Tochter der Verlags-
gruppe Ebner Ulm GmbH & Co. KG ist.

ISSN: 2194-4105

© 2016 Neue Mediengesellschaft Ulm mbH

**Jetzt Ihre
web & mobile developer
auf dem iPad lesen**



**Jetzt online
weiterbilden!**

„Moderne Probleme
fordern modernes
Wissen. Mit Webinaren
bleibt man auf
dem neuesten Stand.“

Johannes Hofmeister
Softwareentwickler,
Psychologe, Sprecher



developer-media.de/webinare

MOBILITY-PLATTFORMEN

Mit Mobility-Plattformen ins mobile Zeitalter

Wie die digitale Transformation die Entwicklung von Enterprise-Software verändern wird.

Die digitale Transformation stellt Unternehmen vor vielfältige Herausforderungen. Eine davon ist das Thema Enterprise Mobility. In der Geschäftswelt der Zukunft nutzen Mitarbeiter stationäre Desktop-PCs und Notebooks nur noch im Notfall. Stattdessen verwenden sie mobile Devices: Smartphones, Tablets und Wearables wie etwa Smartwatches und Smartglasses (Bild 1).

Über diese greifen sie auf mobile Applikationen zu, mit denen sie verschiedenste Aufgaben erledigen: Ein angestellter Servicetechniker gibt von unterwegs aus seine Anfahrts- und Arbeitszeiten in eine Smartphone-App ein oder ein Servicemitarbeiter im Bekleidungshandel macht einen Kunden per Tablet auf einen alternativen Artikel aufmerksam, der in einer anderen Filiale oder im Online-Shop verfügbar ist. Diese Entwicklung verursacht einen tiefgreifenden Wandel in der IT-Landschaft von Unternehmen: Sie müssen in der Lage sein, je nach Bedarf in kurzer Zeit mobile Apps zu entwickeln – für unterschiedlichste Endgeräte und Betriebssysteme. Mobility-Plattformen machen dies möglich.

Mobile Endgeräte zu benutzen gehört für die Mehrzahl der deutschen Verbraucher zum Alltag. Laut einer aktuellen Studie des Branchenverbands Bitkom gibt es hierzulande 44 Millionen Smartphone-Nutzer. In deutschen Unternehmen hingegen ist das Thema Mobility längst noch nicht so fest verankert wie im Consumer-Bereich. Mit dem Smartphone Firmen-E-Mails checken oder mit dem Tablet an einer Telefonkonferenz teilnehmen?

Für die meisten Arbeitnehmer stellen solche Aktivitäten eher die Ausnahme als die Regel dar – noch. In den nächsten Jahren wird sich dies grundlegend ändern. Menschen, die in ihrer Freizeit nicht mehr auf Smartphone und Co. verzichten, möchten zunehmend auch bei der Arbeit von den Vorteilen mobiler Endgeräte profitieren. Die aktuelle Studie »Growing The Digital Business« des Beratungs- und Technologiedienst-



Foto: Samsung

Mitarbeiter in Unternehmen nutzen immer häufiger Mobile Devices für ihre Arbeit (Bild 1)

leisters Accenture hat ergeben: Branchenübergreifend sind mindestens 80 Prozent der Entscheider der Meinung, dass mobile Apps notwendig sind, um digitale Technologie gewinnbringend zu nutzen. Ein Großteil der Unternehmen in Deutschland setzt jedoch derzeit noch kaum mobile Apps ein.

Wachsende Vielfalt der Endgeräte

Für Unternehmen ist dies nicht zuletzt deshalb eine enorme Herausforderung, weil der Markt für Mobilgeräte einer fortschreitenden Fragmentierung unterworfen ist. Das beginnt bei den Gerätegattungen: Neben Laptop, Tablet und Smartphone gibt es inzwischen auch hybride Gattungen wie die sogenannten Phablets oder Convertibles, außerdem sind Wearables stark im Kommen. Je nach Hersteller und Modell unterscheiden sich die Geräte in ihren Formfaktoren, etwa der Bildschirmgröße. Auch bei den Betriebssystemen hat eine Fragmentierung stattgefunden. In der Ära der Desktop-PCs benötigten Unternehmen entweder Windows- oder OS-X-Server. Bei Mobilgeräten dominieren mit Android, iOS und Windows derzeit drei Betriebssysteme den Markt. Hinzu kommen eine Handvoll kleinere. Multipliziert man die Merkmale Gerätegattung, Formfaktoren und Betriebssystem, wird das Ausmaß der Fragmentierung erst richtig deutlich.

Was bedeutet das nun für Unternehmen, die ihren Mitarbeitern mobiles Arbeiten ermöglichen möchten – und zwar in aller Konsequenz, sodass Mitarbeiter mit ihrem Mobilgerät auf sämtliche relevanten Systeme jederzeit zugreifen und somit alle Aufgaben erledigen können, die bislang auf den stationären Desktop-Arbeitsplatz beschränkt sind?

Natürlich ist es möglich, dass Unternehmen die Hardware selbst bereitstellen, um so eine einheitliche mobile Infrastruktur zu schaffen. Alle Mitarbeiter mit einem oder gar mehreren Mobilgeräten auszustatten ist jedoch nicht nur kostspielig, sondern aus Mitarbeitersicht auch umständlich. Schließlich verfügen die meisten bereits privat über ein Smartphone. Und wer möchte schon zwei Smartphones bei sich tragen, ein privates und ein geschäftliches?

Die Alternative lautet »Bring your own device« (BYOD) – Mitarbeiter nutzen ihre eigenen Mobilgeräte sowohl privat als auch im Beruf. Neben der Frage, mit welchen Maßnahmen sich im Rahmen einer BYOD-Strategie die Sicherheit der sensiblen Unternehmensdaten gewährleisten lässt, bringt diese Variante insbesondere folgende Herausforderung mit sich: Die oben beschriebene Vielfalt der Endgeräte, die bislang nur im Konsumentenbereich existierte, hält nun auch in Unternehmen Einzug.

Mitarbeiter werden zu Konsumenten

Dazu kommt ein weiteres Phänomen, das man häufig als Consumerization bezeichnet. In Bezug auf die Business-Nutzung von Mobilgeräten stellen Mitarbeiter zunehmend dieselben hohen Anforderungen an Soft- und Hardware wie Verbraucher. Im Mittelpunkt steht dabei die User Experience. Dieser Trend des Experience First wird das allgemeine Verständnis von Enterprise-Anwendungen von Grund auf verändern. Die Mitarbeiter der Zukunft möchten sich mit ihren Mobilgeräten nicht durch sperrige Unternehmensportale quälen, deren Funktionen auf bestimmte Geschäftsprozesse als solche ausgerichtet sind, aber nicht auf die Bedürfnisse der User. Ein Enterprise-System mag beispielsweise den gesamten Serviceprozess von der Bestellung bis zur Lieferung abdecken, aber für Mitarbeiter dennoch unkomfortabel zu bedienen sein. Diese erwarten von einer mobilen Enterprise-Anwendung, dass diese genau jene Funktionen bietet, die sie für eine bestimmte Aufgabe benötigen. Dazu muss sie nicht zuletzt über entsprechende Schnittstellen zu anderen Systemen verfügen. Die Entwicklung in Richtung mobiles Arbeiten und der steigende Anspruch an die User Experience implizieren also: Unternehmen werden einen stark wachsenden Bedarf an intuitiv zu bedienenden mobilen Enterprise-Apps haben, die nicht so sehr umfangreiche Geschäftsprozesse abbilden als vielmehr einzelne Aufgaben von bestimmten Mitarbeitern.

Enterprise-Mobility-Plattformen

Doch wie können Unternehmen diesem Trend in Sachen Software-Entwicklung nachkommen – angesichts der zunehmenden Fragmentierung der IT-Landschaft? Stellt sich zunächst die Frage nach der Art der mobilen Applikationen: native Apps oder Web-Apps? Native Apps werden speziell für das

jeweilige Endgerät entwickelt und auf diesem installiert; mindestens einmal im Jahr benötigen sie ein umfangreiches Update. Unternehmen müssen die App also regelmäßig weiterentwickeln lassen, und zwar in vielfacher Ausführung für die unterschiedlichen Endgeräte und Betriebssysteme. Web-Apps hingegen, die endgeräteunabhängig ohne Installation über den Webbrowser laufen, eignen sich kaum für mobiles Arbeiten, da Nutzer auf eine gute Internetverbindung angewiesen wären.

Doch es gibt eine Lösung, wie Unternehmen eine große Anzahl mobiler Apps, seien es native, webbasierte oder hybride Anwendungen, innerhalb kurzer Zeit und zu überschaubaren Kosten entwickeln und pflegen können: durch Enterprise-Mobility-Plattformen, die eine Cloud-basierte Verbindung zwischen Backend und Frontend besitzen. Als Middleware bietet diese die Möglichkeit, sämtliche mobilen Apps eines Unternehmens auf einer gemeinsamen Code-Basis zu entwickeln. Per Cross-Kompilierung lassen sich die Apps an verschiedenste Formate und Betriebssysteme anpassen und über den gesamten Lebenszyklus hinweg warten. Alle nativen Anwendungen, die über die Cloud-basierte Mobility-Plattform laufen, können Anwender problemlos auch offline nutzen. Sobald der Mitarbeiter wieder eine Internetverbindung hat, erfolgt die Synchronisation mit der Plattform und über diese mit dem IT-Backend.

Verkürzte Entwicklungszyklen

Eine Mobility-Plattform ermöglicht höchste Agilität bei der Entwicklung neuer Enterprise-Apps. Für ein Speditionsunternehmen etwa, das seine Fahrer mit einer Telematik-App zur Übermittlung von Fahrdaten an die Disponenten ausstatten möchte, hätte ein solches Vorhaben noch vor wenigen Jahren ein Entwicklungsprojekt von mehreren Monaten bedeutet. Mit einer Mobility-Plattform hingegen lässt sich heute innerhalb von 20 Tagen ein Prototyp entwickeln, der dann bereits für einen begrenzten Nutzerkreis zum Beta-Test freigeschaltet werden kann.

Nach dem ersten Nutzerfeedback werden Fehler behoben und gegebenenfalls Design und Features optimiert. Selbst nach Go-live der fertigen App sind IT-Abteilung und IT-Dienstleister in der Lage, Bugfixes und Anpassungen aufgrund eines veränderten Bedarfs jederzeit vorzunehmen. Indem Mobility-Plattformen dazu beitragen, die Entwicklungszyklen von Enterprise-Apps enorm zu verkürzen, werden sie zukünftig als wichtiger Katalysator der digitalen Transformation in Erscheinung treten. ■



Stefan Moch

ist Head of Mobile Solutions bei Arvato Systems. Er ist Referent und Autor mehrerer Fachbücher und Fachartikel.

www.arvato-systems.de

NEUE GESETZE UND RICHTSSENTScheidungen DES JAHRES 2015

Rückschau 2015

Kein Jahr vergeht ohne neue Gesetze und Gerichtsentscheidungen – so auch 2015.

Normalerweise treffen juristische Neuerungen in erster Linie die Betreiber von Online-Shops, da der Bereich des E-Commerce naturgemäß viele unterschiedliche Teilrechtsgebiete umfasst. Aber 2015 war ein Jahr, in dem so ziemlich jeder Website-Betreiber, Webdesigner und Webprogrammierer betroffen war.

IT-Sicherheit

Diesbezüglich ist etwa die Einführung des IT-Sicherheitsgesetzes zu nennen. Mit dieser Novelle wurde unter anderem der § 13 Abs. 7 TMG eingeführt, der sich alle Betreiber einer nicht nur rein privaten Internetseite in die Pflicht nimmt. Und zwar bezieht sich diese neue Vorschrift ganz konkret auf Pflichten in puncto Datenschutz und IT-Sicherheit. Denn nach Maßgabe des Gesetzgebers muss sichergestellt werden, dass kein unerlaubter Zugriff auf die für die Website genutzten technischen Einrichtungen möglich ist und diese gegen Verletzungen des Schutzes personenbezogener Daten und gegen Störungen gesichert sind.

Die jeweiligen Vorkehrungen müssen dem Stand der Technik entsprechen, also stets aktuell gehalten werden. Insbesondere weist die Norm auf den Einsatz eines als sicher anerkannten Verschlüsselungsverfahrens hin. Die geforderten technischen Schutzmaßnahmen müssen allerdings technisch möglich und wirtschaftlich zumutbar sein.

Schon kurz nach Einführung des IT-Sicherheitsgesetzes hat das Bayerische Landesamt für Datenschutzaufsicht angekündigt, Internetangebote dahingehend zu prüfen. Die Behörde hat auch einige Websites beanstandet, die ein Kontaktformular genutzt, aber keine verschlüsselte Übertragung der Kontaktdaten sichergestellt haben.

Datenschutzhinweise

Eine weitere Neuerung gibt es im Datenschutzrecht zu vermelden. Das Bundesministerium der Justiz und für Verbraucherschutz (BMJV) hat Mitte November 2015 ein spezielles Muster für Datenschutzhinweise vorgestellt. Dieser sogenannte One-Pager weist die Besonderheit auf, dass alle relevanten Daten auf einer einzigen DIN-A4-Seite dargestellt werden (Bild 1). Allerdings sei darauf hingewiesen, dass die Nutzung dieses Muster nicht verpflichtend ist. Es stellt lediglich eine Möglichkeit der Aufbereitung derjenigen Informationen dar, die jedes Unternehmen nach wie vor vorhalten und mitteilen muss. Insoweit hat sich an den datenschutzrechtli-



Der One-Pager des BMJV wurde im Rahmen des nationalen IT-Gipfels 2015 vorgestellt (Bild 1)

chen Vorgaben also nichts geändert. Je nach Internetpräsenz kann der Umfang der datenschutzrechtlichen Pflichtinformationen schlichtweg so groß sein, dass er sich nicht sinnvoll auf einer einzigen Seite darstellen lässt. Daher stellt dieses gut gemeinte, aber nicht gut gemachte Muster der Bundesregierung wohl nicht mehr als eine Orientierungshilfe dar.

Energiekennzeichnung

Bereits schon seit Anfang des vergangenen Jahres gilt eine geänderte Pflicht für Energiekennzeichnungen zum Beispiel in Online-Shops. Grund dafür ist die Delegierte EU-Verordnung Nr. 518/2014 vom 5. März 2014 im Hinblick auf die Kennzeichnung energieverbrauchsrelevanter Produkte im Internet und damit einhergehende Anpassung der deutschen Energieverbrauchskennzeichnungsverordnung (EnVKV).

Beim Verkauf via Internet ist vom Shop-Betreiber sicherzustellen, dass das Etikett und das Datenblatt eines Produkts ohne zusätzlichen Verwaltungsaufwand angezeigt werden. Dies kann aus Platzgründen auch mittels einer geschachtelten Anzeige realisiert werden.

Dabei wird eine Darstellungsform eingesetzt, bei welcher der Zugang zu Bildern oder Text per Mausklick oder Mausover auf ein anderes Bild beziehungsweise durch Berühren oder Aufziehen eines anderen Bildes auf einem Touchscreen-Gerät erfolgt.

Von den neuen Kennzeichnungspflichten betroffen sind die folgenden Produkte:

- Haushaltsgeschirrspüler,
- Haushaltskühlgeräte,
- Haushaltswaschmaschinen und -wäschetrockner,
- TV-Geräte,
- Staubsauger,
- Raumheizgeräte,
- Warmwasserbereiter,
- Lampen und Leuchten.

Bei all diesen Produkten muss jeweils ein elektronisches Etikett sowie ein elektronisches Produktdatenblatt angezeigt werden; bei Lampen und Leuchten genügt ein elektronisches Etikett. Bei Etikett und Produktblatt muss deren Größe so gewählt werden, dass sie gut sichtbar und leserlich sind.

Die neuen Informationspflichten treffen jedoch nur die Modelle, die ab dem 1. Januar 2015 neu in den Handel kommen – Altprodukte können freiwillig die neue Kennzeichnung erhalten.

Batterien und Akkus

Zum 27. November 2015 trat eine Änderung des Batteriegesetzes (BattG) in Kraft. Neben Herstellern und Importeuren sind auch Online-Händler als Vertrieber von Batterien und Akkus davon betroffen. Im Kern geht es um folgende Neuerungen:

- rückwirkendes Verbot für Knopfzellenbatterien mit mehr als 0,0005 Gewichtsprozent Quecksilber,
- erweitertes Verbot für Gerätebatterien mit mehr als 0,002 Gewichtsprozent Cadmium (gilt für alle ab dem 1. Januar 2017 erstmalig in den Verkehr gebrachte Gerätebatterien für schnurlose Elektrowerkzeuge),
- Neuregelung der Pfandpflicht beim Online-Vertrieb von Fahrzeugbatterien (zwecks Vermeidung der Rücksendungen von Fahrzeugbatterien per Post an den Onlinehändler aufgrund von potenziellen Umweltgefahren).

Die Pflicht zur Umsetzung einer Rücknahmepflicht durch alle (Online-)Händler, die Batterien beziehungsweise Akkus vertreiben, bleibt weiterhin bestehen. Dies gilt gleichsam für die Hinweispflicht auf die Einhaltung der Regelungen des

BattG. Alle betroffenen Online-Händler müssen also entsprechende Pflichtinformationen bereitstellen – was in etwa mit der Pflicht zur Bereitstellung des Impressums oder der Datenschutzerklärung zu vergleichen ist.

Elektronik

Auf das Elektro- und Elektronikgerätegesetz (ElektroG) hat eine Anpassung erfahren, die neuen Regelungen gelten seit dem 24. Oktober 2015. Ausgangspunkt war auch in diesem Fall europäisches Recht, nämlich die EU-Elektroaltgeräterichtlinie (Waste Electrical and Electronic Equipment oder kurz WEEE). Umfasst werden – untechnisch gesprochen – nahezu alle Geräte, die für ihren bestimmungsgemäßen Einsatz Strom benötigen. Jeder Händler, der auch nur ein einziges solches Produkt im Sortiment hat, muss die Regelungen des ElektroG beachten. Kernpunkte des ElektroG sind:

- Registrierungspflicht für Hersteller bei der Stiftung Elektro-Altgeräte (EAR),
- Hersteller von Photovoltaikmodulen müssen sich bis zum 1. Februar 2016 bei der EAR registrieren,
- erweiterte Informationspflichten,
- Pflicht zum Recycling von Altgeräten durch einen qualifizierten Entsorgungsbetrieb und entsprechendem Nachweis gegenüber der EAR.

Online-Händler haben die Pflicht, bis spätestens zum 24. Juli 2016 die Voraussetzung zur kostenlosen Rücknahme von Elektroaltgeräten in zumutbarer Entfernung zu ihren Endkunden im gesamten Bundesgebiet einzurichten. Dies gilt für alle Händler mit einer Versand- und/oder Lagerfläche ab 400 qm für den Versand von Elektrogeräten. Damit verbunden sind umfassende Registrierungs-, Melde- und Hinweispflichten. Dabei haben die einzelnen EU-Mitgliedsstaaten zum Teil unterschiedliche Regelungen hinsichtlich der genauen Abwicklung der Rückgabe, was für Online-Händler wichtig ist, die nicht ausschließlich innerhalb Deutschlands verkaufen.

Gemäß ElektroG müssen alle davon erfassten Produkte, die nach dem 13. August 2005 in der EU in Verkehr gebracht worden sind, dauerhaft so gekennzeichnet sein, dass die Identität des Herstellers, der Zeitpunkt des Inverkehrbringens sowie das Symbol der durchgestrichenen Mülltonne eindeutig erkennbar sind. Dauerhaft bedeutet hier: Die Kennzeichnung muss mit dem Produkt fest verbunden sein und darf nicht ohne Weiteres ablösbar sein. Zur eindeutigen Identifizierung des Herstellers sind folgende Daten anzugeben: Name, Handelsmarke, Warenzeichen und registrierte Firmennummer. ■

Links zum Thema

- Video-Trainings des Autors
www.video2brain.com/de/trainer/michael-rohrlich
- Blog des Autors zum Thema Online-Recht für Webmaster
<http://webmaster-onlinerecht.de>
- Blog des Autors zum Verbraucherrecht online
<http://verbraucherrechte-online.de>
- One-Pager des BMJV
www.bmjbv.de/SharedDocs/Downloads/DE/PDF/Verbraucherschutz/OnePager/11192915_OnePager-Datenschutzhinweise.pdf?__blob=publicationFile&v=2



Michael Rohrlisch

ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes. Online ist er über folgende Website erreichbar:
www.rechtssicher.info

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Trends am Arbeitsmarkt

Bis zu 4,4 Prozent mehr Gehalt

Für Software-Ingenieure, Software-Architekten und Entwickler erwartet der Personaldienstleister Robert Half für das Jahr 2016 Gehaltssteigerungen von bis zu 4,4 Prozent, beziehungsweise zwischen 1.000 und mehr als 3.000 Euro im Jahr. Laut Robert Half bleibt der Bedarf an IT-Profis in den Bereichen Big Data, Cloud und Mobility sehr hoch. Zusätzliche Nachfrage nach qualifizierten Fachkräften entsteht, da CIOs und CTOs einen besonders großen Einfluss der IT auf Effizienzsteigerungen im Unternehmen sehen. Diese Entwicklungen haben steigende Gehälter zur Folge. In der Gehaltsübersicht 2016 des Personaldienstleisters belegen die ersten fünf Plätze unter den IT-Profis die Berufe Netzwerk- und Security-Administratoren (plus 5 Prozent), Software-Entwickler Java (plus 4,4 Prozent), Software-

Architekten (plus 4,3 Prozent), Software-Ingenieure (plus 4,3 Prozent) und Webentwickler/TYPO3 (plus 4,1 Prozent). Nicht in die Spitzengruppe geschafft haben es die PHP-Entwickler, für die aber immerhin noch ein Gehaltsplus von 2,9 Prozent prognostiziert wird.

Besonders interessant: Robert Half nennt auch Jahressgehälter, abgestuft nach Berufserfahrung. So liegt das Einstiegsgehalt 2016 für PHP-Entwickler bei 37.000 und 44.250 Euro, Webentwickler, die TYPO3 beherrschen, bekommen zwischen 40.000 und 46.250 Euro. Java-kundige Berufseinsteiger sind mit 43.250 bis 50.500 Euro angegeben (Bild 1). Mit noch etwas mehr Gehalt steigen Software-Architekten und Software-Ingenieure in ihr Berufsleben ein: Robert Half nennt hier zwischen 45.250 und 52.250 Euro. Mit zunehmender Berufserfahrung steigen die Gehälter weiter. Mit neun Jahren Erfahrung nennt Robert Half für PHP- und Webentwickler Jah-

resgehälter von bis zu 62.000 Euro, bei den anderen Software-Berufsgruppen steigen die Werte jeweils auf mehr als 70.000 Euro.

Unter www.roberthalf.de/ steht eine Broschüre aktuellen Gehaltsangaben für IT-Positionen zum kostenlosen Download bereit.

Jobs für Web- und Mobile-Entwickler

Wie Bild 2 zeigt, suchen die meisten Stellenangebote für Mobile-Entwickler derzeit Mitarbeiter für eine Anstellung in den Bundesländern Bayern, NRW und Baden-Württemberg. Eine Auswertung nach Städten ergab, dass der Löwenanteil (41,9 Prozent) der freien Jobs in München, Berlin, Hamburg und Frankfurt zu finden ist. Diese Großstädte sind auch für Webentwickler die erste Wahl.

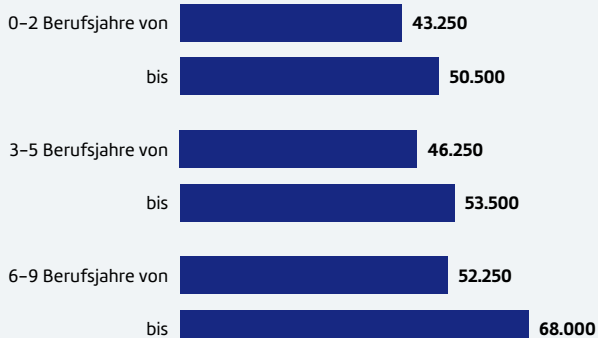
Tabelle 1 informiert darüber, welche Technologiekenntnisse derzeit in Stellenanzeigen besonders stark nachgefragt werden.

Tabelle 1

Rang	Technologie	Anteil *
1	Cloud	14,7 %
2	MySQL	11,0 %
3	SharePoint	9,8 %
4	HTML5	8,7 %
5	Responsive Web	8,0 %
6	Big Data	6,5 %
7	Android	6,3 %
8	Microsoft SQL Server	6,2 %
9	Windows 10	6,2 %
10	iOS	5,6 %
11	CSS3	4,2 %
12	Angular.js	3,5 %
13	ASP.NET	3,1 %
14	WPF	2,5 %
15	NoSQL	2,1 %
16	WCF	1,5 %

* Prozentualer Anteil der Treffer

Gehalt für Java-Entwickler



Das Einstiegsgehalt für Java-Entwickler gibt Robert Half in seinem Gehaltsreport 2016 mit 43.250 Euro an (Bild 1)

web & mobile developer 2/2016

Quelle: www.roberthalf.de/gehalt

Jobs für Mobile-Entwickler



Die meisten Jobs für Mobile-Entwickler gibt es derzeit in Bayern, NRW und Baden-Württemberg (Bild 2)

web & mobile developer 2/2016

Quelle: Jobkralle.de

Zahl des Monats

Immerhin **49,3 %** der Informatikstudenten kann sich eine selbstständige Tätigkeit nach dem Studium vorstellen.

22,2 % wollen diesen Schritt sogar direkt nach dem Studium tun.

Quelle: Etengo.de

Bitkom

Perspektiven für Bachelor und Master

Unternehmen wollen in den kommenden fünf Jahren verstärkt Hochschulabsolventen einstellen. Dabei haben auch Abgänger von Fachhochschulen steigende Chancen. Das ist das Ergebnis einer repräsentativen Befragung von Geschäftsführern und Personalverantwortlichen in IT- und TK-Unternehmen. 42 Prozent geben an, der Bedarf an Fachhochschulabsolventen werde bis 2020 steigen. Auch Abschlüsse von Dualen Hochschulen wie Berufsakademien sind wertvoll: 39 Prozent sehen hier einen wachsenden Bedarf. Absolventen von Universitäten wollen 35 Prozent verstärkt einstellen (Bild 3). »Der Trend in der digitalen Wirtschaft ist eindeutig: Wichtig für den Berufseinstieg ist eine fundierte akademische Ausbildung, bei der ein möglichst hoher Praxisbezug erwartet wird«, sagt Bitkom-Chef Rohleder.

Bachelor- und Master-Abschlüsse sind bei den Unternehmen übrigens gleichermaßen gefragt. An Bedeutung verliert dagegen die Promotion.

Glassdoor

Jobinterviews

Die Karriereplattform Glassdoor hat die Daten von 154.000 Arbeitnehmern aus sechs Ländern – darunter auch Deutschland – verglichen und herausgefunden, dass neue Mitarbeiter zufriedener im Job sind, wenn sie ein schwieriges Bewerbungsgespräch meistern mussten. Laut Glassdoor-Studie liegt die optimale Schwierigkeitsstufe bei 4 auf einer Skala von 1 (sehr leicht) bis 5. War das Bewerbungsgespräch allzu schwierig, sank die Zufriedenheit der Mitarbeiter wieder.

Bei der Gretchenfrage, wie das Jobinterview aussehen muss, um den richtigen Bewerber auszuwählen, geben die Macher der Studie einige Tipps. So solle man sich einen genau-

en Plan für das Interview zu-rechtlegen, nicht vor schwierigen Fragen zurückschrecken und prüfen ob der Bewerber zur Unternehmenskultur passt.

Bitkom Research

Weiterbildungsbedarf

Die deutsche Wirtschaft sieht großen Bedarf bei der Weiterbildung ihrer Fachkräfte zu Digitalthemen. Mehr als vier von zehn Unternehmen halten es für notwendig, Mitarbeiter in der Datenanalyse zu schulen. Jedes dritte Unternehmen hält zusätzliche Kenntnisse rund um Social Media und Programmieren (je 35 Prozent) für notwendig. Das sind Ergebnisse einer repräsentativen Bitkom-Umfrage. Jedes vierte Unternehmen sieht großen Bedarf bei der Fortbildung zu Datenschutz und Datensicherheit, 16 Prozent bei allgemeinen PC-Anwenderkenntnissen. Nur drei Prozent sehen im eigenen Unternehmen keinen Weiterbildungsbedarf zu diesen Themen.

Solcom

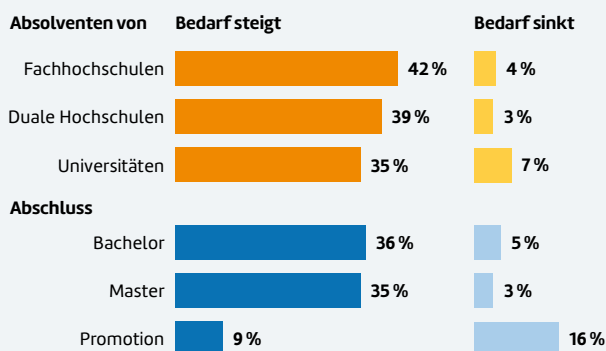
Freiberuflersorgen

Nur sieben von zehn Freiberuflern empfinden sich laut einer Solcom-Umfrage als gut abgesichert. Trotz der allgemein guten Absicherung in der Breite kann auch heute nicht jeder der Befragten eine Krankenversicherung aufweisen. Zudem haben auch nur drei Viertel eine Altersvorsorge. Eine Haftpflicht-Versicherung gehört jedoch für fast neun von zehn Umfrageteilnehmern zum eigenen Schutz. Über 95 Prozent der Freiberufler geben weniger als 30 Prozent ihres Einkommens für ihre Absicherung aus.

Die höchsten beruflichen Risiken sehen die Umfrageteilnehmer in erster Linie bei der Kündigung beziehungsweise beim Abbruch des Projekts.

Beim Thema Selbstmarketing befürchten fast zwei Drittel der befragten Freiberufler den Verstoß gegen eine Geheimhaltungsvereinbarung als größtes Risiko für ihre Tätigkeit (Bild 4).

Bedarf an Absolventen

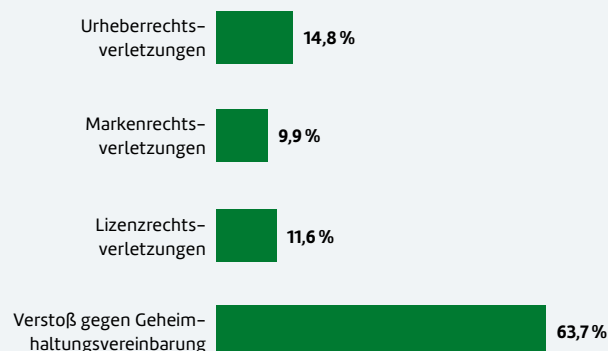


Absolventen von Fachhochschulen sind 2016 stärker gefragt. Ein Dokortitel ist nicht mehr so wichtig (Bild 3)

web & mobile developer 2/2016

Quelle: Bitkom

Probleme beim Selbstmarketing

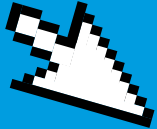


Einen Verstoß gegen Geheimhaltungsvereinbarungen sehen Freelancer als größtes Risiko (Bild 4)

web & mobile developer 2/2016

Quelle: www.solcom.de

dotnetpro Newsletter



Top-Informationen für den .NET-Entwickler.
Klicken. Lesen. Mitreden.



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 Internetdienstleistungs 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



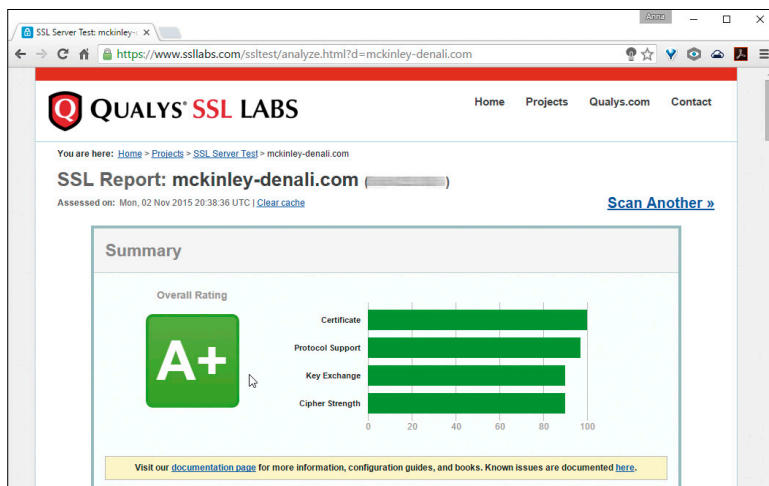
digitalmobil GmbH & Co. KG

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 3/2016 erscheint am 11. Februar 2016

Sicherheit von Webanwendungen und mobilen Apps



Die Sicherheit von Webanwendungen und mobilen Apps treibt vielen Entwicklern immer wieder Sorgenfalten auf die Stirn. Das Bekanntwerden unerwarteter Verwundbarkeiten aufgrund uralter Bugs in OpenSSL (Heartbleed), in der Bash-Shell (Shellshock), durch verwundbare Hypervisors (Venom) und das mittlerweile leicht hackbare Duo aus HTTP/1.1 und SSL/TLS hat die IT-Gemeinde erschüttert. Neue Angriffsvektoren bedrohen die Sicherheit von Webanwendungen, Backend-Systemen und mobilen Apps auf vielfältige Art und Weise. Während Großunternehmen mit Milliarden Dollar starken IT-Budgets sich auch von einer gravierenden IT-Sicherheitslücke leicht erholen können, kann ein vergleichbarer Vorfall einem kleinen Entwicklerteam einen irreparablen Schaden zufügen.

JavaScript-Framework Ext JS 6

Bis zu den Vorgängerversionen existierten bei Sencha noch zwei getrennte Welten: das Framework Ext JS ausschließlich für Desktop und Tablets, und das Framework Sencha Touch speziell auf mobile Endgeräte ausgerichtet. Für die neue Version 6 von Ext JS entschloss sich der Hersteller, seine beiden getrennten Frameworks zu einem gemeinsamen Framework für die Entwicklung verschiedener Anwendungstypen zu vereinen.

Search Engine Blocking

Was beim Großteil von Websites völlig normal ist, kann allerdings in einigen Fällen nachteilig sein: Unter gewissen Umständen sollte eine Internetseite gar nicht (oder noch nicht) in den Ergebnislisten von Suchmaschinen auftauchen. Typischerweise zählen hierzu Websites, die sich in der Entwicklung befinden oder lediglich als Platzhalter für eine Internetpräsenz dienen, die in Kürze veröffentlicht werden soll. Hier hilft Search Engine Blocking weiter.

Accelerated Mobile Pages

Der nächste große Trend im Medienbereich wird sicher die plattformunabhängige Veröffentlichung von Inhalten werden. Während Facebook seit Mai 2015 seine Instant Articles gestartet hat, arbeitet Google ebenfalls an einem ähnlichen Format mit dem Namen Accelerated Mobile Pages oder kurz AMP. Dieses Format wird zusammen mit Twitter als Open-Source-Lösung entwickelt und soll Anfang 2016 reif für den produktiven Einsatz sein.

dotnetpro



Ausgabe 2/2016 ab 21. Januar 2016 am Kiosk

Im Schwerpunkt der dotnetpro 2/2016 geht es um die Analyse von Daten großer Menge oder großer Komplexität. Beispielhaft erläutert werden in dem Schwerpunkt das Machine Learning unter Microsoft Azure sowie die Textanalyse mit NER.

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](https://plus.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 Schmidtman, Klaus Ahlering • Tel. 089/74117-125 • sales@nmg.de

Wir machen Sie **digitalmobil.**

Wir liefern passgenaue Strategien und
Lösungen für Ihre Inhalte auf

- iPhone/iPad
- Android
- BlackBerry
- Windows Phone 7
- dem mobilen
Browser



Besuchen Sie uns unter www.digitalmobil.com