

Graphen statt Relationen

So bauen Sie mit Neo4j
leistungsfähige Graph-
Datenbanken auf S. 20



INFO-
Programm
gemäß
§ 14
JuSchG



Auf der Heft-CD

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

Ausgabe 5/16

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>



Graph-Datenbanken

Alle Strukturen der realen Welt lassen sich als Graphen interpretieren.

Mit relationalen Datenbanken stößt man bei der Verarbeitung von Graphen mit vielen Beziehungen schnell an Grenzen, da diese in ihrem Datenmodell den Fokus vollkommen auf Attributwerte, also Eigenschaften, und nicht auf die modellierte Struktur (den Graphen) legen. Für die Speicherung von Graphen benötigt man ein Datenbanksystem, das besonderen Wert auf die Beziehungen zwischen den beteiligten Entitäten legt – wie es bei einer Graph-Datenbank der Fall ist. Das Schwerpunktthema dieser Ausgabe befasst sich ab Seite 20 mit dem Aufbau einer Graph-Datenbank mit Neo4j.

Bei Tonic handelt sich um eine Sandbox-Umgebung für JavaScript/Node.js im Webbrowser. Visuell ist Tonic sehr ähnlich wie Mathematica-Arbeitsblätter und sehr interaktiv gestaltet. Der Dienst ist kostenlos und bietet eine ganze Reihe höchst interessanter Features. Jochen Schmidt stellt den Dienst in einem Artikel ab Seite 38 im Detail vor.

»Die Validierung von Formularen ohne JavaScript ist nicht sonderlich flexibel.«

Noch vor einigen Jahren war die Validierung von Formularfeldern nur mit JavaScript möglich. Mittlerweile bietet HTML schon von Haus aus gewisse Möglichkeiten zur Überprüfung von Formularfeldern an. Allerdings ist die Validierung ohne JavaScript nicht sonderlich flexibel. Doch es gibt Abhilfe: Dank des sogenannten Constraint Validation API lässt sich die native Validierung von HTML5 nämlich auch per JavaScript steuern. Wie das funktioniert, schildert ein Artikel von Philip Ackermann ab Seite 52.

Zeitgemäße Betriebssysteme und Applikationen im Desktopbereich haben eine grafische Bedienoberfläche. Im Serverbereich und auch bei Entwicklungswerkzeugen verschiebt sich das Bild. Das primäre Interesse von Admins, Entwickler oder Softwaretestern liegt in einer möglichst effizienten Erledigung der jeweiligen Arbeitsaufgabe. Mit Scripting auf Anwendungsebene lassen sich die Vorteile grafischer Oberflächen mit der Flexibilität, Interaktivität und Ausdruckstärke einer externen Skriptsprache verbinden. Wie das mit ClearScript und C# geht, erläutert ein Artikel von Jens Geyer ab Seite 116.

Ihr Max Bold
chefredakteur@maxbold.de



Frank Simon

zeigt den Aufbau einer Graph-Datenbank (S. 20)



Jochen Schmidt

präsentiert die Sandbox-Umgebung Tonic (S. 38)



Philip Ackermann

stellt das Constraint Validation API vor (S. 52)

INHALT

Aktuell

Apps aus der Cloud

1&1 kooperiert mit Bitnami

Feature

Graph-Datenbanken mit Neo4j

Alle Strukturen der realen Welt lassen sich als Graphen interpretieren

HTML, CSS & JavaScript

Unbekanntere CSS3-Features

CSS3 bietet auch weniger bekannte, aber durchaus nützliche Features

Tonic: Interaktive Node.js-Umgebung im Browser

Die Idee von webbasierten Code-Sandkästen ist nicht neu

Modularisierung in JavaScript

Wie man in JavaScript Quelltext in Module zusammenfassen kann

Constraint Validation API

Das Constraint Validation API vereinfacht die Validierung von Formulareingaben

Diagramme mit Google Chart Tools

Mit den Google Chart Tools lassen sich vielfältige Diagramme in HTML-Seiten einbauen

Mobile Development

MVC in der iOS-Entwicklung

Das MVC-Pattern gehört zu den wichtigsten Patterns in der iOS-Entwicklung

Schnelle App-Entwicklung

Entwickler haben es schwer, den oft dringenden Wünschen von Geschäftskunden gerecht zu werden

Enterprise Mobility Management (EMM)

EMM-Lösungen sind beim Management von Smartphones und Tablets unverzichtbar



6

20

32

38

44

52

69

72

84

88

Foto: Mathias Vietmeier



Alle Strukturen der realen Welt lassen sich als Graphen interpretieren und in Graph-Datenbanken ablegen

S. 20



Mit dem CSS-Befehl **object-fit** lassen sich unterschiedliche Bildgrößen in HTML-Dokumenten in den Griff bekommen

S. 32

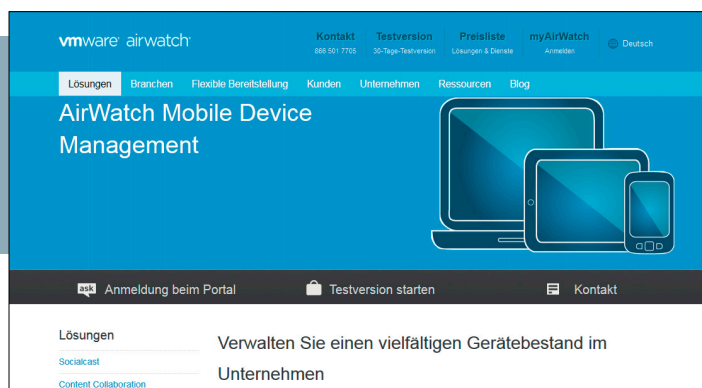
Experten in dieser Ausgabe



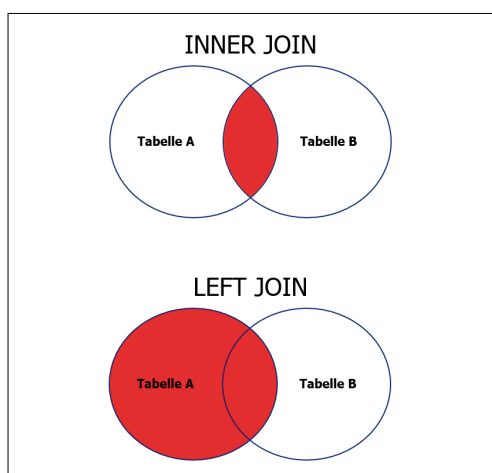
Dr. Florence Maurice präsentiert einige weniger bekannte, aber dennoch nützliche CSS3-Features. 32



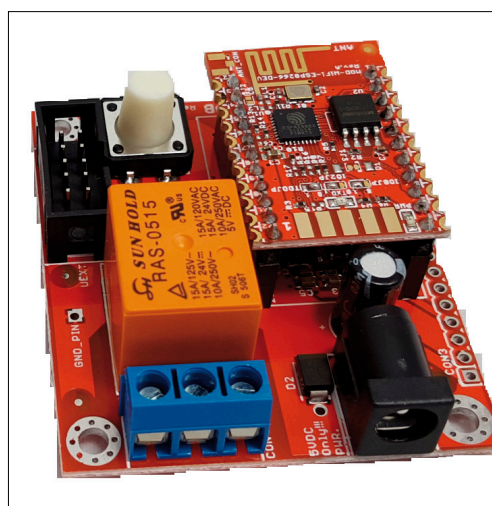
Markus Schraudolph zeigt, wie man den SQL-Befehl JOIN bei der Abfrage von Datenbanken richtig einsetzt. 108



Die Einführung eines EMM-Systems zusammen mit Business-Apps kann Mitarbeitern helfen, produktiver zu arbeiten **S. 88**



Eine der Schlüsselideen von relationalen Datenbanken ist die logische Trennung von Daten in einzelnen Tabellen. Mit dem JOIN-Befehl kann man die Daten wieder zusammenführen **S. 108**



Die ESP8266-Platine von Olimex kommt direkt mit Relais und eignet sich somit hervorragend für Schaltaufgaben **S. 130**

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>

iOS-Apps mit Handoff-Funktion

Mit dem Handoff API ist es möglich, eine geräteübergreifende Bearbeitungsfunktion für Apps zu erstellen **94**

Spieleprogrammierung unter OpenGL

Spiele für Android statt mit Unity unter OpenGL zu programmieren kann lehrreich sein **98**

Backend

JOINS in SQL-Datenbanken meistern

JOINS sind ein mächtiges Werkzeug der Datenbankabfrage **108**

Server virtualisieren

Im Server-Umfeld ergeben sich durch den Einsatz von Virtualisierungssystemen vielfältige Möglichkeiten **114**

ClearScript und C#

Auch Entwickler und Admins wissen die Vorteile grafischer Oberflächen zu schätzen **116**

Beyond Dev

Grafik für Entwickler: Adobes Videolösungen

Diverse Adobe-Programme helfen bei der Erstellung von Videoclips und Animationen **126**

Den Mikrocontroller ESP8266 programmieren

Der Baustein ESP8266 befördert Ihr Projekt kostengünstig auf die Datenautobahn **130**

Standards

Editorial **3**

Heft-CD **50**

Online-Recht **138**

Impressum **141**

Arbeitsmarkt **142**

Dienstleisterverzeichnis **145**

Vorschau **146**

NEWS & TRENDS

AKTUELLE NEWS AUS DER ENTWICKLERSZENE

Adobe Digital Index

Deutschland nur im Mittelfeld

Deutschlandweit stammen rund ein Drittel aller Webseiten-Aufrufe von Mobilgeräten. Im globalen Vergleich der mobilsten Länder 2015 liegt Deutschland damit auf dem 18. Rang. Dies ist das Ergebnis des aktuellen Adobe Digital Index (ADI), für den die Daten von über 4500 Unternehmen ausgewertet wurden. Mit einem Anteil an mobilen Zugriffen von 49 Prozent konnte sich Irland den Titel als Mobile-Europameister sichern. Weltweit hat Saudi-Arabien mit 62 Prozent klar die Nase vorn. Insgesamt kamen 2015 rund 37 Prozent aller Website-Aufrufe von Mobilgeräten – dies bedeutet eine Steigerung von 18 Prozent gegenüber dem Vorjahr.

www.cmo.com/adobe-digital-index.html

Globaler Mobile-Zugriff

Saudi-Arabien	61,96 %
Irland	49,35 %
Philippinen	47,11 %
Japan	47,04 %
Großbritannien	43,92 %

Die Top 5 der mobilsten Länder 2015 im Überblick.

web & mobile developer 5/2016
Quelle: Adobe Digital Index

Apps aus der Cloud

1&1 kooperiert mit Bitnami

Bitnami stellt über 100 Open-Source-Apps auf Server-Infrastrukturen ausgewählter Partner als One-Click-Installationen bereit.

Dank der neuen Partnerschaft können 1&1-Kunden jetzt mit nur einem einzigen Klick interessante Webapplikationen und Development Stacks in der Cloud ausrollen und betreiben. Die Apps werden zentral ausgeliefert und lassen sich mit ein paar Mausklicks in wenigen Minuten installieren.

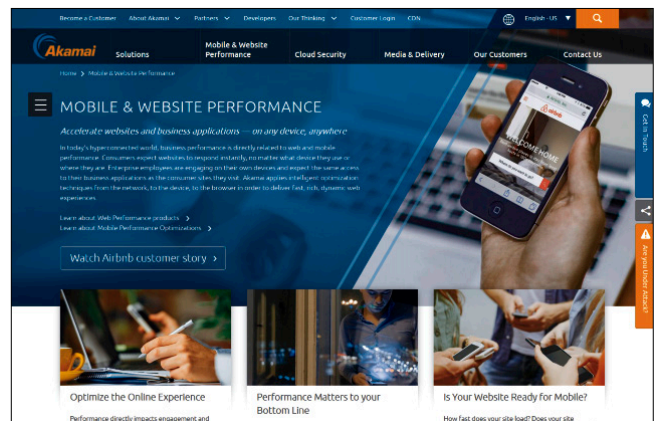
Robert Hoffman, CEO 1&1 Internet SE, verdeutlicht: »Unser Ziel ist es, den bestmöglichen Service beim Aufbau von Cloud-Infrastrukturen und cloudbasierter Entwicklung zu



Mit einem Klick Webapplikationen und Development Stacks in der Cloud ausrollen

bieten. Bitnami war deshalb die ideale Wahl bei der Suche nach einem Partner. Unsere Kunden haben dank des 1&1 Cloud App Centers nun die Möglichkeit, Apps in kürzester Zeit auszurollen, zu testen und kontinuierlich zu nutzen. Eine einfache Verwaltung ermöglicht zudem, Innovationen schneller als bisher voranzutreiben.«

<http://hosting.1und1.de/cloud-app-center/cloud-apps>



Predictive Content Delivery: Sichere Bereitstellung qualitativ hochwertiger Videos

Akamai

Optimierte Auslieferung von Videos auf mobilen Endgeräten

Akamai Technologies hat eine neue Lösungen für Predictive Content Delivery präsentiert. Damit lässt sich die sichere Bereitstellung qualitativ hochwertiger Videos in HD-Qualität auf mobilen Endgeräten in allen Netzen optimieren.

Da häufig angeschaut Videos auch auf den Devices zwischengespeichert werden können, lässt sich eine bessere Bildqualität erzielen. Benutzer können die Filme somit auch offline auf jedem Device und über jedes Netz anschauen.

»Mobile Endgeräte spielen eine immer wichtigere Rolle für den Aufbau von Kundenbindung und -zufriedenheit bei Online-Videos. Mit den Lösungen von Akamai für Predictive Content Delivery erhalten unsere Kunden leistungsstarke Werkzeuge, um diese anspruchsvollen Herausforderungen meistern zu können«, sagt Lior Netzer, Vice President and

General Manager, Emerging Mobile Business Unit bei Akamai.

Die Predictive-Content-Delivery-Lösungen (PCD) von Akamai adressieren die zentralen Anforderungen von Content-Providern, Online-Videoplattformen und Betreibern von Mobilfunknetzen. Sie sind in zwei Konfigurationen erhältlich: erstens einem Software Development Kit (SDK), das sich einfach in vorhandene oder neue Media-Apps integrieren lässt, und zweitens der sofort einsatzfähigen White-Label-App Akamai WatchNow.

<https://www.akamai.com/de/de>

TYPO3

Die Version 8 arbeitet mit PHP 7

Das Kernteam von TYPO3 hat mit der Entwicklung der nächsten Plattform-Edition TYPO3 CMS 8 LTS begonnen und stellt die entsprechende Roadmap dazu vor. Im Rahmen des neuen Entwicklungszyklus ist geplant,

Zahl des Monats

29 volle Arbeitstage pro Jahr sind Manager mit dem Checken und Bearbeiten von E-Mails beschäftigt. Das entspricht in etwa dem Jahresurlaub eines Angestellten, die ein Manager allein mit dem Abarbeiten von E-Mails verbringt.

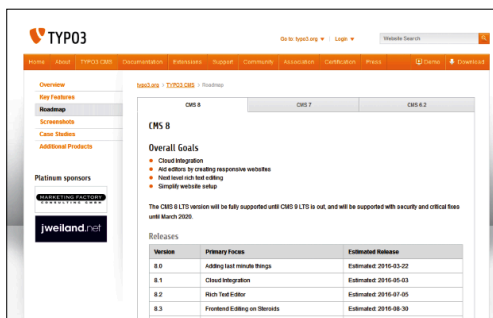
Quelle: Chartered Management Institut

bis April 2017 alle acht bis zehn Wochen Zwischenversionen zu veröffentlichen. Dies gewährleistet, dass Entwickler neue Projekte direkt auf Version 8 aufsetzen können.

Das Hauptziel der kommenden Version 8 ist die Vereinfachung der täglichen Arbeitsprozesse für die Nutzer. Dazu zählen neue Funktionen zur Bear-

gen Minuten upgraden. Dies ist besonders für Entwickler interessant, die für ihre TYPO3-CMS-Instanzen früh auf die 8.x-Linie wechseln möchten.

Es ist zu erwarten, dass jede größere Distribution mit der Veröffentlichung von TYPO3 CMS 8 LTS im April 2017 stabile Builds als Standard bietet. Das Team garantiert eine Unterstüt-



Die Version 8
des CMS
TYPO3
arbeitet mit
PHP 7

beitung von Inhalten im Frontend und dem TYPO3-Backend. Zudem soll die Plattform durch die weitere Entschlackung der Code-Basis noch robuster und flexibler werden. Auch die grafische Bedienoberfläche wird verbessert. Ein weiterer Fokus beim TYPO3 CMS 8 Entwicklungszyklus liegt in der Vereinfachung des Setups für Cloud-Infrastrukturen.

Unternehmen benötigen mindestens PHP Version 7, um TYPO3 CMS 8 zu nutzen. Das Entwicklerteam ist sich bewusst, dass diese Entscheidung durchaus ehrgeizig ist. Allerdings gibt es dafür gute Gründe: Die wichtigsten Betriebssysteme und Distributionen bieten in ihren stabilen Versionen bereits Repositories, um PHP 7 zu nutzen. Debian GNU/Linux und Ubuntu-Derivate lassen sich in der Regel innerhalb von weni-

gen Minuten upgraden. Dies ist besonders für Entwickler interessant, die für ihre TYPO3-CMS-Instanzen früh auf die 8.x-Linie wechseln möchten.

<https://typo3.org/typo3-cms/roadmap>

Ambiverse

Neue Technologie zur Textanalyse

Ambiverse hat eine Technologie für die Analyse von großen Textbeständen entwickelt, bei der künstliche Intelligenz im Hintergrund buchstäblich mitdenkt. Wer den Aktienkurs eines Unternehmens vorhersagen will, wertet dafür Nachrichtenportale und soziale Medien aus. ►

IDC und Veracode

Sichere Connected-Car-Systeme

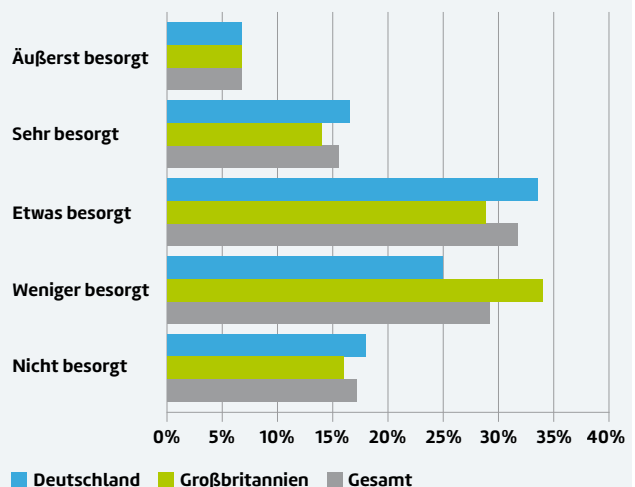
Mehr als die Hälfte der deutschen Autofahrer (57 Prozent) machen sich Sorgen um die Sicherheit von Assistenzsystemen im Auto wie die adaptive Geschwindigkeitsregelung oder das automatische Einparken.

Das ist das Ergebnis einer Studie zur Sicherheit von Anwendungen in vernetzten Automobilen, sogenannten Connected Cars, von Veracode, Sicherheits-Spezialist für mobile und Web-Anwendungen, und dem Beratungshaus IDC. Für die Studie wurden zum einen mehr als 1000 Autofahrer in Deutschland und dem Vereinigten Königreich befragt. Zum anderen hat IDC auch KFZ-Hersteller und Vertreter der Automobilindustrie um ihre Sicht der Dinge gebeten. Das Ergebnis: Laut IDC wird es noch bis zu drei Jahre dauern, bis Anwendungen und Systeme im Auto ausreichend gegen Cyberbedrohungen geschützt sind.

Anwendungen aus dem Internet stellen eine Herausforderung für die Sicherheit dar. Alle interviewten Hersteller haben Sicherheitsbedenken gegenüber Anwendungen, die sie nicht selbst entwickelt haben. Diese könnten dazu führen, dass der Hersteller nicht mehr die volle Kontrolle über die Sicherheit des Fahrzeugs hätte. Hersteller sollten für die Sicherheit eines Connected Cars haftbar sein. 87 Prozent der befragten Autofahrer sind der Ansicht, dass alle Sicherheitsaspekte eines Autos beim Hersteller liegen. Dazu zählt auch die Fähigkeit von Apps, Cyberangriffen zu widerstehen.

www.veracode.com

Die Sicherheit von Assistenzsystemen im Auto



Die Mehrheit der deutschen Autofahrer macht sich Sorgen um die Sicherheit von Assistenzsystemen im Auto.

web & mobile developer 5/2016

Quelle: IDC

Statista Digital Market Outlook

40 Prozent Wachstum im Connected-Car-Markt

Google und Apple haben bereits die ersten Schritte im Connected-Car-Markt gewagt, und auch Microsoft startet mit einer umfangreichen Offensive, um sich in der Geburtsstunde der vernetzten Automobile fest im Markt zu etablieren.

Im Lauf der vergangenen Jahre ist es alltäglich geworden, über das Smartphone im Auto die gewünschte Playlist abzuspielen oder über die Freisprechanlage Telefonate zu führen. Doch die intelligente Verbindung von alltäglichen Prozessen mit dem eigenen PKW kann noch einiges mehr, wie die Prognosen des Digital Market Outlooks (DMO) von Statista belegen.

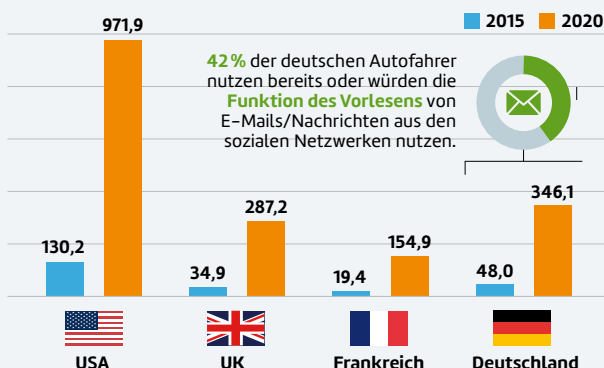
Bis dato steckt die Branche der Connected Cars noch in den Kinderschuhen: In diesem Jahr werden knapp über 3 Millionen Connected Cars in Deutschland zu zählen sein. Im Jahr 2020 soll sich laut DMO diese Zahl bereits vervierfachen haben, bis dahin werden 12,4 Millionen intelligente Autos auf Deutschlands Straßen unterwegs sein und weltweit sogar über 160 Millionen.

Den Großteil des Wachstums dieses Marktes trägt auch die Entwicklung der neuen Features und Möglichkeiten bei. Das größte Segment ist jedoch laut des DMO nicht das Infotainment, sondern die Sicherheit und Fahrassistenten. Dieser Markt wird klar aus China dominiert. Der Umsatz für die sichere und unterstützende Aufrüstung vernetzter Autos wird in China diesjährig über 3,5 Milliarden Euro betragen. Google und Apple haben in Kooperation mit VW bereits vorgelegt – ab Anfang 2016 ist nun nahezu jedes Volkswagen-Fahrzeug kompatibel mit dem Smartphone. Somit kann der Fahrer mobil seine Tankfüllung überprüfen oder das Auto ver- oder entriegeln. Das am stärksten wachsende Segment der vernetzten Automobile ist mit über 200 Prozent Umsatzwachstum in 2015 jedoch die intelligente Wartung und Diagnose des Autos – und sogar des Fahrers. Pay-how-you-drive ist ein Versicherungsformat, in dem Beiträge nach Fahrkunst des Fahrzeugbesitzers festgelegt werden.

<https://de.statista.com/outlook>

Nimmt Cortana bald auf dem Beifahrersitz Platz?

Umsatzentwicklung im Segment In-Car Content & Services (in Mio. Euro)



Prognosen des Digital Market Outlooks (DMO) von Statista belegen eine wachsende Bedeutung von Connected-Car-Services

Befragt wurden 3303 PKW-Besitzer (priv. Fahrzeug und Dienstwagen), die einen Neuwagen fahren oder den Kauf eines Neuwagens demnächst planen.

web & mobile developer 5/2016

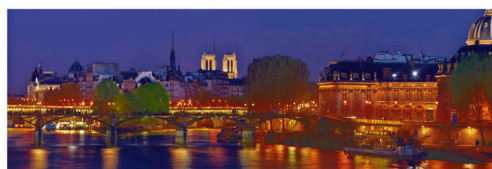
Statistica Digital Market Outlook, LSP Digital

Für diese Recherche gibt es mittlerweile viele Dienstleister, die ihre Suchmaschinen mit passenden Schlagwörtern füttern. Sie stoßen jedoch schnell an Grenzen, wenn mehrdeutige Begriffe nur in schwammig formulierten Zusammenhängen erscheinen. Saarbrücker Informatiker haben daher eine Technologie entwickelt, die mit Hilfe von künstlicher Intelligenz die

lich wachsen und es immer aufwendiger wird, diese sinnvoll zu strukturieren und mit Erfolg zu durchsuchen«, sagt Johannes Hoffart, Forscher am Max-Planck-Institut für Informatik und Gründer der Firma Ambiverse. Sein Team hat daher eine Technologie für die Analyse von großen Textbeständen entwickelt, bei der viel Rechenpower und künstliche Intelligenz im

„Um 5 Uhr hat Paris immer eine erotische Stimmung.“

Andis Nin



Paris Hilton oder Paris, France?

Images from Wikipedia – CC-BY-SA 3.0

Künstliche Intelligenz: Ambiverse analysiert Textbestände

Suche in großen Textbeständen wesentlich verbessert. Das neue Verfahren unterstützt auch Autoren bei der Recherche und beim Schreiben von Texten, indem es automatisch Hintergrundinfos anzeigt.

Die Firma Ambiverse, eine Ausgründung des Max-Planck-Instituts für Informatik in Saarbrücken, hat die neue Technologie auf der Cebit vorgestellt.

Auch im Zeitalter von Diensthandys und firmeninternen Chatrooms werden die meisten Informationen in Unternehmen nicht über das gesprochene Wort, sondern über Mails, Datenbanken und interne Newsportale ausgetauscht. »Laut einer Marktanalyse des Medienunternehmens Gartner nutzen derzeit jedoch nur ein Viertel der Unternehmen weltweit automatisierte Methoden, um ihre Textbestände zu analysieren und besser zu verstehen. In fünf Jahren werden es nach Gartners Schätzungen mehr als die Hälfte sein, da die Datenmengen in Firmen kontinuier-

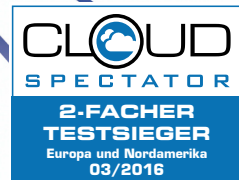
Hintergrund buchstäblich mitdenkt.

»Wir greifen bei unseren Textanalysen auf frei verfügbares Wissen zurück, zum Beispiel auf Wikipedia oder große Nachrichtenportale im Internet. Dieses können wir mit unternehmensspezifischem Wissen anreichern«, erläutert Hoffart. Über komplexe Rechenverfahren werden die Texte dann genauer durchleuchtet und mit Hilfe von linguistischen Werkzeugen analysiert. »Wir können dann Firmen und Branchen einzelnen Kategorien zuordnen und zum Beispiel untersuchen, wie sich die eigene Produkte im Vergleich mit den Angeboten der Konkurrenz im Markt positionieren konnten«, sagt Hoffart. Die Herausforderung sei dabei, dass die Namen von Unternehmen oder Produkten meist nicht nur einmal auf der Welt vergeben werden, sondern in verschiedenen Bedeutungen und in vielfältigen Zusammenhängen erscheinen.

www.ambiverse.com



APPS CLOUD-READY!



Powered by
intel Cloud
Technology

1&1 CLOUD SERVER

0,- €/Monat
danach ab
4,99 €/Monat*

Das **1&1 Cloud App Center** ist der schnellste Weg, Ihre Applikation zum Laufen zu bringen! Wählen Sie aus über 100 topaktuellen kostenlosen **Apps Ihre Anwendung** aus und kombinieren Sie diese mit der Geschwindigkeit und Performance von **Deutschlands bestem Cloud Server**.

- ✓ **Sichere und leistungsstarke Plattform**
- ✓ **Kein Server-Know-how notwendig**
- ✓ **Minutengenaue Abrechnung**



DE:02602/96 91
AT: 0800/100 668



1und1.info

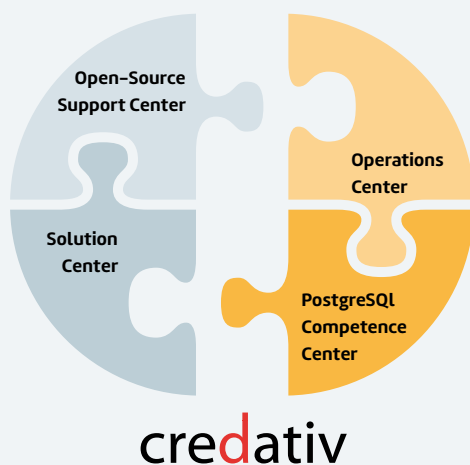
*1&1 Cloud Server 1 Monat 0,- €/Monat, danach ab 4,99 €/Monat. Einmalige Einrichtungsgebühr 9,99 €. Keine Mindestvertragslaufzeit. Preise inkl. MwSt. Intel® und das Intel® Logo sind eingetragene Marken der Intel Corporation in den USA und anderen Ländern. 1&1 Internet SE, Elgendorfer Str. 57, 56410 Montabaur.

EnterpriseDB und creativ

Mehr Support für Postgres

creativ ist ein auf Open-Source-Lösungen im professionellen Unternehmenseinsatz spezialisiertes Beratungsunternehmen, das jetzt Beratungsleistungen für Kunden von EDB anbieten will, vor allem in Deutschland, wo das Unternehmen umfassende Leistungen für deutschsprachige Kunden bereitstellen kann, aber auch an anderen Standorten. Kunden sollen künftig von der Performance und Zuverlässigkeit der professionellen EDB-Lösungen in Kombination mit dem lokalen Zugang zu Know-how und Supportleistungen profitieren können.

Das Unternehmen wurde 1999 gegründet und ist außer in Deutschland auch in den USA, Großbritannien, Kanada, den Niederlanden sowie Indien tätig. Im Rahmen einer Verein-



EnterpriseDB (EDB) hat die Zusammenarbeit mit creativ bekannt gegeben.

web & mobile developer 5/2016

Quelle: EDB

barung mit EnterpriseDB wird creativ seine Beratungsleistungen sowie einige der EDB-Leistungspakete wie Architectural Health Check und Database Migration Assessment zunächst in Deutschland anbieten. Im weiteren Verlauf wird EDB die Leistungen aber auch zur Unterstützung seiner Kunden in anderen Märkten, in denen creativ präsent ist, nutzen können.

An jedem seiner Standorte betreibt creativ Open-Source-Support-Center, die sowohl einen professionellen 24x7-Support als auch operatives Know-how anbieten. Darüber hinaus ist creativ ausgewiesener Spezialist für das Datenbanksystem PostgreSQL – die Berater des Unternehmens unterstützen die Open-Source-PostgreSQL-Community bereits seit der Unternehmensgründung.

EDB spielt eine führende Rolle bei der Transformation hin zu Open-Source-Datenbank-Managementsystemen und hilft Unternehmen dabei, die an ihre Rechenzentren zur Unterstützung strategischer Geschäftsziele gestellten Anforderungen zu erfüllen. Heute setzen bereits 20 Prozent der Fortune-500-Unternehmen das EDB-Postgres-Datenbank-Managementsystem ein, um die hierdurch eingesparten Mittel sinnvoller zur Unterstützung ihrer digitalen Geschäftsinitiativen einsetzen zu können.

www.enterprisedb.com



Prominenz auf der Industrie-4.0-Tagung des Hasso-Plattner-Instituts: die Vorstände Bernd Leukert (SAP) und Reinhard Clemens (Deutsche Telekom), Brandenburgs Ministerpräsident Dietmar Woidke, HPI-Direktor Prof. Christoph Meinel und Deloitte Consulting Deutschland-Partner Olaf Peter Schleichert (v. r. n. l.)

Industrie 4.0**100 Milliarden Euro für schnelles Internet**

Um die vernetzte Produktion voranbringen zu können, muss Deutschland in den nächsten zehn Jahren allein 100 Milliarden Euro in den Ausbau des schnellen Internets investieren.

Diese Zahl nannte Staatssekretär Matthias Machnig vom Bundeswirtschaftsministerium auf einer Konferenz des Hasso-Plattner-Instituts (HPI) zur Industrie 4.0.

Ein solcher Schub privater Investitionen in den Breitband-Netzausbau sei für die erfolgreiche Digitalisierung der Wirtschaft eine entscheidende Voraussetzung, sagte Machnig. Nach seinen Angaben verfügen bisher nur sieben Prozent der deutschen Nutzer über einen Anschluss ans besonders schnelle Glasfasernetz.

Laut Machnig ist die deutsche Industrie mit ihrem Digitalisierungsgrad zwar weltweit führend, es handle sich aber oft lediglich um Insellösungen. Gerade kleine und mittlere Unternehmen müssten auf dem Weg in das Internet der Dinge und Dienste mitgenommen werden, forderten Machnig und der brandenburgische Ministerpräsident Dietmar Woidke auf der Potsdamer Konferenz am HPI.

www.hpi.de

HPI und Stanford**Intensivprogramm Digitale Transformation**

Im September 2016 wollen das Stanford Center for Professional Development und das deutsche Hasso-Plattner-Institut (HPI) ihr erstes gemeinsames Intensivprogramm für Manager aus Industrie und Mittelstand starten.

Unter dem Titel »Leading Digital Transformation and Innovation« sollen dann an den Standorten Palo Alto und Potsdam sowie online neueste Kenntnisse aus den Bereichen Informationstechnologie, Wirtschaft und Design Thinking vermittelt werden.

»Wir bieten Führungskräften in diesem weltweit einzigartigen Programm eine besondere Mischung aus akademischen Inhalten und praktischen Erfahrungen rund um digitale Transformation und Innovation«, erläutert HPI-Direktor Prof. Christoph Meinel.

www.hpi.de

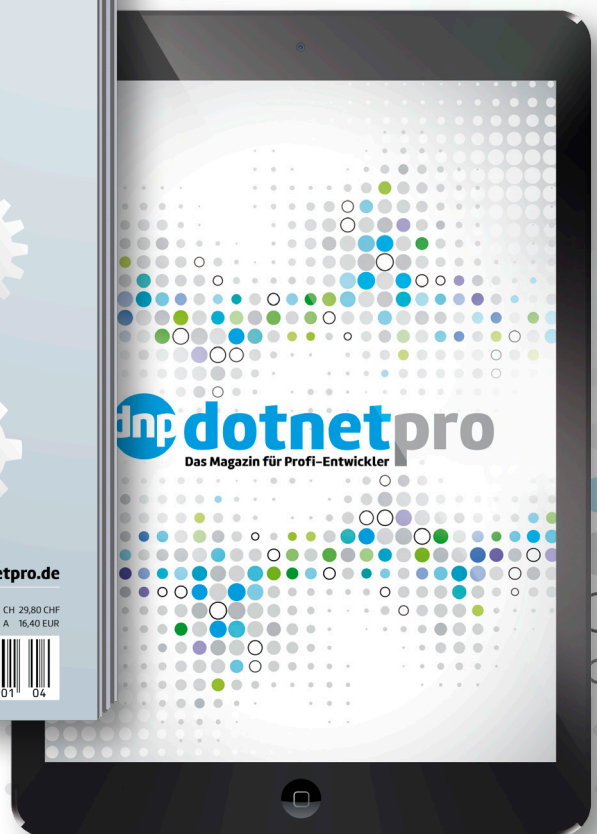


Innovationsmethode Design Thinking: Damit können Manager Initiativen zur digitalen Transformation entwickeln

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

Konvergenzfunktion

Ubuntu-Tablet
Aquaris M10

Der spanische Hersteller BQ stellt mit dem Aquaris M10 Ubuntu Edition das erste konvergente Gerät mit Ubuntu-PC-Erfahrung vor. Mit der Konvergenzfunktion lässt sich das Tablet wie ein Desktop-PC nutzen. Wird es mit einer Tastatur und einer Maus verbunden, kann es im Desktop-Modus verwendet werden, während die Touchfunktion auf dem Gerät im Tablet-Modus erhalten bleibt.



Foto: BQ

Das weltweit erste Tablet
mit einem Ubuntu-
Betriebssystem

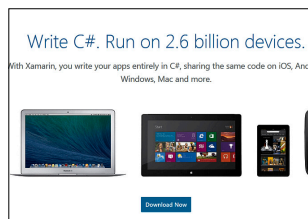
Durch die Kombination aus PC-Erfahrung und mobilem Gerät macht die Konvergenz das Tablet zu einem nützlichen Business-Tool. Die Oberfläche passt sich der Bildschirmgröße an. Das Aquaris M10 Ubuntu Edition ist mit einem 10,1-Zoll Full HD 10-Punkt Multi-Touch-Display und einem 7280-mAh-Akku ausgestattet. Es enthält einen MediaTek MT8163A Quad-Core-Prozessor mit einer Taktung von bis zu 1,5 GHz und einen MediaTek Mali-T720 MP2-Grafikprozessor. Mit einem Gewicht von 470 Gramm und einer Dicke von 8,2 Millimetern ist das Tablet sehr kompakt.

www.bq.com/de

Plattformübergreifend

Microsoft übernimmt
Xamarin

Das Unternehmen Xamarin steht für Entwickler-Tools, mit deren Hilfe man auf Basis von C#/XAML und .NET Programme verfassen kann, die sowohl auf Windows als auch auf Android,

Microsoft wird Xamarin
übernehmen

iOS und Linux laufen. Spezielle Tools wandeln die XAML-Beschreibung der Bedienoberfläche in nativen GUI-Code.

Scott Guthrie hat im offiziellen Microsoft-Blog bekannt gegeben, dass Microsoft das Unternehmen Xamarin übernehmen wird, sofern die zuständigen Kartellbehörden dem Kauf zustimmen. Xamarin ist das Unternehmen, das vor ein paar Jahren das Mono-Projekt fortgeführt hat, eine Portierung von .NET für Linux und Mac. Mit der Gründung von Xamarin nutzte das Unternehmen sein Know-how, um leistungsfähige Werkzeuge zu kreieren, mit denen man mit einer C#/XAML-Codebasis sowohl Windows- als auch Linux- und Mac-Programme schreiben kann. In den vergangenen Jahren wurden die Xamarin-Tools auf Android und iOS ausgeweitet. Schon seit Längerem arbeiten Microsoft und Xamarin eng zusammen.

Xamarin hat derzeit mehr als 15.000 Kunden in 120 Ländern, darunter mehr als 100 der 500 größten Unternehmen der Welt. Mehr als 1,3 Millionen Entwickler sollen die Xamarin-Tools bereits nutzen.

<http://xamarin.com>



IBM hat neue objektbasierte Cloud-Speicherdienste vorgestellt

IBM Cleversafe

Objektbasierte
Speicherservices

IBM hat neue, objektbasierte Cloud-Speicherdienste vorgestellt, die Kunden dabei helfen können, ihre wachsenden Mengen an polystrukturierten Daten leichter zu verwalten.

Der neue IBM Cloud Object Storage kombiniert Technologie von der kürzlich erworbenen Object-Storage-Firma Cleversafe mit der IBM Cloud. Unternehmen erhalten damit schnelle, flexible, hybride Cloud-Storage-as-a-Service-Lösungen. IDC schätzt, dass 80 Prozent aller neuen Cloud-Anwendungen datenintensiv sein werden. Cleversafe-Objektspeicher-Technologie ermöglicht es Kunden mit datenintensiven Anwendungen, bis in den Exabyte-Bereich zu skalieren.

Mit den IBM Cloud-Object-Storage-Services kann IBM Unternehmen die Auswahl aus multiplen APIs bieten und die Möglichkeit, riesige Datenmengen effizient on-premise, in der IBM Cloud oder in einer hybriden Umgebung zu speichern, und zwar mit dem gleichen Niveau an Zuverlässigkeit und Konsistenz der Technologie in allen Bereitstellungsmodellen.

»Eine der größten Herausforderungen für Unternehmen ist das massive Wachstum der Daten, angeheizt durch Cloud, Mobile, Internet der Dinge, Analytik, Social Media, kognitive und andere Technologien. Or-

ganisationen setzen auf diese Themen, um ihr Geschäft zu transformieren und mit Kunden in Verbindung zu treten«, sagt John Morris, Cleversafe-CEO. »Durch die Zusammenführung der Cleversafe-Technologie mit der Cloud können wir Kunden die Wahl und Flexibilität geben, die sie benötigen, um ihre Speicherkosten, den Wahl des Speicherorts und die Einhaltung von Kontrollanforderungen bezüglich ihrer Datensätze und wesentlichen Anwendungen besser auszubalancieren.«

www.cleversafe.com

Developer Week 2016

DevOps zahlt sich aus

Immer mehr deutsche Entwicklungsabteilungen setzen auf das Organisationsprinzip DevOps, um damit eine effektivere und konfliktfreiere Zusammenarbeit von Anwendungsentwicklung und IT-Betrieb zu erzielen. Dass die Einführung der DevOps-Maßnahmen in der Praxis tatsächlich fruchtet, hat die Entwicklerkonferenz Developer Week in ihrer aktuellen Umfrage unter Software-Entwicklern herausgefunden: Mehr



Developer Week 2016 veröffentlicht Ergebnisse aus Entwickler-Umfrage zu DevOps

als drei Viertel der deutschen Entwickler sind demnach der Meinung, dass seit der Anwendung von DevOps weniger Probleme beim Liveangang von Neuentwicklungen auftreten. Gleichzeitig werden Release-Zyklen verkürzt (68 Prozent) und die Produktivität in der IT insgesamt gesteigert (62 Prozent).

Immer komplexer werdende IT- und Software-Projekte stellen hohe Anforderungen an unternehmensinterne Organisationsstrukturen und Projektmanagement. Kein Wunder also, dass viele Entwicklungs- und IT-Abteilungen gerade nach Wegen suchen, um interne Prozesse zu optimieren. Im Zentrum steht dabei meist eine verbesserte und konfliktfreie Zusammenarbeit von Entwicklern und Administratoren. DevOps liefert dazu einen Maßnahmenkatalog, der beide Abteilungen näher aneinander heranrückt, um kürzere Release-Zyklen und am Ende bessere Produkte zu erhalten. Die Developer Week hat dazu von November 2015 bis Januar 2016 eine Umfrage unter 165 Software-Programmierern durchgeführt.
<http://developer-week.de/Programm>

Galaxy S7 / Galaxy S7 edge Neue Flaggschiffe von Samsung

Ein weiteres Mal setzt Samsung mit seinen neuen Flaggschiffen Galaxy S7 und dem Galaxy S7 edge Maßstäbe im Smartphone-Premiumsegment.

Insbesondere in puncto Kamera stellen die Flaggschiff-Smartphones einen neuen Standard dar. Das Galaxy S7 und das S7 edge besitzen eine neue Dual-Pixel-Technologie, die für einen blitzschnellen Autofokus und dank der F1.7-Offenblende auch bei dunklen Lichtverhält-

nissen für gestochen scharfe Bilder sorgt. Weitere Highlights sind die IP68-Zertifizierung und ein leistungsfähiger Akku, der mit besonders viel Ausdauer auch ausgiebige Gaming-Sessions ermöglicht.

Auch Fehler der Vergangenheit wurden revidiert: Die neuen Modelle verfügen wieder über einen microSD-Slot, der Karten mit einer Kapazität bis 200 GByte aufnehmen kann.

Das Display ist 5,1 Zoll groß und bietet eine Auflösung von 2560 x 1440 Pixeln. Das gekrümmte Display des S7 edge



Samsung präsentierte auf dem Mobile World Congress das neue Galaxy S7

ist bei gleicher Auflösung mit 5,5 Zoll etwas größer. Im Inneren verwendet das S7 einen hauseigenen 64-Bit-Achtkernprozessor und 4 Gigabyte Arbeitsspeicher.

www.samsung.com

Max-Planck-Institut Fehlerfreie Software

Die Vision von Derek Dreyer vom Max-Planck-Institut für Softwaresysteme ist eine Programmiersprache, bei der fehlerhaftes Entwickeln unmöglich ist. Auf dem Weg dorthin setzt er auf Mozillas Programmiersprache Rust.

Die beliebtesten Webbrowser heißen Internet Explorer, Safari, Firefox oder Chrome. Millionen von Laien nutzen sie, um im World Wide Web Überweisungen zu tätigen, einzukaufen oder sich Filme anzuschauen. Umso gefährlicher ist es daher, wenn ►

Bitkom

10 Milliarden Euro Umsatz mit Smartphones

Die Nachfrage nach Smartphones hat im vergangenen Jahr noch einmal kräftig angezogen. 2015 wurden in Deutschland 26,2 Millionen Geräte verkauft, das ist ein Plus von 7,4 Prozent gegenüber dem Vorjahr. Der Umsatz stieg sogar um 22 Prozent auf 10,3 Milliarden Euro und knackte damit erstmals die symbolische Marke von 10 Milliarden Euro. Das berichtet der Digitalverband Bitkom auf Grundlage aktueller Prognosen des European Information Technology Observatory (EITO).

Der Durchschnittspreis eines Smartphones lag 2015 bei 395 Euro, im Jahr zuvor waren es erst 348 Euro. Befeuert wurde der Markt im vergangenen Jahr durch die Einführung neuer Modelle großer Hersteller wie das iPhone 6 oder das Samsung Galaxy S6.

»Der Smartphone-Boom geht den Bitkom-Prognosen zufolge auch 2016 weiter«, konstatiert Dr. Hannes Ametsreiter, Mitglied des Bitkom-Präsidiums. So werden im laufenden Jahr voraussichtlich 28,2 Millionen Geräte verkauft, das ist ein Plus von 7,5 Prozent im Vergleich zu 2015. Der Umsatz soll bei rund 10,4 Milliarden Euro liegen. Der Durchschnittspreis für ein Smartphone würde damit wieder etwas sinken, auf 370 Euro.

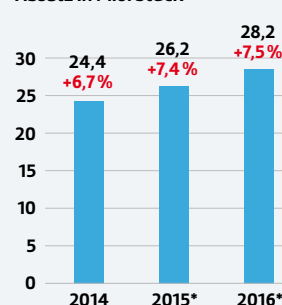
Die guten Verkäufe spiegeln sich auch bei den Nutzerzahlen: Rund drei Viertel der Deutschen ab 14 Jahren (74 Prozent) verwenden heute ein Smartphone.

Im Schnitt verwenden die Smartphone-Nutzer ihr Gerät jeden Tag 82 Minuten. Neben der Standardfunktion Telefonieren (99 Prozent) nutzen 89 Prozent die Foto- oder Videokamera, 80 Prozent Suchmaschinen und 72 Prozent den Kalender beziehungsweise Terminplaner. 69 Prozent hören Musik über das Smartphone und weitere 67 Prozent greifen mit ihrem Mobiltelefon auf soziale Netzwerke zu. Rund die Hälfte liest oder schreibt zudem SMS (55 Prozent) sowie andere internetbasierte Kurznachrichten wie WhatsApp oder Threema (50 Prozent) und E-Mails (55 Prozent). »Mobile first ist das Motto. Wer fünf Minuten auf den Bus oder die U-Bahn warten muss, geht in dieser Zeit mal eben schnell ins Internet und recherchiert oder erledigt etwas«, so Ametsreiter. Entsprechend sagen 67 Prozent der Smartphone-Nutzer: »Smartphones sind für mich eine große Erleichterung im Alltag.«

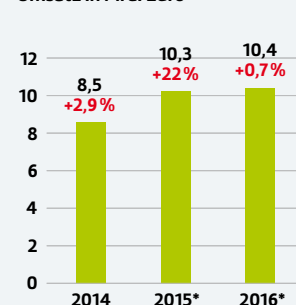
www.bitkom.org

Das Geschäft mit Smartphones boomt

Absatz in Mio. Stück



Umsatz in Mrd. Euro



Der Smartphone-Boom hält unvermindert an.

web & mobile developer 5/2016

*vorläufige Schätzung/Prognose
Quelle: EITO, IDC

Thinktecture

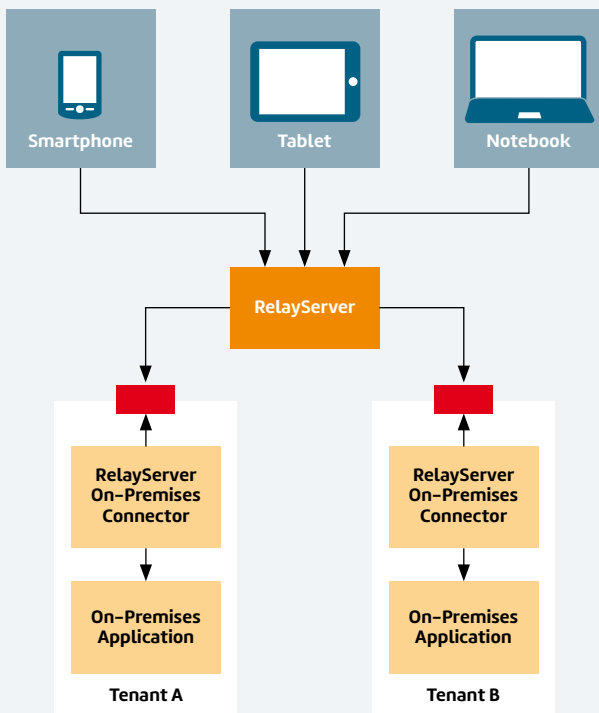
Open-Source-Projekt RelayServer

Das neue Open-Source-Projekt RelayServer der Thinktecture AG ermöglicht Nutzern vieler klassischer Unternehmensanwendungen, die ihre Daten im lokalen Netzwerk abspeichern, von Smartphones und Tablets mobil auf ihre Daten zuzugreifen. Der RelayServer kann dazu von Software-Entwicklern leicht in bestehende Anwendungen integriert werden. Eine Datenspeicherung in der Cloud wird durch diesen Ansatz vermieden. Das System ist laut Hersteller vollständig mandantenfähig, sodass mit einem RelayServer die On-Premises-Applikationen beliebig vieler Kunden angebunden werden können. Durch ein integriertes Management-Web können neue Mandanten sehr einfach angelegt und verwaltet werden.

Der RelayServer kommt ohne spezielle Libraries für den Client aus und funktioniert daher gleichermaßen für Browser-Applikationen, Cross-Plattform-Apps wie auch native Apps. Christian Weyer, Vorstand der Thinktecture AG, sagte im Rahmen des Release-Termins: »Wir sehen in den letzten Jahren in der DACH-Region im Rahmen unserer Beratungsarbeit eine hohe Nachfrage nach Cross-Plattform-Lösungen, die Anwendern den mobilen Zugriff auf Unternehmensdaten ermöglichen. Eine Datenhaltung in der Cloud ist dabei aus rechtlichen Gründen oder wegen der Komplexität der Migration häufig ein Hindernis für angedachte Projekte. Mit dem Open-Source-Projekt RelayServer geben wir Unternehmen eine Alternative an die Hand, die sich leicht in neue und bestehende Projekte integrieren lässt.«

<https://github.com/thinktecture/relayserver>

Mobiler Zugriff auf Unternehmensdaten ohne Cloud



Der RelayServer kommt ohne spezielle Libraries für den Client aus.

web & mobile developer 5/2016

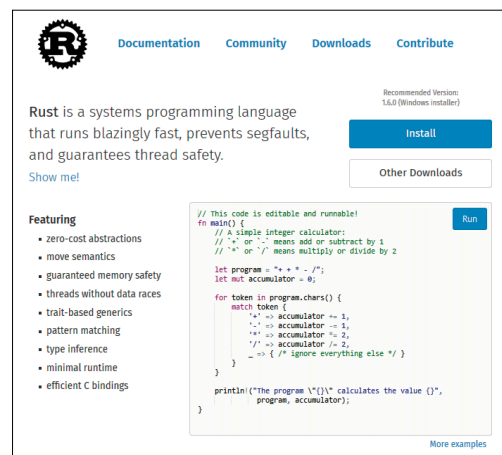
Quelle: Thinktecture

in den Browsern Programmierfehler schlummern, die als Sicherheitslücken ausgenutzt werden können. Solche Programmierfehler entstehen zwangsläufig, da Programmierer komplexen, parallel ablaufenden Programmcode schreiben müssen. Nur so können sie sicherstellen, dass die Browser ihren Anwendern in kürzester Zeit eine Rückmeldung liefern. »Manche Arbeiten kann man nur mit einem besonders

fortgeschritten, dass man mit ihr auch den neuartigen Webbrowser Servo entwickeln, der Firefox ersetzen könnte. Rust gewährt Programmierern eine feinkörnige Kontrolle über das Speichermanagement und die Zuteilung von Ressourcen.

Aber auch Rust ist nicht ohne Schwachstellen. Bereits im vergangenen Jahr wurden einige entdeckt. Daher begrüßen die Entwickler von Rust Dreyers Vorhaben, systematisch ver-

Auf der Basis von Rust soll die perfekte Programmiersprache entstehen



scharfen Messer erledigen und dabei auch zu tief schneiden. Bei den Programmiersprachen ist es ähnlich. Software, die auf schnellem, parallel ausgeführtem Code basiert, birgt auch die Gefahr brisanter Programmierfehler«, erklärt Derek Dreyer, der Fakultätsmitglied des Max-Planck-Instituts für Softwaresysteme ist, das Standorte in Saarbrücken und Kaiserslautern hat.

In der Saarbrücker Dependence des Instituts leitet Dreyer die Gruppe »Grundlagen der Programmierung« und lehrt an der Universität des Saarlandes Informatik. Seine Vision ist eine Programmiersprache, bei der fehlerhaftes Entwickeln unmöglich ist. Um diese zu verwirklichen, setzt er auf die Programmiersprache Rust. Hinter ihr steht die Mozilla Foundation. Seit 2010 arbeitet sie an Rust. Inzwischen ist die Sprache so

bliebene Fehler aufzuspüren. Um diese zu finden oder ihre Abwesenheit nachzuweisen, müssen die Forscher neue Beweismethoden entwickeln. Dreyer und sein Team sind damit weltweit die Ersten, die sich im Bezug auf Rust dieser Herausforderung stellen. Die Informatiker erforschen daher, wie die letzten Komponenten von Rust, deren Sicherheit noch nicht nachgewiesen ist, fehlerlos arbeiten können.

Der Europäische Forschungsrat (ERC) unterstützt Dreyer mit dem sogenannten Consolidator Grant. Mit ihm erhalten erfahrene Wissenschaftler die Möglichkeit, fünf Jahre mit einem Budget von rund zwei Millionen Euro ein Forschungsprojekt umzusetzen. Voraussetzung dafür ist laut ERC die wissenschaftliche Exzellenz des Forschers und des Projektantrags.

www.rust-lang.org



Developer Week 2016

20.-23. Juni 2016,
Messe Nürnberg



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

Mit Code

DWX16wmd

€ 149,-
sparen!



Keynote Speaker
Jurgen Appelo
Managing
for Happiness



Keynote Speaker
Scott Hanselman
Java Script, The Web, Mobile and
the Rise of the New Virtual Machines

developer-week.de



DeveloperWeek

Aussteller & Sponsoren:

adorsys



brainLight®
LIFE IN BALANCE

FR
Fast Reports
Reporting must be Fast!

Veranstalter:

developer
media

Neue
Mediengesellschaft
Ulm mbH

GU
GFU Cyrus AG



MAXIMAGO

MATHEMA

SDX

Präsentiert von:

SHS VIVEON

SOS
SOFTWARE SERVICE
VALUE ADDED DISTRIBUTION

TEXTCONTROL

hello, technology

UID

dotnetpro

web & mobile
DEVELOPER

Digitalisierung

Deutschland droht den Anschluss zu verlieren

Die Digitalisierung beschäftigt mittlerweile jede Branche. Doch die Tragweite für das eigene Kerngeschäft und konkrete Herausforderungen des Themas werden in deutschen Unternehmen nicht ausreichend erkannt.

Gleichzeitig scheitern Digitalisierungsvorhaben an Management- und Umsetzungsfehlern. Zu diesem Ergebnis kommt eine Befragung von Führungskräften und Vorständen unter den rund 2000 Großunternehmen in Deutschland, welche die Digitalberatung etventure gemeinsam mit der GfK Nürnberg durchgeführt hat.

Immerhin jedes vierte der befragten Großunternehmen gibt an, dass der Stellenwert der digitalen Transformation in den vergangenen 12 Monaten deutlich gestiegen ist. Ein weiteres Drittel stellt einen gestiegenen Stellenwert fest. In nicht einmal der Hälfte der Unternehmen (48 Prozent) ist die Umsetzung der digitalen Transformation durch Mitglieder des Vorstands oder der Geschäftsführung gesteuert. In fast drei Viertel der Unterneh-



Nur ca. **35 %** haben »Digitalisierung« auf ihrer Top-3-Agenda



Knapp **50 %** der Führungskräfte scheuen vor notwendigen radikalen Entscheidungen zurück



Bei weniger als **50 %** steuern die Vorstände oder Geschäftsführer selbst digitale Veränderungsprozesse



Bei **65 %** verhindern veraltete Strukturen die Digitalisierung

Deutschland droht in der Digitalisierung den Anschluss zu verlieren.

men (76 Prozent) ist allerdings entweder die hauseigene Unternehmensentwicklung oder die IT-Abteilung mit der Digitalisierung beauftragt.

Die Untersuchung von etventure enthüllt erstmals dezidiert, was die größten Hemmnisse in den Unternehmen bei der digitalen Transformation sind. Mit deutlichem Abstand und 65 Prozent Nennung steht an erster Stelle die Verteidigung bestehender Strukturen in den Unternehmen. Je größer die befragten Firmen sind, desto häufiger wird dieser Aspekt angeführt. Als weitere Hürden werden fehlende Zeit (54 Prozent) sowie fehlende Erfahrung (52 Prozent) genannt. Gut 40 Prozent sagen überdies, dass notwendige weitreichende und radikale Entscheidungen von den Führungskräften gescheut werden oder man zu festgefahren sei.

www.etventure.de/deutschlandstudie



Das Smartphone Cat S60 verfügt über eine Infrarotkamera

Cat S60 um ein extrem widerstandsfähiges Premium-Rugged Smartphone, das strenge Militärstandards erfüllt. Stürze aus 1,80 Metern Höhe auf Beton übersteht es, zudem ist es sowohl staub- als auch wasserdicht und dank Corning Gorilla Glass 4 bestens gegen Kratzer und Displaybruch geschützt. Sein Display soll sich sogar bei direkter Sonneneinstrahlung noch gut lesen lassen.

www.bullitt-group.com

Cat S60**Infrarotkamera integriert**

Die integrierte Thermalkamera von FLIR erlaubt Besitzern des neuen Cat S60 neue Einsatzmöglichkeiten. So lässt sich damit unter anderem das Entweichen von Wärme rund um schlecht isolierte Fenster und Türen erkennen, oder auch das Risiko einer Überhitzung von elektrischen Geräten und Schaltkreisen.

Die Thermalkamera zeigt Wärme an, die für das Auge nicht erkennbar ist, und visualisiert Temperaturunterschiede. Dabei kann sie Oberflächen Temperaturen bei einem Abstand von bis zu 30 Metern messen, selbst bei Nebel oder Rauch. Die Einsatzmöglichkeiten für Handwerker im Baugewerbe, Mitarbeiter von Versorgungsbetrieben, Feuerwehrleute und Rettungsdienste, aber auch für Outdoor-Sportler sind damit unerschöpflich.

Aufgrund seiner Wasserdichtigkeit lässt sich das Cat S60 sogar als Unterwasserkamera nutzen – Tiefen von bis zu 5 Metern soll es laut Hersteller eine Stunde lang unbeschadet überstehen. Vorgestellt hat das Cat S60 die Bullitt Group, sie ist weltweiter Lizenzinhaber der Marke Caterpillar für mobile Produkte.

Abgesehen von seinem Pionierstatus im Bereich Thermal Imaging handelt es sich beim

Sprachsteuerung**Ford präsentiert das SYNC 3-System**

Das neue Kommunikations- und Entertainmentsystem wird ab dem Sommer in Europa für die Ford-Baureihen Mondeo, S-MAX sowie Galaxy gegen Aufpreis erhältlich sein.

Mit SYNC 3 steuern Autofahrer ihr angeschlossenes Smartphone sowie Audio-, Navigations- und Klimatisierungsfunktionen ganz einfach per



Mit Ford SYNC 3 lassen sich während der Fahrt zahlreiche Smartphone-Apps mittels Sprachbefehl nutzen

Sprachbefehl. Ford SYNC 3 verfügt zudem über einen neu gestalteten 8-Zoll-Touchscreen, dessen Bedienung an die eines Smartphones oder Tablets angelehnt ist.

www.ford.de



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.

Apache Spark

Cloud Dataflow

Python

Speicherung, Analyse, Plattformen und Tools in den Konferenz-Sessions:

- Apache Spark
- Cloud Data Platform
- Google Dataflow
- MapReduce
- Python
- SQL Server Integration
- Und vieles mehr

Die Workshops bieten intensives Training in diesen Technologien:

- Microsoft Azure
- Multi-Model-Database mit CQRS und node.js

smart-data-developer.de #smartddc



SMARTDATADeveloperConference

Präsentiert von:



web & mobile
DEVELOPER

Veranstalter:



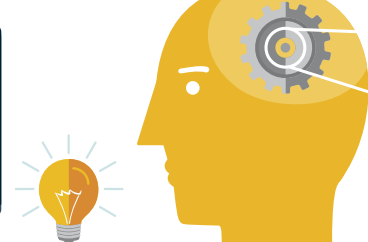
Neue
Mediengesellschaft
Ulm mbH



SMART DATA

Developer Conference

Big Data Analytics



Konferenz-Programm: 18. April 2016

		Blog		
09.00 Uhr	Clouds in the Wind – Big und Smart Data in der Cloud und trotzdem beweglich • <i>Michael Nolting</i>			
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 UhrKaffeepause				
10.30 Uhr	Smart Data mit .NET	Mehr als SQL Server in the Cloud • <i>Constantin Klein</i>	Best Practices	Datenqualität für Entwickler • <i>Werner Keil</i>
	Gewinnen Sie in dieser Session einen Überblick über die Microsoft Cloud Data Platform und die Möglichkeiten beider Bereiche „Data & Storage“ und „Analytics“.		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 • <i>Thomas Worm</i>	Best Practices	CQRS und Multi-Model-DB • <i>Jan Fellien</i>
	Die SQL Server Integration Services ermöglichen die Integration von Datenbanken in vorhandene IT-Landschaften. Die Session behandelt Framework und Eigenheiten der SSIS.		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 UhrMittagspause				
13.30 Uhr	Smart Data mit .NET	Verteilte Algorithmen am Beispiel MapReduce • <i>Sascha Dittmann</i>	Best Practices	Das Analytics Framework von OTTO • <i>Rupert Steffner und Jan Lendholt</i>
	MapReduce kann sowohl als eigenständiges System als auch als Abfragesprache implementiert werden. Gängige Problemstellungen von verteilten Algorithmen werden umgesetzt.		Fokus der Session ist das Streaming Analytics Framework und dessen containerized Micro-Service-Architektur. Gezeigt werden Design Patterns und Use Cases.	
14.10 Uhr	Smart Data mit .NET	Datenanalyse mit Python • <i>Johannes Hofmeister</i>	Best Practices	Google Dataflow: Batch&Stream Processing • <i>Alexander Albrecht</i>
	Python bietet für die Datenanalyse einen interessanten Kompromiss zwischen allgemeinen Programmiersprachen und statistischer Auswertung. Mit Fokus auf komplexe Analysen.		Dataflow aka Apache Beam synthetisiert die Verarbeitung von Echtzeit- und Batch-Daten. Der Vortrag stellt die zentralen Konzepte und Möglichkeiten der Technologie vor an Beispielen.	
14.50 Uhr	Smart Data mit .NET	Interaktive Analysen mit Apache Spark • <i>Olivia Klose und Sascha Dittmann</i>	Best Practices	Performance trotz Entity Framework • <i>André Krämer</i>
	Bei interaktiven Datenabfragen schwächeln die meisten Big Data Technologien. Diese Session zeigt die wichtigsten Grundlagen der In-Memory Technologie anhand von Codebeispielen.		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!	
15.30 UhrKaffeepause				
16.00 Uhr	Smart Data mit .NET	Looking for the meaning to our data • <i>Diego Poza</i>	Mensch & Maschine	The UX of Data: Responsive Visualisierung • <i>Peter Rozek</i>
	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.		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! • <i>Olivia Klose und Marcel Tilly</i>	Mensch & Maschine	Ergonomische Datendarstellung • <i>Daniel Greitens</i>
	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?		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 und Referenten

Workshop 1

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

[Blog](#)

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 Data with Microsoft Azure

[Blog](#)

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:



Alexander Albrecht
bakdata GmbH



Sascha Dittmann
Microsoft Deutschland GmbH



Jan Fellien,
devCrowd GmbH



Daniel Greitens
MAXIMAGO GmbH



Johannes Hofmeister
Universität Heidelberg



Werner Keil
Creative Arts & Technologies



Constantin Klein
Freudenberg
IT GmbH & Co. KG



Olivia Klose
Microsoft Deutschland GmbH



André Krämer
Software, Training
& Consulting



Jan-Hendrik Lendholt
Otto (GmbH & Co KG)



Michael Nolting
Sevenval
Technologies GmbH



Diego Poza
Auth0 Inc.



Peter Rozek
ecx.io germany GmbH



Rupert Steffner
Otto (GmbH & Co KG)



Marcel Tilly
Microsoft Research



Thomas Worm
DATEV eG



Mathias Vietmeier

GRAPH-DATENBANKEN MIT NEO4J

Strukturen als Graphen

Alle Strukturen der realen Welt lassen sich als Graphen interpretieren.

Graph-Datenbanken eignen sich besonders gut für das automatische Speichern, Abfragen und Verarbeiten grafischer Datenstrukturen.

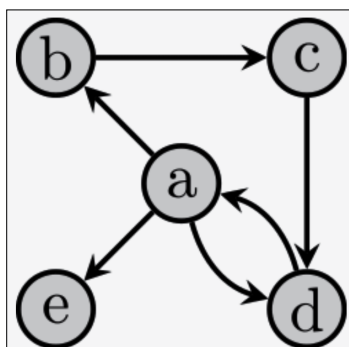
Graphen findet man in sehr vielen Anwendungsfällen; sie stellen eine mathematische Repräsentation von Objekten und ihrer Beziehungen dar. In manchen Fällen, wie Verkehrsnetzen, Beziehungsnetzen zwischen Personen, Reaktionsnetzen biochemischer Vorgänge oder Stücklisten der Fertigungsindustrie, ist das Graphenmodell unmittelbar ersichtlich. In anderen Fällen, wie der objektorientierten Modellierung oder der Hyperlink-Struktur des Webs, kann man Graphen auch durch Interpretation schematisiert aufbauen.

Insbesondere bei komplexen beziehungsorientierten Fragestellungen wie der Ableitung von Empfehlungen bei der Betrachtung eines Produkts durch einen Inte-

ressenten können Graphen behilflich sein. Mit relationalen Datenbanken stößt man bei der Verarbeitung von Graphen mit vielen Beziehungen schnell an Grenzen, da diese in ihrem Datenmodell den Fokus vollkommen auf Attributwerte,

also Eigenschaften, und nicht auf die modellierte Struktur (den Graphen) legen. Für die Speicherung von Graphen benötigt man ein Datenbanksystem, das besonderen Wert auf die Beziehungen zwischen den beteiligten Entitäten legt – wie es bei einer Graph-Datenbank der Fall ist.

Eine Graph-Datenbank stellt die Beziehungen zwischen den Entitäten in den Fokus ihres Datenmodells. Im Unterschied zur relationalen Datenbank, bei der sich die Verknüpfungen erst implizit durch Interpretation der Attributwerte ergeben, sind sie bei einer Graph-Datenbank expliziter, das heißt bereits inhärenter Bestandteil der Daten.



Ein gerichteter Graph mit fünf Knoten und sechs Kanten (Bild 1)

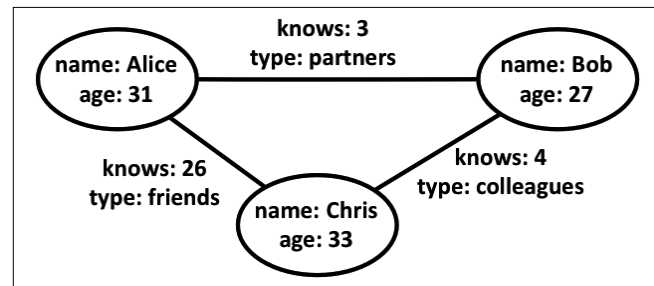
Als ein Teilgebiet der Mathematik und Informatik befasst sich die Graphentheorie (Euler, Kirchhoff, König, Petri) schon seit fast drei Jahrhunderten mit Graphen, ihren Eigenschaften und Beziehungen. Unter einem Graphen versteht man eine abstrakte Struktur, die eine Menge von Objekten zusammen mit den zwischen diesen Objekten bestehenden Verbindungen repräsentiert. Abstrakt bezeichnet man die Objekte auch als Punkte, Ecken oder Knoten.

Die paarweisen Verbindungen zwischen den Knoten stellen Linien beziehungsweise Bögen dar, sie heißen auch Kanten. Einen Graphen veranschaulicht man durch eine Abbildung, bei der man die Knoten/Punkte durch Kreise und die Kanten durch Pfeile oder Linien zeichnet (**Bild 1**).

Mathematisch gesehen, handelt es sich bei einer Kante um eine Menge aus genau zwei Knoten. So bedeutet $\{a, b\}$ oder (a, b) eine Kante zwischen den Knoten a und b . Kanten sind im Normalfall ungerichtet, können aber auch gerichtet sein, was ein Pfeil anstatt einer Linie kennzeichnet. Eine gerichtete Linie, deren Richtung von b nach a führt, nennt man Pfeil, und sie wird durch das Knotenpaar (b, a) angegeben.

Bewegt man sich in einem Graphen von einem Knoten a zum Knoten z , so beschreitet man einen Weg von a nach z . In der Informatik verwendet man für den Begriff des Weges häufig die Bezeichnung Pfad; in Anlehnung an den englischen Begriff Path. Ein Knoten ohne Vorgänger im Graphen heißt Wurzel; während man Knoten ohne Nachfolger Blätter nennt.

Manchmal spricht man anstatt von Weg auch von einem Kantenzug. Die Knoten a und z nennt man Endknoten des Weges; Knoten, die keine Endknoten sind, heißen innere Knoten des Weges. Ein Weg, bei dem Start- a und Endknoten x identisch sind, heißt Zyklus oder Kreis. Beim Beschreiten des Weges spricht man auch vom Traversieren des Graphen; dies erfolgt über seine Knoten a, b, c, \dots, z ; die in ihrer Gesamtheit Knotenfolge genannt wird. Unter der Länge eines Wegs von a nach z versteht man die Anzahl der Kanten. Als kürzesten



Bei einem **Property-Graphen** besitzen sowohl Knoten als auch Kanten Eigenschaften beziehungsweise Attribute, die konkrete Werte annehmen können (**Bild 2**).

Weg von einem Knoten s zu einem Knoten t in einem Graphen bezeichnet man einen Pfad/Kantenzug von s nach t , dessen Länge minimal ist. Die Tiefe eines Knotens v entspricht der Länge des Weges von einem Wurzelknoten r nach v .

Das erweiterte Graphenmodell von Neo4j seit Version 2

Eine Graph-Datenbank verwaltet einen oder mehrere Graphen, alle zugehörigen Elemente inklusive deren expliziter Struktur: Jeder Knoten kennt alle seine benachbarten Knoten. Das Datenmodell des Graphen ist recht einfach und lässt sich daher leicht visualisieren. Tatsächlich handelt es sich bei der Modellierung der realen Welt durch einen Graphen um das der Modellwelt tatsächlich immanente Datenmodell. Deshalb lässt sich im Prinzip jeder zu modellierende Sachverhalt als Menge von Knoten auffassen, die durch Beziehungen miteinander verknüpft sind.

Beziehungen (Relationen) modelliert man über Kanten direkt am Knoten; bei beiden Konstrukten handelt es sich im Unterschied zur Relationentheorie um gleichwertige Elemente. Eine Graph-Datenbank in Neo4j besteht aus Knoten (Nodes) und Beziehungen (Relations) mit Eigenschaften (Properties) bei Knoten und Kanten.

Das Datenmodell von Neo4j erweitert also einen Graphen um zusätzliche Eigenschaften (die Properties) an Knoten und Kanten. Sowohl Knoten als auch Verbindungen können beliebige Mengen von Attribut-Wert-Paaren in den Eigenschaften (Properties) enthalten. Deshalb spricht man beim Datenmodell von Neo4j auch von einem Property-Graphen (**Bild 2**).

Neo4j speichert Daten in Knoten, diese enthalten Attribut-Wert-Paare; gelegentlich auch Schlüssel-Wert-Paare genannt. Die Knoten wiederum sind über gerichtete, benannte (getypte) Verbindungen (Beziehungen) miteinander verknüpft; die ebenfalls Daten (Attribut-Wert-Paare) enthalten. Jeder Knoten kann über eine beliebige Anzahl von Beziehungen mit einem anderen verbunden sein.

Beziehungen können gleichwertig in jede Richtung traversiert werden, das heißt, dass eine Modellierung der entgegengesetzten Richtung nicht erforderlich ist und entfallen kann. Im Unterschied zu einfachen allgemeinen Graphen sind Property-Graphen komplexer, gleichzeitig aber auch leistungsfähiger, da sie aus verschiedenen Typen von Knoten und Kanten bestehen und somit mehr Semantik umfassen. ►

ACID-Prinzip im Überblick

Relationale Datenbank führen alle Transaktionen gemäß dem ACID-Prinzip durch.

Die Abkürzung ACID steht für:

- **Atomacy (Atomarität):** Eine Datenbank führt eine Transaktion immer vollständig oder gar nicht aus.
- **Consistency (Konsistenzerhaltung):** Die Daten der Datenbank befinden sich vor und nach Ausführung der Transaktion immer in einem konsistenten Zustand.
- **Isolation (Abgrenzung):** Transaktionen können parallel und voneinander unabhängig ausgeführt werden; sie beeinflussen sich nicht gegenseitig.
- **Durability (Dauerhaftigkeit):** Jede erfolgreich ausgeführte Transaktion speichert ihre Daten dauerhaft über eine Commit-Operation; im Fehlerfall macht eine Abort/Rollback-Operation die Transaktion wieder rückgängig.

Zudem können Knoten in Neo4j seit Version 2.0 mit keinem, mit einem oder mit mehreren Labels (Marken) versehen sein, um Untermengen (Subsets) zum Beispiel für verschiedene Rollen der Anwendungsdomäne zu bilden. Labels besitzen den Charakter von Metadaten und können als Marken oder Kategorien aufgefasst werden; sie bilden Indexe oder einzuhaltende Bedingungen (Constraints) ab.

Kanten als Beziehungen modellieren semantische Verknüpfungen zwischen den beiden Knoten. Eine Beziehung besitzt eine Richtung, einen Typ, einen Start- und einen Endknoten. Während die Eigenschaften von Knoten eher qualitative Datenwerte annehmen, handelt es sich bei den Eigenschaften der Kanten häufiger um quantitative Datenwerte wie Kosten, Entfernungen oder Gewichte.

Relationale Datenbanken im Vergleich mit Graph-Datenbanken

Ein wesentliches Unterscheidungsmerkmal einer relationalen Datenbank gegenüber einer Graph-Datenbank liegt in ihren Verarbeitungseinheiten: Während Erstere ganze Mengen von Datensätzen aus den Tabellen verarbeitet, basiert die

Rank			DBMS	Database Model	Score		
Jan 2016	Dec 2015	Jan 2015			Jan 2016	Dec 2015	Jan 2015
1.	1.	1.	Neo4j	Graph DBMS	33.01	-0.18	+8.58
2.	3.	4.	OrientDB	Multi-model	6.09	+0.30	+3.56
3.	2.	3.	Titan	Graph DBMS	5.65	-0.33	+2.74
4.	4.	2.	Virtuoso	Multi-model	3.04	-0.15	+0.04
5.	5.	7.	ArangoDB	Multi-model	1.41	+0.03	+0.79
6.	6.	6.	Giraph	Graph DBMS	0.93	+0.02	+0.17

Neo4j besitzt als einziges Graph-Datenbanksystem nennenswerte Marktanteile (Bild 4)

Wichtige Konfigurationsdateien von Neo4j

Im **conf**-Unterverzeichnis der Neo4j-Installation gibt es verschiedene Dateien, um die Datenbank oder das Datenbank-Management-System zu konfigurieren.

- **neo4j-server.properties:** Hierüber stellt man das Verhalten zur Laufzeit des Servers (Port, Security, IP-Adresse, Thread-Anzahl, HTTPS) ein und trifft Vorgaben für die Datenbank (Name, Verzeichnis) und das Logging (Einschalten, Dateiname).
- **neo4j.properties:** Enthält die Einstellung zum Tuning der Datenbank (Cache, Buffer, Cypher-Parser, Database-Upgrade, Index, Checkpoints, Logging).
- **neo4j-wrapper.conf:** Einstellungen für die Umgebung und Startparameter des Neo4j-Servers (JVM-Tuning-Parameter).
- **neo4j-http-logging.xml:** Konfiguration des Logging-Systems (Format, Policy) für den von Neo4j bereitgestellten HTTP-REST-Server.
- **ssl:** Dieses Unterverzeichnis umfasst die Zertifikationsdateien für HTTPS-Operationen mit dem REST-Server.
- **data/log:** In diesem Unterverzeichnis der Neo4j-Installation stehen die Log-Dateien: *console.log*, *http.log* und *wrapper.log*.

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Die **Storage-Engine** einer Graph-Datenbank kann direkt auf alle Nachbarn eines Knotens ohne zusätzlichen Index zugreifen (Bild 3)

Verarbeitung in einer Graph-Datenbank auf Pfaden oder anders ausgedrückt auf Wegen/Kantenfolgen. Ein Graph-Datenbanksystem verarbeitet direkt Beziehungen und kennt diese als eigenständige Verarbeitungseinheit. Soll ein relationales Datenbanksystem alle an einer Beziehung teilnehmenden Elemente verarbeiten, so muss dieses in der Regel eine Verbundoperation (Join) zwischen den an der Beziehung beteiligten Tabellen ausführen.

Im Unterschied zur Graph-Datenbank findet beim Join in der relationalen Welt ein zusätzliches, künstliches und wertbezogenes Mapping über Fremdschlüssel statt. Im Graphen erübrigt sich diese Verbundoperation und kann vollständig entfallen. Daraus resultiert beim Traversieren des Graphen eine wesentlich höhere Performance gegenüber einer relationalen Datenbank. Graph-Datenbanken liefern selbst bei Wegen mit großer Tiefe und vielen beteiligten Knoten (Bild 3) noch akzeptable Antwortzeiten – während eine relationale Datenbank in einem solchen Fall sich für die Praxis bereits als untauglich erweisen kann. Für die Ausführung der Joins resultiert ein immenser Overhead, der sich negativ auf den Durchsatz auswirkt. Hinzu kommt die schiere Unfähigkeit einer relationalen Datenbank, rekursive Beziehungen performant zu verarbeiten.

Tabellen und Beziehungen einer relationalen Datenbank basieren immer auf einem Schema. Das Schema definiert die Struktur der Daten im Voraus und kommt bei der Ablage ihrer Werte zum Einsatz. Dabei prüft das Datenbanksystem, ob die Integrität der Daten, die auch über das Schema definiert ist, gewahrt bleibt. Eine relationale Datenbank eignet sich besonders für die Verarbeitung gut strukturierter Daten, bei denen Konsistenz, Korrektheit und Vollständigkeit wichtige Kriterien darstellen. Im Gegensatz dazu besitzen Graphen kein Schema, sie arbeiten orientiert an der aktuellen Ausprägung der Daten und setzen keine fest vorgegebene Struktur voraus. Eine Graph-Datenbank speichert Daten und ihre Beziehungen entsprechend ihrem Auftreten beziehungsweise Entstehen. Sie eignet sich besonders gut für Modelle der realen Welt, in denen Beziehungen eine sehr große Bedeutung innehaben.

Vom Start-up zum Key-Player durch wichtige Leistungsmerkmale

Auslöser für die Entwicklung von Neo4j im Jahre 2000 war die mangelhafte Performance einer relationalen Datenbank-Anwendung, die man mit einer Graph-Datenbank beseitigen wollte. Die Programmierung von Neo4j erfolgte in Java und

fand fast sieben Jahre lang im universitären Umfeld statt. Im Jahr 2007 startete Neo Technology mit den Firmengründern Emil Eifrem und Johan Svensson ein Start-up-Unternehmen in Malmö, Schweden, mit dem Vertrieb der Graph-Datenbank Neo4j. Das erste offizielle Release Neo4j-Version 1.0 erschien Anfang 2010 als Open-Source-Software. Mit zunehmendem Erfolg verlegte Neo Technology im Jahr 2011 seinen Hauptsitz von Schweden nach San Mateo, Kalifornien, USA und eröffnete neben Malmö weitere Niederlassungen in England, Deutschland und Frankreich.

Seit Längerem nimmt Neo4j mit über 33 Prozent Marktanteil den ersten Platz im Ranking der Graph-Datenbanken ein (Bild 4). Alle nachfolgenden Produkte besitzen weniger als einstellige Prozentanteile. Dieser Erfolg begründet sich durch die zentralen Eigenschaften der Software: Bei Neo4j handelt es sich um ein extrem flexibles Datenbank-Management-System, das zwei verschiedene Betriebsarten anbietet: Zum einen kann man Neo4j als Teil einer Anwendung embedded (in-process) betreiben; die Anwendung besitzt einen direkten Zugriff auf die lokal auf der Festplatte befindlichen Datenbank. In diesem Fall können Programmierer direkt das Core-Java-

Click below to sign in and access your personal Neo4j

[Get Started with My Neo4j 7-Day Sandbox](#)

Username: neo4j

Password: flower-tongue-slope-place

Nach Anforderung einer Sandbox steht Neo4j sieben Tage zum Testen über das Web zur Verfügung (Bild 5)

API von Neo4j nutzen. Zum anderen kann Neo4j als Server-Datenbank betrieben werden, ähnlich, wie man es bei traditionellen relationalen Datenbanken gewohnt ist. Im Server-Betrieb verbinden sich verschiedene Anwendungen als Client mit der Neo4j-Datenbank und bearbeiten parallel die Daten.

Als zweiter wichtiger Vorteil unterstützt Neo4j, anders als viele NoSQL-Datenbanken, das aus der relationalen Welt bekannte ACID-Prinzip für Transaktionen. Dies macht Neo4j als Datenbank vollständig transaktionsfähig und (weitgehend) ACID-konform. Damit steht Neo4j den als robust geltenden relationalen Datenbanken in dieser Hinsicht in fast nichts nach. Auch ein Betrieb als OLTP-Anwendung (Online Transaction Processing) ist möglich. Im Unterschied zu relationalen Datenbanken leidet die Skalierbarkeit trotz Einhaltung des ACID-Prinzips bei der Graph-Datenbank Neo4j nicht. Mehrere Neo4j-Server kann man zu einem Cluster zusammenschließen. Fügt man einem solchen Neo4j-Cluster einen zusätzlichen Server hinzu, so kann dieser mehr Leseanfragen bearbeiten als eine einzelne Neo4j-Instanz für sich betrachtet.

Somit ist der Neo4j-Cluster, in seiner Einheit als Gesamtsystem gesehen, resistent gegenüber dem Ausfall einzelner Server und garantiert eine hohe Verfügbarkeit. Als Cluster-System erfüllt Neo4j die Anforderungen des Webs, bei denen neben der hohen Benutzeranzahl insbesondere der Leseaufwand einer Datenbank den Schreibaufwand oft um ein Viel-

faches übersteigt. Ferner garantiert Neo4j die referenzielle Integrität der Daten automatisch im Sinne der Entity-Relationship-Modellierung: Ein Knoten (Entity) kann nur gelöscht werden, wenn an ihm keinerlei Kanten mehr hängen. Eine Kante (Relationship) lässt sich nur erstellen, wenn diese sowohl einen Anfangs- als auch einen Endknoten besitzt. Schlussendlich verfügt Neo4j über die einfache, an SQL angelehnte Query Language Cypher (CQL). Als deskriptive Abfragesprache implementiert CQL zusätzlich ein mächtiges Pattern-Matching für die musterbasierte Suche in Graphen.

Sofort-Einstieg durch Nutzung einer Server-Datenbank

Für das erste Arbeiten mit einer Graph-Datenbank von Neo4j benötigt man nicht zwingend eine eigene Installation der Software. Vielmehr bietet Neo Technology über die Produkt-Homepage von Neo4j neben dem Download-Link auch das Einrichten einer sogenannten Sandbox an. Dabei handelt es sich um einen Neo4j-Server, auf den man über ein Webinterface im Browser zugreifen kann, um das Produkt für sieben Tage zu testen. Nach Klick auf die Sandbox-Schaltfläche der Neo4j-Homepage gibt man einige Daten zum Unternehmen, zur Person und zur E-Mail-Adresse ein und wählt *Create My Sandbox* aus. Anschließend erhält man in einer E-Mail einen Zugangslink zum Datenbank-Server mit den notwendigen Anmeldedaten (Username, Passwort) (Bild 5).

Für den maximal sieben Tage andauernden Test stehen alle wichtigen Funktionen des Datenbank-Management-Systems zur Verfügung. Die Server-Datenbank der Sandbox enthält bereits den Graphen einer Film-Datenbank (Movie Graph) mit Daten über Schauspieler, Filme und ein Beispiel für die Umstellung einer relationalen Datenbank, Northwind Graph genannt, auf eine Graph-Datenbank. Klickt man im ►

Entity-Relationship-Modell für den Datenbankentwurf

Das Entity-Relationship-Modell hat sich seit vielen Jahren als De-facto-Standard für die Modellierung einer Datenbank bewährt.

Als Modell veranschaulicht es in einer Grafik (ER-Diagramm genannt) alle verwendeten Elemente und beschreibt diese genauer, sowohl was Struktur als auch Bedeutung (Semantik) betrifft. In einem Entity-Relationship-Modell kommen drei Grundelemente vor:

- **Entity (Entität):** Stellt ein individuell identifizierbares Objekt des zu modellierenden Umweltausschnitts dar.
- **Relationship (Beziehung):** Entspricht einer Verknüpfung, Beziehung oder einem Zusammenhang zwischen zwei oder mehreren Entitäten.
- **Property (Attribut):** Ist eine Eigenschaft, die im Kontext zu einer Entität gehört.

Das Entity-Relationship-Modell entsteht als Ergebnis der Typisierung von Objekten/Entitäten, ihrer Beziehungen untereinander und der zugehörigen Attribute.

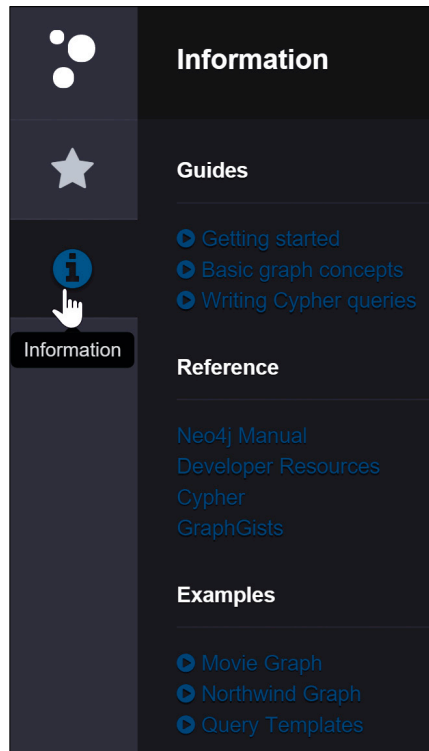
Webbrowser nach Anmeldung in der Sandbox in der linken Sidebar auf das kleine i-Symbol (Bild 6), so öffnet sich nach rechts ein Examples-Bereich mit weiteren Auswahloptionen. Wählt man eine der beiden Einträge aus, zum Beispiel *Northwind Graph*, so erscheint im Webbrowser rechts ein Fensterbereich mit weiteren Erläuterungen. Mittels des dort ganz rechts stehenden kleinen Pfeil-Symbols zeigt man sich schrittweise weitere Erklärungen und Arbeitsschritte an. Damit macht man sich näher mit dem Produkt und der Arbeitsweise von Neo4j bekannt.

Wählt man im Webbrowser der Sandbox einen angezeigten Cypher-Befehl durch Anklicken aus, so übernimmt das Webinterface die zugehörigen Code-Zeilen in den oberen Bereich. Positioniert man den Mauszeiger auf das kleine, nach rechts gerichtete Abspiel-Symbol (Bild 7), zeigt dieses den Hover-Help-Text *Play* als Tooltip an. Ein Klick darauf, und der Neo4j-Server führt die angezeigten Cypher-Befehle aus. Danach erscheint darunter ein neuer Ausgabebereich mit Informationen über die Befehlsausführung.

Über das im ganz linken Bereich stehende kleine Stern-Symbol mit *Favorites* als Hover-Help-Text (Tooltip) erhält man Einblick in aktuelle Datenbestände oder Systeminformationen der aktuellen Datenbank. Hierzu wählt man aus den neu angezeigten Ausgabebereichen einen der Texteinträge aus, beispielsweise *Get some data*. Danach erscheint im rechts oben stehenden Bereich des Webbrowsers der zugeordnete Cypher-Befehl. Ein Klick auf die Schaltfläche *Play* bringt diesen zur Ausführung. Anschließend zeigt der Webbrowser darunter die Daten in Form eines Graphen visualisiert an (Bild 8).

Marktorientierte Produkt-Editionen mit flexiblen Lizenzmodellen

Klickt man auf der Homepage von Neo4j die Schaltfläche *Download* an, so öffnet sich eine neue Webseite zum Herunterladen der Software. Dort werden im Unterschied zu früher aktuell zwei verschiedene Produkt-Editionen von Neo4j zur Auswahl angeboten: Enterprise und Community. Während die



Die rechte Sidebar der Sandbox enthält mehrere Auswahlbereiche (Bild 6)

kostenlose Community Edition alle wichtigen Funktionen des Datenbank-Management-Systems enthält, ergänzt die Enterprise Edition diese um wichtige, für den professionellen Betrieb der Datenbank im Unternehmen hilfreiche Features. Zu diesen gehören: Enterprise Lock Manager, Cache Sharding, Clustered Replication, Cypher Query Tracing, Property Existence Constraints, Hot Backup und Advanced Monitoring.

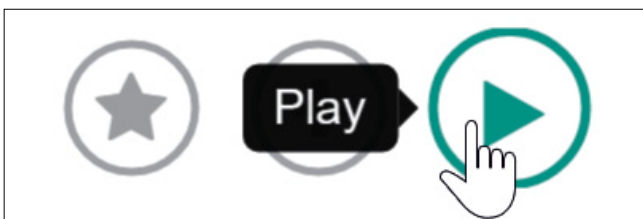
Die kostenpflichtige, über ein Subscription/Abo-Modell erhältliche Enterprise Edition vertreibt Neo Technology in zwei Lizenzmodellen: Enterprise Subscription und Startup Program. Hat man sich für das Herunterladen der Enterprise Edition entschieden und einige Daten zum Ansprechpartner mit E-Mail-Adresse eingegeben, erhält man per E-Mail einen Download-Link. Anschließend lässt sich die Software für 30 Tage kostenlos testen und in vollem Umfang nutzen. Nach Herunterladen und erfolgter Installation nimmt ein Mitarbeiter von Neo Technology Kontakt mit dem genannten Ansprechpartner auf. Dieser hilft bei techni-

schen Fragen und nimmt eine Einstufung des Anwenderunternehmens für das am besten geeignete Lizenzmodell vor.

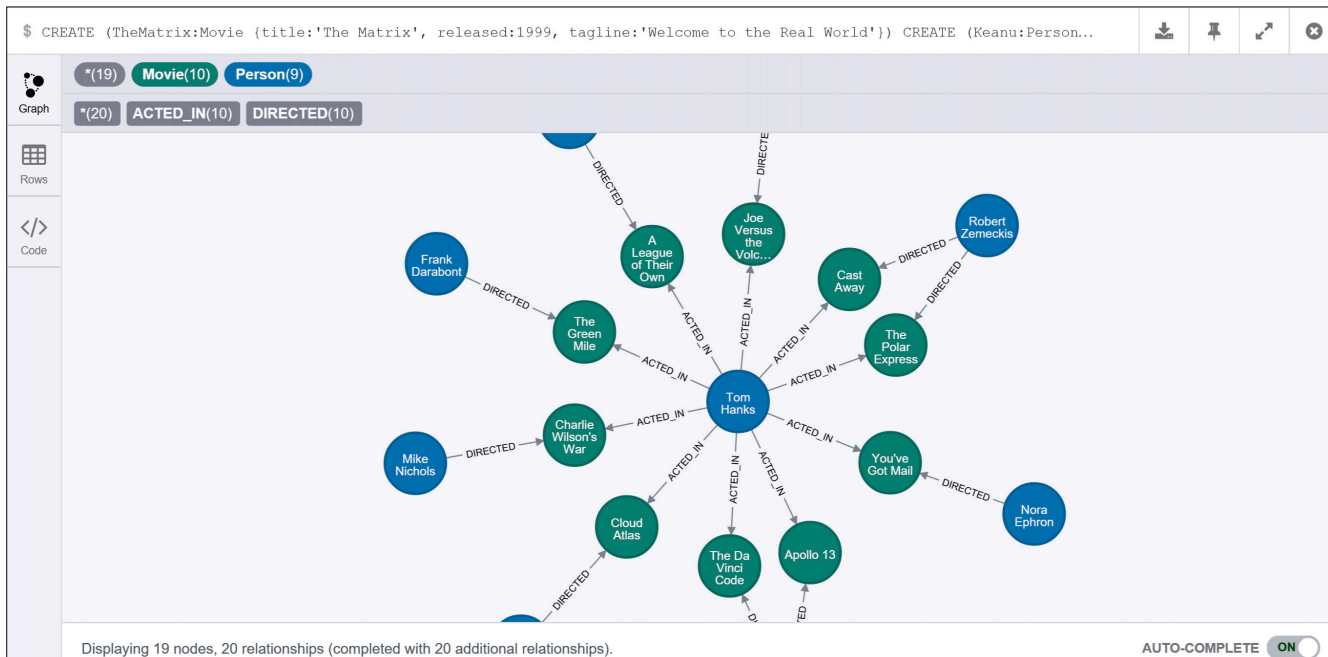
Neben der normalen Enterprise Subscription teilt Neo Technology die Unternehmen des Startup-Programms nochmals in zwei Gruppen ein. Kleinere Unternehmen mit bis zu 20 Mitarbeiter können die Enterprise Edition kostenlos nutzen. Dabei kann das betroffene Unternehmen gegebenenfalls erforderliche Supportleistungen separat erwerben. Für größere Firmen mit bis zu 50 Mitarbeitern steht ein spezielles Startup-Preismodell der Enterprise Edition zur Verfügung. Diese Softwarelizenz umfasst bereits Supportleistungen eines Neo4-Experten, um für das Unternehmen einen optimalen Einsatz von Neo4j sicherzustellen. Ergänzend dazu bietet Neo Technology für Softwarehäuser, die ein eigenes mit Neo4j entwickeltes Produkt vertreiben, ein Neo4j-OEM-Partnerprogramm für Dritthersteller an.

Installation und Einrichten des Datenbank-Management-Systems

Ein Klick auf die Schaltfläche *Download Community Edition* öffnet eine neue Website *Installing Neo4j* und lädt automatisch die für das aktuelle Betriebssystem erforderliche Neo4j-Version herunter. Der Link *Other Releases* auf der Download-Homepage macht die passenden Neo4j-Versionen für alle anderen Betriebssysteme zugänglich. Zusätzlich findet man dort auch den aktuellen Milestone der in Entwicklung befindlichen Beta-Version sowie noch verfügbare ältere Neo4j-Versionen. Hinweise zur Installation der Graph-Datenbank erhält man über die durch den Download-Prozess geöffnete Website *Installing Neo4j*. Dazu wählt man einen der oben ste-



Ein Klick auf das Play-Symbol im Webinterface der Sandbox oder des Neo4j-Browsers führt Cypher-Befehle aus (Bild 7)



Der Neo4j-Browser zeigt Graphen in visueller Form an (Bild 8)

henden Informationsbereiche *Mac OS X (dmg)*, *Linux/Unix (tar)*, *Windows (exe)* oder *Windows (zip)* aus.

Klickt man auf den unten stehenden Link *Neo4j Manual: Server Installation Instructions* (Bild 9), so öffnet sich die zugehörige Seite im Neo4j-Online-Manual. Zusätzlich hat Neo Technology für die komplette Dokumentation der Software eine spezielle Homepage eingerichtet (<http://neo4j.com/docs>). Darüber erreicht man sowohl die HTML- als auch die PDF-Ausgaben für alle aktuell verfügbaren und unterstützten Neo4j-Versionen. Auch die Manuals verschiedener Beta-Versionen (Milestones) von Neo4j sind hierüber zugänglich.

Ergänzend zur Systemdokumentation bietet diese Website die zu einer bestimmten Neo4j-Version passende Referenzcard der Cypher Query Language zum Herunterladen an. Ganz unten auf der Homepage der Dokumentation befindet sich der Bereich *Other documentation*. Dort erreicht man die Javadocs des API (Entwicklung von Plug-ins und Erweiterungen) sowie den GraphGists-Pool.

Die Entwicklung von Neo4j basiert auf der Programmiersprache Java. Daher empfiehlt Neo Technology für das Arbeiten mit der Graph-Datenbank eine JDK-Installation. Sinnvollerweise sollte man eine JDK-8-Version mit dem aktuell-

GraphGists – ein Datenbank-Pool für Anwender

GraphGists stellt einen Pool von Graph-Datenbanken verschiedener Anwender dar.

Eine GraphGist dient dem Austausch von Dokumenten (Graph-Datenbank inklusive Cypher-Abfragen) mit anderen Benutzern.

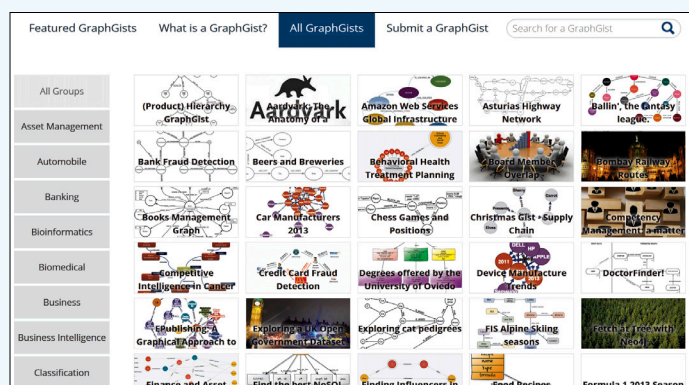
Über die Homepage der Neo4j-Systemdokumentation erreicht man den GraphGists-Pool; ganz unten auf dieser Webseite steht der dazugehörige URL-Verweis. Auf der GraphGists-Site befinden sich die folgenden Bereiche:

Featured GraphGist: Zeigt einige ausgewählte Graph-Datenbanken mit Verweisen zum näheren Erkunden an.

What is a GraphGist: Erklärt die Modellierung und Dokumentation einer Graph-Datenbank anhand einer einfachen Textdatei.

All GraphGist: Listet den aktuellen Pool an GraphGists nach Branchen gruppiert auf.

Submit a GraphGist: Um eine GraphGist in den Pool aufzunehmen,



GraphGists stellt einen Pool von Neo4j-Graph-Datenbanken über das Internet für den Austausch zwischen Anwendern bereit

men, benötigt man einen Account von Twitter oder GitHub. Diese Webseite zeigt eine Preview des Graphen an oder überträgt mittels eines URL die GraphGist in den Pool.

ten Update verwenden, um eventuelle bereits vom Hersteller in Neo4j behobene Fehler beim Betrieb des Datenbank-Management-Systems von vornherein auszuschließen. Den im Hintergrund laufenden Serverteil der Datenbank startet man über das Programm Neo4j Community Edition durch Klick auf die Start-Schaltfläche (Bild 10).

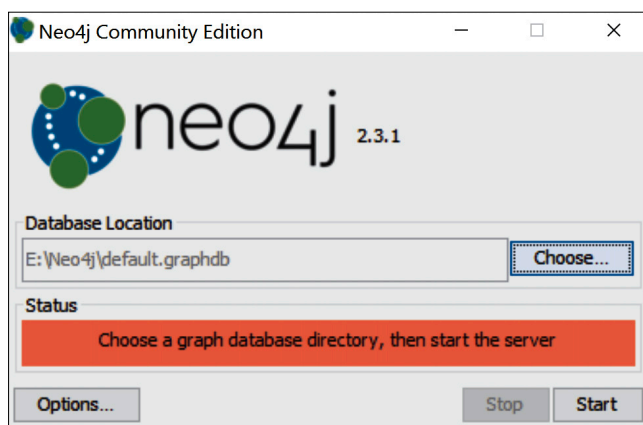
Nach Installation von Neo4j erreicht man das Webinterface der Neo4j-Console, den sogenannten Neo4j-Browser, über den URL <http://localhost:7474>. Dort meldet man sich mit dem Usernamen *neo4j* und dem gleichlautenden Passwort über die Schaltfläche *Connect* an. Nach erstmaligem Verbindungsaufbau zur Datenbank verlangt diese eine Änderung des Kennworts des standardmäßig eingerichteten Benutzers. Das Kennwort eines aktuell angemeldeten Benutzers ändert im Nachhinein auch der Cypher-Befehl `:server change-password`.

Über den Cypher-Befehl `:play` start stehen im Webbrowser drei Informationsbereiche *Learn about Neo4j*, *Jump into code* und *Monitor the system* zur Verfügung. Diese Bereiche bieten nach entsprechender Auswahl Grundlagen von Graph-Datenbanken, der Cypher Query Language oder der Systeminformationen an. Für die ersten beiden Bereiche öffnen sich dort über das kleine, nach rechts zeigende Pfeil-Symbol zusätzliche Scroll-Bereiche, welche die jeweiligen Grundlagen weiter detaillieren.

Im Hauptfenster des Neo4j-Community-Edition-Programms (Bild 10) kann, solange noch kein Datenbank-Server gestartet ist, über die Schaltfläche *Choose ...* im Ausgabebereich *Database Location* die zu startende Neo4j-Datenbank festgelegt werden. Eine dortige Auswahl speichert sich nur dann persistent, wenn man das Neo4j-Community-Edition-Programm als Administrator ausführt.

Klickt man im Hauptfenster des Neo4j-Community-Edition-Programms auf die Schaltfläche *Options ...*, so erscheint das zugehörige Options-Fenster mit fünf Bereichen und gleichnamigen Schaltflächen (Bild 11):

- **Command Prompt ...:** Öffnet eine Eingabeaufforderung/Terminalfenster mit einer gesetzten Umgebungsvariable/Pfad zur aktuellen Neo4j-Installation. Darüber stehen die beiden Werkzeuge Neo4jShell oder Neo4jImport zur Verfügung.



Das Neo4j-Community-Edition-Programm dient zum Auswählen der Datenbank, zum Starten und Stoppen des Datenbank-Servers sowie zur Konfiguration (Bild 10)



Bereits beim Herunterladen von Neo4j steht die Systemdokumentation über das Web zur Verfügung (Bild 9)

- **Database Tuning:** Startet einen Texteditor/Notepad zur Bearbeitung der Datei *neo4j.properties*. Diese Neo4j-Systemdatei enthält alle Einstellungen zur Verbesserung der Performance einer Neo4j-Datenbank.
- **Server Configuration:** Bietet über einen Texteditor/Notepad das Ändern der Server-Konfigurationsdatei *neo4j-server.properties* an.
- **Java VM Tuning:** Übergibt beim Start des Neo4j-Datenbank-Management-Systems passende, in der Datei *neo4j-community.vmoptions* stehende Optionen als Parameter an die Java VM.
- **Plugins and Extensions:** Zeigt den Ablageordner für Server-Plug-ins und andere Erweiterungen von Neo4j an.

Neben einer nahezu automatischen Installation lässt sich eine Neo4j-Installation für die verschiedenen Betriebssysteme (Windows, Mac OS X, Linux/Unix) auch schrittweise und an die eigenen Bedürfnisse angepasst durchführen. Für Windows steht dazu ein 32- und 64-Bit ZIP-Archiv, für Mac OS X und Linux/Unix ein tar-Archiv zur Verfügung. Für das Linux-Derivat Debian hat Neo4 Technology ein Debian-Repository eingerichtet; dieses lässt sich für alle auf Debian basierenden Linux-Derivate wie Ubuntu nutzen. Mittels eines Debian-kompatiblen Package-Managers installiert man über den Parameter *neo4j=versionsnummer* ganz gezielt eine bestimmte Neo4j-Version. Ältere, nicht im aktuellen Debian-Repository enthaltene sowie Beta-Versionen der Software findet man in den beiden Debian-Repositories *oldstable* und *testing*.

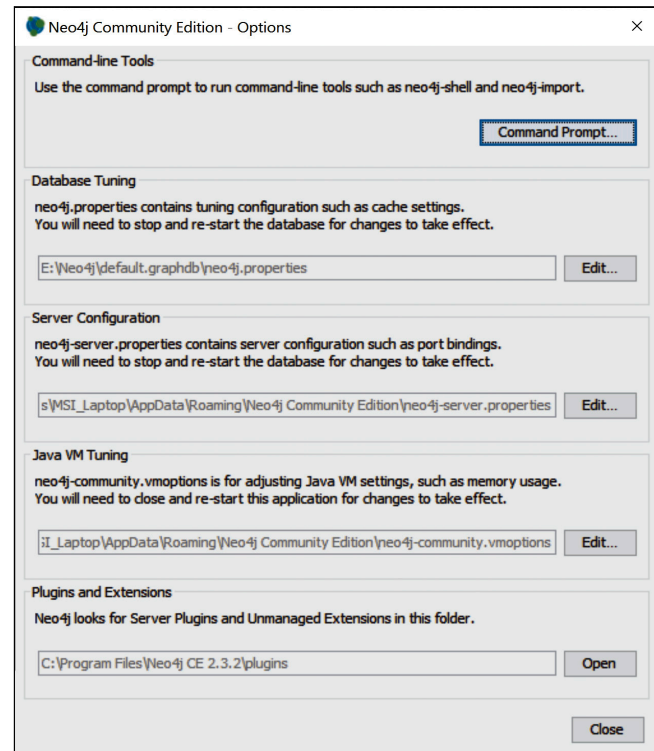
Individuell angepasste Installation für verschiedene Betriebssysteme

Nach dem Entpacken der Archiv-Datei definiert man für das jeweilige Betriebssystem eine Umgebungsvariable *NEO4J_HOME*, die passend auf das Installationsverzeichnis von Neo4j zeigt. Zudem muss man darauf achten, die notwendigen Berechtigungen für die Benutzer einzurichten. Für administrative Arbeiten mit dem Datenbank-Management-System wie Starten und Stoppen der Server-Datenbank benötigt man eine Berechtigung zur Ausführung der im *bin*-Installationsverzeichnis vorhandenen Programme. Zusätzlich müssen alle Benutzer, die mit den Daten der Graph-Datenbank arbeiten sollen, eine Schreibberechtigung für sie besitzen. Für die Rolle eines Neo4j-Administrators empfiehlt es sich, einen eigenen Benutzer einzurichten; diesem macht man dann das *neo4j*-Kommando zur Ausführung verfügbar. Der *neo4j*-Befehl kennt nachfolgende Übergabeparameter:

- **console:** Startet im Vordergrund einen Server-Prozess, den man mittels [Strg C]/[Ctrl C] abbrechen/beenden kann.
- **info:** Gibt Informationen über die Systemkonfiguration aus.
- **restart:** Führt einen Neustart des Servers als Hintergrundprozess durch.
- **start:** Startet einen Hintergrundprozess für den Server.
- **status:** Zeigt den aktuellen Status des Servers an.
- **stop:** Beendet einen im Hintergrund laufenden Server-Prozess.

Unter Windows greift Neo4 Technology für individuell angepasste Installationen auf die Windows PowerShell (ab Version 2.0) zurück. Damit steht die neue Windows-Schnittstelle zur Administration und Fernwartung von Systemsoftware auch für Neo4j zur Verfügung. Um Befehle in der PowerShell ausführen zu können, muss die Ausführung von Skripten auf dem Windows-System aktiviert werden. Hierzu öffnet man, zum Beispiel über das Datei-Menü im Windows Explorer, ein Eingabefenster der PowerShell als Administrator. Im Fenster *Administrator: Windows PowerShell* aktiviert man über den Befehl `Set-ExecutionPolicy Unrestricted` und Bestätigung durch den Buchstaben *A* die Berechtigung zur Ausführung von Skripten.

Anschließend importiert man mittels des Befehls `Import-Module C:\Program Files\neo4j-community-2.3.2\bin\Neo4j-Management.psd1` (man beachte die Anführungszeichen



Das Konfigurationsfenster erreicht man über die Options-Schaltfläche des Neo4j-Community-Edition-Programms (Bild 11)

bei "Program Files" im Verzeichnisnamen) die Neo4j-Module. Mit diesen Neo4j-Management-Modulen kann man einfache Aufgaben durchführen, etwa die Einstellungen einer Neo4-Datenbank einsehen oder Neo4j-Windows-Services administrieren. Allerdings sind die Neo4j-Module nur während einer laufenden PowerShell-Sitzung bekannt; bei Start einer neuen müssen die Module jeweils wieder importiert werden. Zudem muss die Umgebungsvariable `java_home` zur Ausführung auf das Installationsverzeichnis des JDK verweisen.

Jetzt lassen sich alle Aufgaben der Systemadministration für Neo4j über die Windows PowerShell ausführen. Hierzu muss man die PowerShell auch als Administrator unter Windows aufrufen. Beispielsweise startet der Befehl `'C:\Program Files\neo4j-community-2.3.2\' | Start-Neo4jServer -Console` ein Konsolenfenster für einen Neo4j-Server. Zugehörige Systemmeldungen seitens der Neo4j-Software erscheinen in einem separaten Ausgabefenster. Der Befehl `Get-Command -Module Neo4j-Management` gibt die in der Windows PowerShell verfügbaren Funktionen des Neo4j-Moduls aus. Der Befehl `Start-Neo4jShell` öffnet eine Neo4j-Shell, um direkt mit der Datenbank zu kommunizieren. Die Mächtigkeit der PowerShell ermöglicht aber auch die Realisierung komplexer Aufgaben wie den Import von Server-Einstellungen, das Einrichten von Neo4j-Clusternodes und Ähnliches.

Visualisierung der Graph-Datenbank per Browser-App

Neo Technology liefert Neo4j mit einer Web-App (Browser-Schnittstelle, manchmal auch Neo4j-Browser genannt) aus; diese besitzt seit Version 2.2.0 ein verbessertes User-Inter- ►

Grundbegriffe zum Cloud Computing

Als Projektleiter und Entwickler sollte man nachfolgende Grundbegriffe aus dem Cloud Computing kennen:

Drei-Schichten-Architektur: Drei Schichten von der obersten Ebene der Anwendung (Software as a Service/SaaS), zur mittleren Schicht der Plattform (Platform as a Service/PaaS) bis zur tiefsten Ebene der Infrastruktur (Infrastructure as a Service/IaaS) bilden die Grundbausteine einer Cloud-Computing-Architektur. Die oberste Ebene richtet sich an die Endbenutzer, die mittlere Schicht an Entwickler und Programmierer. Die tiefste Schicht stellt die Administrations- und Betriebsebene dar.

Cloud-Typen: Nach Betriebsform und Bereitstellung (Deployment) unterscheidet man verschiedene Cloud-Typen: Private (Dienste über Webtechnologien nur intern, für Mitarbeiter des Unternehmens zugänglich), Public (Dienste über das Web für jedermann, offen zugänglich) und Hybrid Cloud (abgestimmte Zusammenarbeit zwischen internen Abteilungen beziehungsweise externen Anbietern oder Partnern).

Rollen und Zuständigkeiten: Für reibungsloses, wirtschaftliches und leistungsfähiges Zusammenarbeiten definiert man ein Geschäfts- und Service-Modell, das Rollen mit Aufgaben spezifiziert. Bekannte, akzeptierte Rollen findet man in NIST Cloud Computing Reference Architecture oder Cloud Computing Reference Architecture (CCRA) von IBM.

face und eine höhere Performance. Die per Browser zugängliche App entspricht einer Neo4j-Console. Sie eignet sich also sowohl für Datenbank-Administratoren, Endbenutzern als auch Programmierer. Standardmäßig erreicht man die Browser-Console über den Port 7474 von localhost (<http://127.0.0.1>). Über die Datei `neo4j-server.properties` des `conf`-Unterverzeichnisses der Neo4j-Installation stehen nachfolgende Einstellungen für dieses Webinterface zur Verfügung:

- **org.neo4j.server.webserver.port=7474:** Diese Zeile erlaubt einen Zugriff über eine andere Port-Adresse.
- **org.neo4j.server.webserver.address=0.0.0.0:** Darüber macht man eine Neo4j-Datenbank über das Webinterface von allen externen Hosts zugänglich.
- **dbms.browser.remote_content_hostname_whitelist:** Dabei handelt es sich um die Whitelist der Hosts, die über das Webinterface auf eine Datenbank zugreifen können. Ein nachfolgendes `"*"` (nur in der Enterprise-Edition verfügbar) erlaubt einen Zugriff für alle Hosts.
- **dbms.security.allow_outgoing_browser_connections:** Setzt man diesen Zeileneintrag auf `true`, verhindert dies die Ausführung externer Services und das Laden externer Daten.
- **org.neo4j.server.manage_console_engines:** Die Zuweisung einer leeren Zeichenkette (`=`) schaltet über das Webinterface die Neo4j-Shell aus.

Für den Einstieg in das Arbeiten mit dem Neo4j-Browser empfiehlt es sich, in der Sidebar (dem ganz links stehenden Bereich) das kleine *i*-Symbol mit dem Tooltip *Information* anzuklicken. Dies öffnet nach rechts einen in die nachfolgenden vier Abschnitte gegliederten Informationsbereich:

- **Guides:** Bietet verschiedene Leitfäden für den Einstieg in Neo4j, Graph-Datenbanken und Cypher an.
- **References:** Ermöglicht Zugriffe auf die Handbücher oder Webseiten von Neo4j, Developer Resources, Referenzkarte zu den Cypher-Befehlen und GraphGists.
- **Examples:** Umfasst Beispiele zu den Graph-Datenbanken Movie Graph, Northwind Graph und Abfragebeispiele als vorgefertigte Query Templates.
- **Help:** Enthält ein rudimentäres Hilfesystem zum Neo4j-Browser, zur Cypher-Syntax, speziell für Neo4j-Browser realisierte Client-Kommandos und eine Liste verfügbarer Tastaturkürzel.

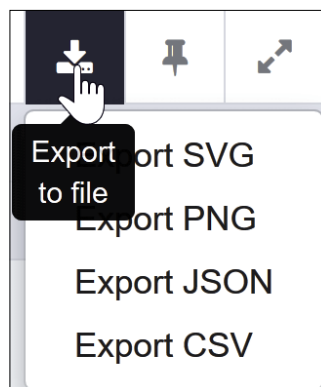
Ganz oben im Neo4j-Browser direkt neben dem `$`-Zeichen befindet sich seine Kommandozeile mit Pufferspeicher für die Befehlshistorie. Dieser funktioniert wie bei Terminal-Fensystem des Betriebssystems: Über die Cursor-Tasten (unten, oben) scrollt man durch bereits eingegebene Befehle und führt bereits ausgeführte Kommandos erneut aus. Gibt man in die Kommandozeile den Befehl `:play intro` ein, so

öffnet sich darunter eine geführte Benutzeranleitung durch den Neo4-Browser (*Introduction – Getting started...*).

Ein Klick auf den kleinen, nach rechts zeigenden Pfeil scrollt durch verschiedene weitere Informationsseiten, die den Aufbau und die Handhabung des Neo4-Browsers erläutern. Ein speziell für das Arbeiten mit dem Neo4j-Browser in Projekten geeignetes Tutorial findet man Web. Das Tutorial eignet sich auch für den Einstieg in die Neo4j-Datenbank, da es schrittweise aufgebaut ist und keinerlei Kenntnisse zu Neo4 voraussetzt. Cypher-Befehle, die als Ergebnis Knoten oder Beziehungen enthalten, visualisiert der Neo4j-Browser sofort als Graphen. In diesem Fall erscheint direkt links unter der Kommandozeile eine Sidebar, mit der man von der Graph-Ansicht in eine

Tabellen-Ansicht umschalten kann.

Rechts neben der Kommandozeile befindet sich ein Download-Icon mit dem Tooltip *Export to file*. Klickt man dieses Symbol an, so öffnet sich ein Kontextmenü, um das Dateiformat festlegen zu können (Bild 12). Die ersten beiden Menüeinträge (SVG, PNG) speichern eine Bilddatei des Graphen; bei Auswahl von CSV erhält man eine Comma-Separated-Values-Datei (CSV) beziehungsweise über JSON eine Textdatei im JavaScript-Object-Notation-Format (JSON). Die CSV-Datei kann man direkt für den Import in eine Tabellenverarbeitung und die JSON-Datei als Schnittstelle für die Weiterverarbeitung in eigenen Programmen verwenden.



Ergebnisse von Cypher-Queries kann man in verschiedenen Formaten speichern (Bild 12)

Einsatz von Neo4j in einer Cloud-Umgebung

Neben einem Einstieg in das Arbeiten mit Neo4j über die Remote-Sandbox oder eine lokale Installation lässt sich die Software auch über eine Cloud-Umgebung nutzen. Dazu sollte man allerdings bereits erste Erfahrungen im Zusammenhang mit Einrichtung und Betrieb einer Graph-Datenbank vor allem als Remote-System besitzen. In der Regel eignet sich Neo4j für jede gängige Cloud-Computing-Plattform wie AWS, Pivotal Cloud Foundry, Jelastic, OpenShift oder Windows Azure. Jedoch arbeitet Neo Technology mit einigen Partnern im Rahmen eines besonders strukturierten Programms (Bild 13) enger zusammen. Diese Partner erhalten seitens des Herstellers besondere Unterstützung für die Implementierung von Neo4j in Cloud-Umgebungen.

Zu ihnen gehören auch Partner, die Neo4j-Instanzen direkt über ihre Cloud-Computing-Plattform hosten. In diesem Fall handelt es sich um eine Cloud-Datenbank, die einen Plat-

I'm a cloud provider interested in offering Neo4j as a database-as-a-service (DBaaS). How can I get started?

Aw, shucks! We think that's a great idea. We have a structured program for this sort of arrangement, so your customers will know you are backed by Neo4j and Neo Technology, Inc.

Ein Partner-Programm zwischen Neo Technology und Cloud-Providern stellt qualitativ hochwertige Schnittstellen sicher (Bild 13)

form-as-a-Service-Dienst (PaaS), genauer gesagt: Database as a Service (DBaaS), für Entwickler und Programmierer bereitstellt. Bei der Zusammenarbeit mit einem Neo4j-Partner, aus Sicht des Anwenders Provider genannt, sollte man in einem Dienstleistungsvertrag, dem Service Level Agreement (SLA), die zu erbringende Dienstleistungen abstimmen.

In der Praxis spezifiziert ein solcher Dienstleistungsvertrag die Schnittstelle zwischen Auftraggeber und Provider. Insofern stellt der SLA-Vertrag die Grundlage für die Aufgabenteilung zwischen Provider und Auftraggeber dar.

Für den Auftraggeber entfällt in der Regel das Konfigurieren, Einrichten und Bereitstellen der für die jeweilige Cloud-Umgebung erforderlichen spezifischen Systemobjekte. Zudem empfiehlt es sich, dass der Provider Aufgaben der Installation und Administration des Neo4j-Datenbank-Management-Systems übernimmt.

So kann spezifiziert sein, dass der Service-Provider alle verbundenen Aufgaben inklusive der Wartung der Neo4j-Systemsoftware übernimmt. Auch kann man vereinbaren, dass alle Aufgaben über Einrichtung, Betrieb, Pflege und Administration der Graph-Datenbank selbst, inklusive ihrer Datenbestände, in der Hoheit des Anwenderunternehmens verbleiben.

Die nachfolgenden Hosting-Provider Elastx AB (Stockholm, Schweden), GrapheneDB Labs (Las Palmas, Gran Canaria) und Graph Story (Memphis, Tennessee, USA) bieten einen DBaaS-Dienst speziell für Neo4j an. Während Elastx erst seit Kurzem Neo4j auf Basis von Jelastic und OpenStack Neo4j als DBaaS unterstützt (Bild 14), sind die beiden anderen Firmen schon länger tätig.

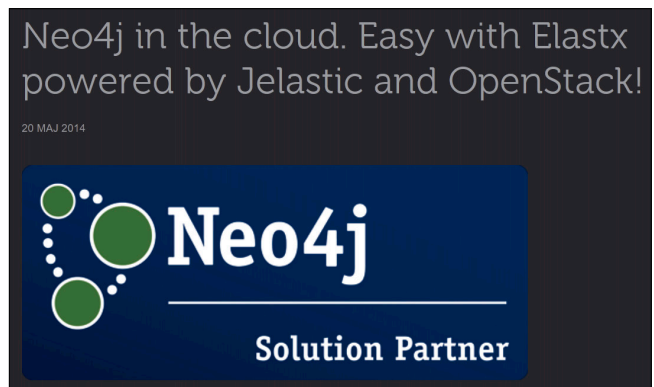
GrapheneDB ist für die Cloud-Plattformen AWS, Azure und Heroku verfügbar, während Graph Story AWS, Azure, Digital Ocean und Softlayer unterstützt. Bei allen drei Hosting-Providern kann man mit der Community Edition starten und im Bedarfsfall auch die Enterprise Edition einsetzen. Über die Homepage der Provider erfolgt recht schnell ein Einstieg zum

Windows PowerShell für individuelle Administration

Die PowerShell unterstützt auf der Betriebssystemebene von Windows die aus der Unix-Welt bekannten Techniken von Pipes und Filtern inklusive der objektorientierten Programmierung.

Microsoft entwickelte die Windows PowerShell als Alternative zu dem in die Jahre gekommenen Windows-Kommandozeilenprogramm cmd.exe. Die PowerShell versteht alle Befehle der DOS-Kommandozeile, Windows Script Host (WSH) und die eigene PowerShell Scripting Language.

Die Windows PowerShell arbeitet objektorientiert und liefert als Ausführungsergebnis immer ein Objekt zurück. Mittels der Skriptsprache von Windows PowerShell kann man das Windows Management Instrumentation (WMI) nutzen. WMI stellt unter Windows eine der wichtigsten Schnittstellen für Administration und Fernwartung von Arbeitsplatzrechnern und Servern dar.



Der schwedische PaaS-Provider Elastx bietet als Neo Technology Solution Partner einen DBaaS-Dienst für Neo4j an (Bild 14)

Einsatz von Neo4j als DBaaS. Dort findet man neben Schritt-für-Schritt-Anleitungen auch einen speziellen Fragenkatalog (FAQ/Frequently Asked Questions) mit Hintergrundinformationen.

Neo4j in der Cloud mit Graph Story

Für den Einstieg in den DBaaS-Dienst mit Neo4j bietet Graph Story über die Homepage mehrere Möglichkeiten zum Einrichten eines 14 Tage gültigen Test-Accounts an:

- **Request A Demo:** Erzeugt automatisch eine E-Mail an demo@graphstory.com mit dem Betreff: *Request a Demo – Graph Story*. Dies setzt allerdings eine Installation eines E-Mail-Client-Programms voraus.
- **Start Your Free Trial/Trial:** Beide öffnen ein Fenster, um sich per E-Mail zu registrieren oder per GitHub-Login direkt anzumelden. Nach Eingabe der E-Mail-Adresse erhält man einen Link, um den neuen Account zu aktivieren und das Kennwort festzulegen.
- **Create Your First Graph Database:** Im unteren Bereich der Homepage befindet sich dieser Link mit demselben Verhalten wie *Request A Demo* (solange man sich nicht angemeldet hat). Nach Anlage eines Accounts erhält man hierüber Vorbelegungen zum Erzeugen der ersten Graph-Datenbank.
- **Chat-Room:** Ganz rechts unten auf der Homepage gibt es ein Symbol in Form zweier Sprechblasen; über dieses Icon gelangt man in einen individuellen Chat-Room, um Fragen zu stellen.

Besitzt man bereits erste Erfahrungen im Hinblick auf das Arbeiten mit einem DBaaS-Dienst und hat man sich für einen tiefergehenden Test von Graph Story entschieden, so empfiehlt es sich, direkt über den dritten Bereich der *Homepage Prices* zu gehen. Dort unterscheidet Graph Story generell zwei Produkte: Standard und Enterprise.

Während Standard für etwa 10 Dollar bestimmte Services als Dienstleistungen umfasst, stimmt im Enterprise-Fall das Anwenderunternehmen aufgrund seiner Anforderungen die erforderlichen Dienstleistungen zusammen mit Graph Story über ein SLA genau ab. Auf Basis der Anforderungen erhält man von Graph Story ein Angebot, das als Grundlage für den individuellen Dienstleistungsvertrag dient. ►

Beim ersten Produkt Graph Story Standard legt man zusätzlich noch einen sogenannten Plan fest. Er bezieht sich auf eine genaue Spezifikation der zugrunde liegenden Basisdienste und wird schrittweise durch zwei Auswahlen festgelegt (Cloud-Computing-Provider, Speicherort mit Speichergröße). Für die Basisdienste verlangen Cloud-Computing-Provider wie Digital Ocean oder AWS abhängig vom Speicherort und der Speichergröße weitere monatliche Gebühren, die zusätzlich zur Grundgebühr für das Graph-Story-Produkt zu entrichten sind. In der Regel zieht Graph Story anhand einzugebender Kreditkartendaten die monatlich zu zahlenden Gebühren selbst ein. Diese Vorgehensweise stellt sicher, dass man selbst keinen Zusatzvertrag mit einem weiteren Cloud-Computing-Provider abschließt, sondern nur einen Ansprech- und Support-Partner hat: den DBaaS-Provider Graph Story.

Einrichten einer Graph-Datenbank

Meldet man sich nach Einrichtung eines Benutzerkontos in der Graph-Story-Cloud an, so erscheint ganz links ein Navigationsbereich mit verschiedenen Symbolen (Bild 15). Über das dritte Symbol von oben (ein kleiner Graph mit der Bezeichnung Graph List aus drei Knoten und zwei Kanten) beginnt man mit dem Anlegen einer Graph-Datenbank in Neo4j. Nachdem man einen *Graph Name* vergeben hat, wählt man das *Data Center* (Speicherort) aus (Bild 16), danach erzeugt ein Klick auf *+Create* die Graph-Datenbank. Anschließend kann man über den Bereich *Import Sample Data Sets* durch Klick auf *Sample Data Sets* und weitere Auswahl einer der beiden *Import data*-Schaltflächen Beispieldaten für den Graphen übernehmen. Nach Erfolgsmeldung hat man in

Create a New Graph (Develop)

Graph Name

Data Center

Sämtliche Aufgaben von Administration, Betrieb und Bereitstellung/Deployment eines Neo4j-Systems erfolgt über die Graph-Story-Cloud (Bild 16)

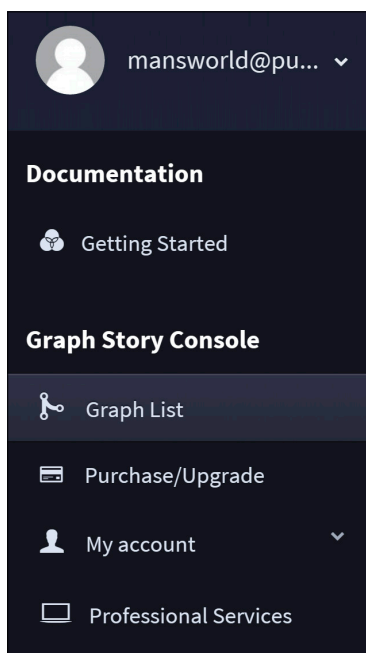
Neo4j eine Graph-Datenbank erzeugt. Einen Import von Daten in eine neu angelegte Graph-Datenbank setzt vorher eine erfolgreiche Bestätigung der angegebenen E-Mail-Adresse und die Vergabe eines Kundenkennworts voraus.

Graph Story unterstützt bereits Testkunden über den erwähnten Chat-Room vorbildlich. Stellt man Fragen, werden diese, sobald ein Mitarbeiter des Kundendiensts online ist, sofort beantwortet. Es ertönt ein angenehmes leises Signal, das auf eine Chat-Nachricht mit Angebot auf Unterstützung durch den Support hinweist. Im Bereich *Documentation* findet man über den Punkt *Getting Started* (das zweite Symbol) und Auswahl der gewünschten Sprache Anleitungen zur Programmierung mit Beispielen und Quellcode.

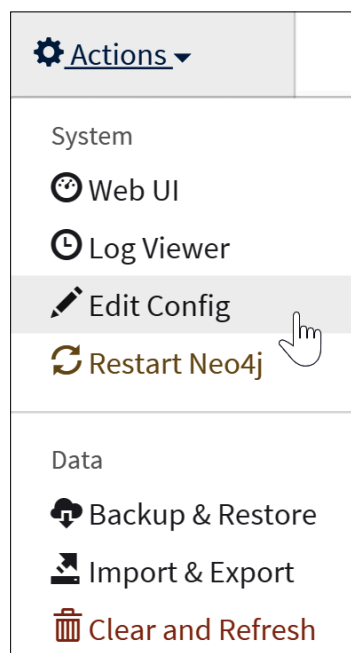
Im Unterschied zu Administration, Betrieb und Bereitstellung/Deployment von Datenbank und Anwendungen erfolgt deren Programmierung in der gewohnten Entwicklungsumgebung auf der lokalen Maschine.

Nach dem Anlegen einer Graph-Datenbank erreicht man über den Bereich des Icons *Graph List* mittels der Dropdown-Liste *Actions* verschiedene Hosting-Features (Bild 17). Zu den wichtigsten Funktionen für den Einstieg in die DBaaS-Dienste gehören Web UI, Edit Config und Clear and Refresh.

Web UI öffnet eine Neo4j-Console, um sich bei der Graph-Datenbank anzumelden. Die zugehörigen Anmeldedaten für Username und Password befinden sich im gleichnamigen Feld im Bereich des Icons *Graph List*. Bei *Edit Config* kann man im sich öffnenden Fenster die Systemeinstellungen für die Neo4j-Software festlegen. Ein Klick auf *Save settings and restart Neo4j!* führt einen Neustart der Datenbank mit den gewünschten Einstellungen durch. Über *Backup & Restore* kann man auf der Host-Plattform eine Sicherung vornehmen oder über eine vorhandene Datensicherung einen alten Datenbestand in die Graph-Datenbank auf dem Host einspielen. Mit den Funktionen *Import* und *Export*



Die Weboberfläche von Graph Story macht über eine Sidebar Dokumentation und Funktionen der Graph Story Console zugänglich (Bild 15)



Alle wichtigen Aufgaben zur Administration und Betrieb von Neo4j findet man in der Dropdown-Liste *Actions* (Bild 17)

Neo4j-REPL-Konsole als Alternative der Sandbox

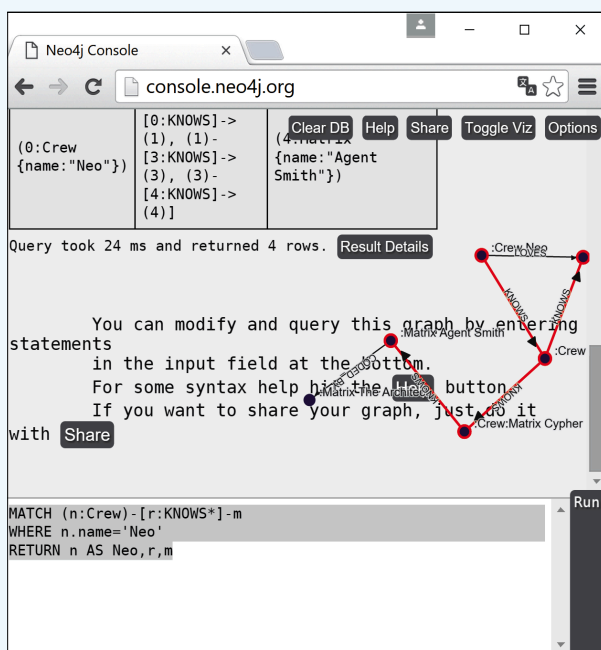
Für interaktives, visuelles Arbeiten mit Neo4j und Testen von Cypher-Statements steht eine weitere Neo4j/Cypher-Konsole über das Web zur Verfügung.

Neben Cypher unterstützt die REPL als Austauschformat für Graphen auch Geoff (Graph Entity Offline File Format) von Nigel Small.

Diese Neo4j/Cypher-Konsole erreicht man über zwei verschiedene Websites: Rabbithole oder Web UI Neo4j. Beide Sites implementierten Neo4j-Programmieren über GitHub. Während Rabbithole die Cloud-Application-Plattform Heroku hostet, stellt Neo Technology die Web UI Neo4j Console bereit.

Die Neo4j-REPL sollte man nicht mit dem im Datenbank-Management-System integrierten Neo4j-Browser (<http://localhost:7474/>) verwechseln.

Ein Klick auf die im Bild ganz rechts stehende Options-Schaltfläche öffnet eine Auswahlliste, um die gewünschte Neo4j/Cypher-Version festzulegen.



Neo4j-REPL realisiert eine Cypher-Shell als Read-Eval-Print Loop (REPL) für explorative Programmierung, um Cypher-Statements ohne die Installation eines Neo4j-Systems direkt auszuführen

Links zum Thema

- DB-Engines Ranking of Graph DBMS
<http://db-engines.com/en/ranking/graph+dbms>
- Homepage der Graph-Datenbank Neo4j
www.neo4j.com
- Geoff, ein Interchange Format für Neo4j-Datenbanken
<http://nigelsmall.com/geoff>
- Download-Seite von Neo4j
<http://neo4j.com/download>
- Webseite mit den Neo4j Release Notes
<http://neo4j.com/release-notes>
- Homepage der Rabbithole Neo4j Console
<http://rabbithole.herokuapp.com>
- Homepage der Web UI Neo4j Console
<http://console.neo4j.org>
- Unterschiede zwischen Enterprise und Community Edition
<http://neo4j.com/editions>
- Informationen und Hinweise zum Neo4j Startup Program
<http://neo4j.com/startup-program>
- Homepage zum Neo4j-OEM-Partnerprogramm
<http://neo4j.com/partners/oem-partner>
- Homepage der GraphGists
<http://graphgist.neo4j.com>
- Homepage des Open-Source-Projekts openCypher
www.opencypher.org
- Slack Group für Neo4j-Anwender
<https://neo4j-users.slack.com>
- StackOverflow-Gruppe für Neo4j-Anwender
<http://stackoverflow.com/questions/tagged/neo4j>
- Lokale Neo4j-Benutzergruppen
www.meetup.com/de-DE/topics/neo4j
- Anleitungen für Neo4j-Debian-Packages
<http://debian.neo4j.org>
- Tutorial zum Neo4j-Browser
<http://neo4j.com/developer/guide-neo4j-browser>
- Hosting Neo4j in the Cloud
<http://neo4j.com/developer/guide-cloud-deployment>

stößt man lediglich die Host-Plattform an; die Sicherung der Daten erfolgt anschließend in Eigenregie. Beim Export erhält man einen Link für das Herunterladen der Datensicherung. Der Aufbau der ZIP-Datei entspricht dem üblichen Dateiaufbau und der Verzeichnisstruktur einer lokalen Neo4j-Datenbank (Unterverzeichnis `/data/graph.db` einer Neo4j-Installation). `Import` übernimmt eine entsprechende ZIP-Datei in den aktuell angemeldeten Graph-Story-Account als neue Datenbank der Hosting-Plattform. ■



Frank Simon

arbeitet in der Software-Entwicklung mit den aktuellen Arbeitsgebieten Entwicklung, Programmierung, Test und Debugging von Cloud-, Rich-Internet-, Mobile- und Web-Anwendungen inklusive deren System-Management.
web_mobile_developers@gmx.eu

UNBEKANNTERE CSS3-FEATURES

Nützliches CSS3

CSS3 bietet auch weniger bekannte, aber durchaus nützliche Features.

Außerhalb der umfassenden Spezifikationen in CSS3 zu Flexbox, Media Queries und Co. existieren auch weniger bekannte Spezifikationen mit nützlichen kleineren Features, die den CSS-Alltag erleichtern – eine Auswahl stellen wir Ihnen hier vor (Tabelle 1). Ausgewählt wurden dabei nur Techniken, deren Browserunterstützung einem Einsatz in der Praxis nicht im Weg steht: Da wäre etwa *currentColor*, das Farbanpassungen vereinfacht, *calc()* für flexiblere Breiten, *object-fit* für die richtigen Anpassungen von Bildern und Videos und mehr.

currentColor

Wahrscheinlich ist Ihnen von der Arbeit mit CSS-Präprozessoren das Konzept der Variablen in CSS bereits bekannt. Diese sind auch in CSS3 vorgesehen, allerdings ist die Browserunterstützung derzeit noch nicht so weit fortgeschritten, dass

wir sie nutzen könnten. Was allerdings heute schon gut funktioniert, ist der Wert *currentColor*. Damit können Sie festlegen, dass der für *color* geltende Wert des aktuellen Elements für andere Farbangaben verwendet wird. In folgendem Beispiel ist der Schatten rot, weil *color* auf *red* gesetzt ist und bei *box-shadow* als Farbe *currentColor* angegeben wird:

```
.button {
  color: red;
  box-shadow: 2px 3px 3px currentColor;
}
```

Und hier ist der Schatten entsprechend hingegen grau:

```
.button {
  color: #333;
```

```
  box-shadow: 2px 3px 3px
  currentColor;
```

```
}
```

currentColor können Sie als Wert bei *border-color*, *outline*, *box-shadow*, *background*, *linear-gradient* et cetera einsetzen – also bei allen Angaben, die eine Farbangabe erwarten.

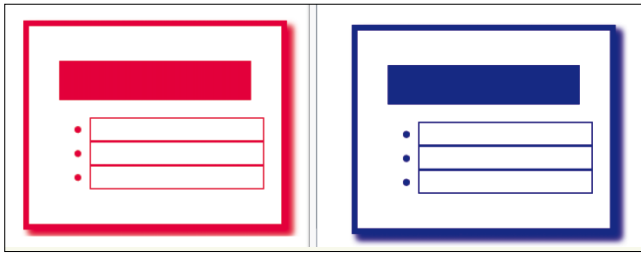
Übrigens ist bei *border-color* der Default-Wert *currentColor*. Wenn Sie einen Rahmen ohne Rahmenfarbe definieren, wird der bei *color* angegebene Wert genommen. Dies gilt ebenfalls für *outline* oder *box-shadow*. Außerdem ist *currentColor* die Standardfarbe bei *alt*-Texten von Bildern und Aufzählungszeichen.

Einfache Änderungen

Prinzipiell ist *currentColor* sehr praktisch beim einheitlichen, gleichzeitig aber flexiblen Stylen von Elementen der Bedienoberfläche (Bild 1). Dabei müssen Sie nur die Farbe des umfassenden Ele-

Tabelle 1: Browserunterstützung der vorgestellten Eigenschaften

Feature	Anmerkung	Chrome/ Opera	Firefox	Safari	IE/Edge
<i>currentColor</i>	Verwendet den aktuell definierten <i>color</i> -Wert	ja	ja	ja	ab 9
<i>calc()</i>	Berechnete Längenangaben	ja	ja (aber nicht bei Farbangaben oder bei <i>line-height</i>)	ja	ab 9 (aber nicht bei Farbangaben)
<i>object-fit</i>	Anwendbar auf ersetzte Elemente wie Bilder zur Bestimmung der Anpassung	ja	ja	ja	nein, Edge unter consideration, Nachbessern per JS
<i>object-position</i>	Positionierung eines ersetzten Elements, etwa eines Bildes	ja	ja	nein	nein
<i>min-content</i>	Neues Schlüsselwort für <i>width/max-width</i> et cetera	ja	ja	ja	Nachbessern möglich durch Aktivierung der Gridlayout-Darstellung
Quantity Queries	Besondere Formatierung ab bestimmter Anzahl an Elementen über Selektoren wie <i>:nth-last-child()</i>	ja	ja	ja	ab 9
<i>background-position</i> mit Schlüsselwort und Längenangabe	Zum Beispiel <i>background-position: bottom 20px right 50px;</i>	ja	ja	ja	ab 9



Dank **currentColor** genügt es, die Farbe an einer Stelle anzupassen, damit sie sich überall anpasst (Bild 1)

ments ändern, und alle Bestandteile, bei denen *currentColor* verwendet wird, passen sich automatisch an.

Praktisch ist *currentColor* ebenfalls, wenn Sie beispielsweise eine *hr*-Linie einsetzen wollen, die dieselbe Farbe haben soll wie der Text. Wenn diese dann weniger kräftig sein soll als die Textfarbe, lässt sich das über *opacity* bewerkstelligen:

```
.artikel {
  color: #333;
}
.artikel hr {
  width: 30%;
  height: 0.313.em;
  border: none;
  background-color: currentColor;
  opacity: 0.2;
}
```

Die Farbe von Links lässt sich darüber ebenfalls an die umgebende Textfarbe anpassen:

```
p a {
  color: currentColor;
}
```

Sehr nützlich ist *currentColor* auch bei SVG-Icons, die Sie auf diese Art immer passend zum Text einfärben können. Wenn das SVG-Element die *id="icon"* hat, geht das folgendermaßen:

```
#icon {
  fill: currentColor;
  stroke: currentColor;
}
```

Eine weitere Einsatzmöglichkeit für *currentColor* sind Animationen, bei denen mehrere Farben animiert geändert werden – wobei aber dank *currentColor* nur das umfassende Element eine andere Farbe erhält.

Die Browserunterstützung für *currentColor* ist sehr gut, einzig der IE 8 bleibt außen vor, was aber im Normalfall kein Problem darstellt.

Viele CSS-Eigenschaften erwarten als Werte Längenangaben – beispielsweise *width*, *height*, *border-width*, *left*, *top* et cetera. Bei anderen Eigenschaften wird eine Längenangabe

mit anderen Angaben kombiniert, etwa bei Schatten oder Farbverläufen:

```
box-shadow: 2px 2px 3px black;
background: linear-gradient(white 50%, black 50%);
```

Der Farbverlauf im Beispiel definiert einen harten Übergang, weil genau bei 50 Prozent Weiß auf Schwarz wechselt.

Als Längenangaben können Sie die klassischen Einheiten wie px, % oder em einsetzen; oder Sie greifen auf die neuen CSS3-Einheiten wie vw, vh oder rem zurück. Eine Alternative wäre der Einsatz von *calc()*, mit dem Sie Berechnungen durchführen können. Eine Angabe wie

```
width: calc(20% + 5%);
```

hat dieselbe Auswirkung wie das Folgende:

```
width: calc(25%);
```

Das Interessante an *calc()* ist, dass sich Einheiten mischen lassen. Sie können beispielsweise vorgeben, dass ein Element 30 % minus 10px breit sein soll. Wie breit das Element dann wirklich ist, hängt von den aktuellen Gegebenheiten ab.

```
width: calc(30% - 10px);
```

Die Möglichkeit, Einheiten zu mischen, ist der entscheidende Unterschied zu den Rechenmöglichkeiten, die bei der Verwendung von CSS-Präprozessoren zur Verfügung stehen.

calc() können Sie in den unterschiedlichsten Situationen einsetzen und mit allen Einheiten, die CSS bietet. Hier ein paar Beispiele, die auch die ungewöhnlicheren Fälle zeigen:

```
font-size: calc(1em + 10px);
transform: rotate(calc(1turn - 16deg));
border-color: hsl(180, calc(2*15%), 65%);
/* calc() bei Farbangaben funktioniert nur in Chrome */
line-height: calc(2*1);
/* calc() wird bei line-height nicht im FF unterstützt */
width: calc(10vmin - 2rem);
```

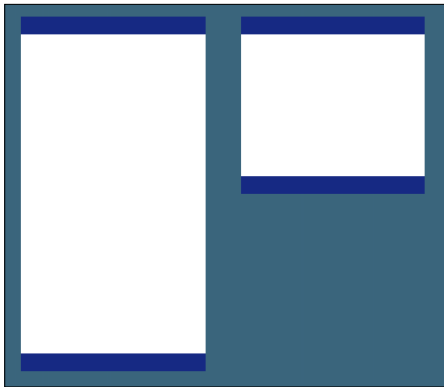
Wie bei allen Funktionen in CSS müssen Sie die Klammern direkt hinter dem Funktionsnamen schreiben:

```
width: calc (10px + 10%); /* falsch */
width: calc(10px + 10%); /* richtig */
```

Beachten Sie, dass Plus und Minus von Leerzeichen umgeben sein müssen:

```
width: calc(10px+10%); /* falsch */
width: calc(10px + 10%); /* richtig */
```

calc() hilft in vielen Fällen. Nehmen wir an, Sie wollen zwei Bereiche nebeneinander platzieren – der eine hat eine fes- ►



Mit `calc()` lässt sich ein Farbverlauf definieren, der unten und oben einen 1em breiten Streifen erzeugt – unabhängig davon, wie hoch das Element ist (Bild 2)

te Breite – im Beispiel 300px –, der andere soll den restlichen Platz einnehmen. Dann können Sie mit `calc()` die Breite des zweiten Elements berechnen lassen:

```
.box1 { width: 300px; }
.box2 { width: calc(100% - 300px); }
```

Ein weiteres Beispiel sind Elemente mit Breitenangaben in Prozent und Rahmen. Die Besonderheit bei Rahmenbreiten ist, dass Sie dort keine Prozentangaben einsetzen können:

```
.box1 {
  width: 20%;
  border-left: 10px solid blue;
}
```

Was macht man jetzt, wenn man ein weiteres Element definieren möchte, dass dieselbe Breite wie `.box1` hat, aber keinen Rahmen? Mit `calc()` können Sie dem Browser die Berechnung der Breite überlassen:

```
.box2 {
  width: calc(20% + 10px);
}
```

Eine Alternative zu dieser Lösung besteht in der Aktivierung des anderen Boxmodells durch `box-sizing: border-box`. Dadurch repräsentiert `width` die Gesamtbreite, das heißt, mögliches `padding` oder `border` werden von dort aus nach innen gezeichnet. Damit ist aber die Box insgesamt nur noch 20 Prozent breit. Außerdem können Sie dank `calc()` einen Farbverlauf definieren, der immer unten und oben einen 1em hohen Streifen in einer anderen Farbe hat – unabhängig davon, wie hoch das Element ist (Bild 2):

```
background:
  linear-gradient(blue 1em, #fff 1em,
    #fff calc(100% - 1em),
    blue calc(100% - 1em));
```

Bei diesem Farbverlauf bestimmen die ersten beiden Angaben, dass Blau bis zu 1em reichen und dann Weiß beginnen soll. Die beiden folgenden Angaben sorgen für den 1em brei-

ten Randstreifen am Ende des Elements – die entsprechende Position liefert `100% - 1em`. Ohne `calc()` müssten Sie dafür zwei separate Farbverläufe definieren.

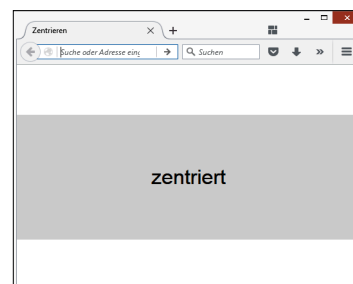
Mindestaußenabstand

In CSS gibt es die nützlichen Angaben `min-width` und `min-height`. Manchmal bräuchte man aber auch so etwas wie `min-margin`, das es allerdings nicht gibt. Dank `calc()` lässt sich jedoch ein entsprechendes Verhalten simulieren. Im folgenden Beispiel wollen wir einen Mindestaußenabstand rechts und links von 1,5em:

```
div {
  margin: 0 auto;
  max-width: calc(100% - 3em);
  width: 600px;
}
```

Durch diese Angaben ist das Element normalerweise 600px breit und hat eine Maximalbreite von `100% - 3em`. Wenn weniger Platz als 600px zur Verfügung steht – weil zum Beispiel das Browserfenster schmaler ist –, kann das Element auch schmaler werden. Aber egal wie schmal das Element wird: Es hat immer mindestens 1,5em seitlichen Außenabstand (3em geteilt durch 2). Das funktioniert deshalb, weil sich

die `max-width`-Angabe durchsetzt, wenn die bei `max-width` angegebene Breite kleiner ist als diejenige bei `width`.



Vertikale Zentrierung: Auch bei der vertikalen Zentrierung hilft `calc()` (Bild 3)

Sehen wir uns zuerst die klassische Methode zur Zentrierung an, die wir danach mit `calc()` verbessern.

```
.zentriert {
  height: 20em;
  position: absolute;
  top: 50%;
  margin-top: -10em;
}
```

Die klassische Methode basiert darauf, dass das Element absolut positioniert und über `top: 50%` die linke obere Ecke in der Mitte platziert wird. Da sich aber nicht die linke Ecke, sondern die Mitte des Elements in der Mitte befinden soll, müssen wir das Element um die Hälfte seiner eigenen Höhe nach oben verschieben. Das Element ist im Beispiel 20em hoch, deswegen benutzen wir `margin-top: -10em`.

Besser zentrieren mit calc()

Das vertikale Zentrieren ist eine klassische und gleichzeitig nicht banale Aufgabe in CSS. Nehmen wir an, das Element, das zentriert werden soll, ist 20em hoch.

Dank *calc()* können Sie sich die Angabe mit *margin-top* sparen, denn Sie können den benötigten Wert direkt bei *top* abziehen (Bild 3):

```
.zentriert {
  height: 20em;
  position: absolute;
  top: calc(50% - 10em);
}
```

Eine Alternative zu der vorgestellten Technik zum Zentrieren besteht im Einsatz von *transform: translateY(50%)*:

```
.zentriert {
  position: relative;
  top: 50%;
  -webkit-transform: translateY(-50%);
  -ms-transform: translateY(-50%);
  transform: translateY(-50%);
}
```

Falls Sie aber ein Element unter Zuhilfenahme von *translate()* animieren wollen, ist es einfacher, nicht auch schon beim Zentrieren *translate()* zu benutzen – und dann ist die Variante mit *calc()* das Mittel der Wahl.

Intuitive Rasterbreiten

Grundsätzlich liegt die Stärke von *calc()* in der Kombination von unterschiedlichen Einheiten – in manchen Fällen bietet *calc()* aber auch schlichtweg eine Arbeitserleichterung. So können Sie mit *calc()* schnell und intuitiv die benötigten Breiten für ein Raster festlegen – im Beispiel sind es fünf Spalten:

```
.col-1 { width: calc(100% / 5*1); }
.col-2 { width: calc(100% / 5*2); }
.col-3 { width: calc(100% / 5*3); }
.col-4 { width: calc(100% / 5*4); }
```

Für die Rasterdefinition geben Sie hinter dem / die gewünschte Gesamtspaltenanzahl und nach * die Anzahl der Spalten an, über die sich das Element erstrecken soll.

Die Browserunterstützung für *calc()* ist sehr gut, es funktioniert ab dem Internet Explorer 9 und ansonsten in allen aktuellen Browsern. Allerdings gibt es ein paar Einschränkungen beziehungsweise Besonderheiten: *calc()* bei Farbangaben funktioniert nicht im Firefox oder IE, wird aber in Safari oder Chrome interpretiert. Außerdem interpretiert Firefox *calc()* nicht bei *line-height*. Prinzipiell gibt es einige Probleme in den nativen Android-Browsern – hier ist es am sichersten, nur mit Plus und Minus zu arbeiten.

Bilder beschneiden und mehr

In der Spezifikation »CSS Image Values and Replaced Content Module Level 3« gibt es eine Reihe von neuen Optionen für die Behandlung und Flexibilisierung von Bildern. Die Spezifikation ist eine Candidate Recommendation und damit im Standardisierungsprozess schon weit fortgeschritten.

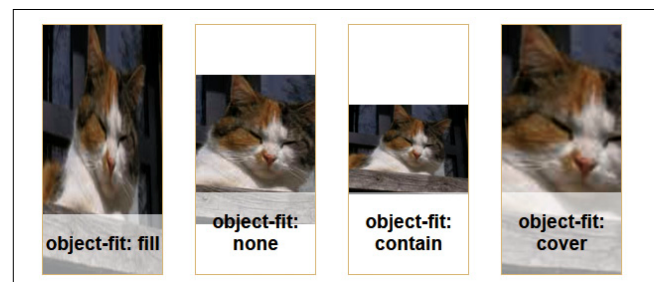
Bei Hintergrundbildern erweist sich die Eigenschaft *background-size* mit den Angaben *cover* und *contain* als praktisch – damit definieren Sie elementfüllende Hintergrundbilder und können zwischen verschiedenen Optionen wählen, wenn die Proportionen des Bildes und des Elements, dem es zugewiesen wird, nicht übereinstimmen. Eine ähnliche Möglichkeit fehlt aber derzeit bei normalen im HTML-Code stehenden Bildern und anderen sogenannten ersetzten Inhalten (replaced contents). Hier kommt die neue Eigenschaft *object-fit* ins Spiel.

Inhärente Breite

Gehen wir von einem Bild aus, dessen inhärente Breite 200 px × 150 px beträgt. Per CSS wird eine Breite von 120 px × 250 px definiert. Im Beispiel wurden absichtlich unterschiedliche Proportionen gewählt:

```
img {
  width: 120px;
  height: 250px;
  display: block;
}
```

Mit Hilfe von *object-fit* können Sie jetzt steuern, wie diese sich widersprechenden Größenangaben aufgelöst wer-



Verschiedene Werte für *object-fit* von links nach rechts: *fill*, *none*, *contain* und *cover* (Bild 4)

den. Soll das Bild etwa die gesamten Ausmaße ausfüllen und bei Bedarf abgeschnitten werden, so notieren Sie beim Bild *object-fit: cover*:

```
img {
  object-fit: cover;
}
```

Dies entspricht dem Verhalten von *background-size: cover* bei Hintergrundbildern.

Bei *object-fit* sind folgende Werte möglich (Bild 4):

- *object-fit: fill* (Standardwert) – das Bild wird so gestreckt, dass es die Ausmaße ausfüllt. Das bedeutet, dass das Bild dabei auch verzerrt werden kann, wenn es andere Proportionen hat als die zugewiesene Breite.
- *object-fit: none*: Das Bild wird nicht angepasst, sondern bei Bedarf beschnitten. ►

- *object-fit: contain* sorgt dafür, dass das Bild ganz zu sehen ist – aber im Zweifelsfall gibt es noch Platz drumherum.
- *object-fit: cover* sorgt dafür, dass die Ausmaße ganz ausgefüllt werden.
- *object-fit: scale-down* verhält sich so, als wäre *none* oder *contain* bestimmt, je nachdem, welches der beiden zu einem kleineren ersetzten Element führt.

Die Browserunterstützung für die Eigenschaft *object-fit* ist gut, nur der IE spielt nicht mit. Für die Microsoft-Browser gibt es mehrere JavaScript-Lösungen – beachten Sie dabei, dass manche der JS-Lösungen nur im Internet Explorer funktionieren, aber nicht in Edge. Dieses Problem hat die Lösung von Primož Cigler nicht: Sie funktioniert im Prinzip in allen Microsoft-Browsern und basiert darauf, dass in nicht unterstützten Browsern per JavaScript das Bild als Hintergrundbild benutzt wird – denn dann lässt sich *background-size: cover* oder *background-size: contain* nutzen.

Alles in Position

Wenn das Bild dank *object-fit: contain* kleiner ist als der zur Verfügung stehende Platz, können Sie die Position des Bildes über *object-position* bestimmen. *object-position* versteht dieselben Werte, die auch bei *background-position* möglich sind – Schlüsselwörter wie *top*, *bottom*, *left*, *right* oder Längenangaben (Bild 5).

Allerdings ist die Browserunterstützung für *object-position* etwas schlechter als bei *object-fit*, da Safari zwar *object-fit*, aber nicht *object-position* implementiert hat.

Wofür braucht man *object-fit*? Sehr gut eignet es sich zum Anpassen von Bildern beim responsiven Webdesign. Die klassische Methode *max-width: 100%; height: auto* ist nur sinnvoll, wenn die Proportionen der Bilder bei allen Viewport-Breiten gleich bleiben sollen – ansonsten bietet *object-fit* mehr Steuerungsmöglichkeiten.

Links zum Thema

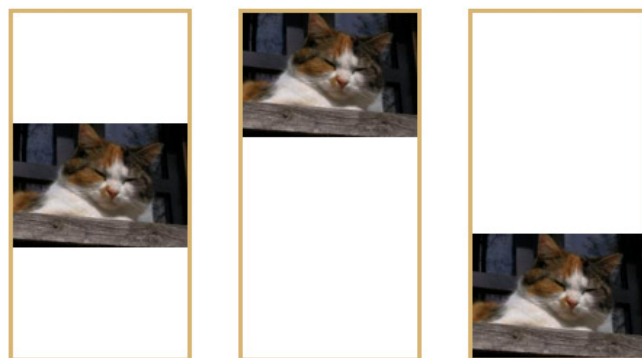
- *currentColor* nutzen
<http://blogs.adobe.com/dreamweaver/2015/02/extending-the-color-cascade-with-the-css-currentcolor-variable.html>
- Einsatzmöglichkeiten von *calc()*
<https://www.smashingmagazine.com/2015/12/getting-started-css-calc-techniques/>
- Fallback für *object-fit* im Internet Explorer
<https://github.com/anselmh/object-fit>
- Fallback für *object-fit* in Edge
<https://medium.com/@primozcigler/neat-trick-for-css-object-fit-fallback-on-edge-and-other-browsers-afbc53bbb2c3#.144pynd6j>
- *min-content* für Bildunterschriften
<http://thenewcode.com/662/Design-From-the-Inside-Out-With-CSS-MinContent>

Ein Beispiel für unterschiedliche Proportionen wäre ein quadratisches Vorschaubild, bei dem das zugehörige Bild Quer- oder Hochformat hat. Dank *object-fit* sind unterschiedliche Proportionen kein Grund für doppelte Bilder.

Interessante Effekte mit *object-fit* sind insbesondere bei Galerien möglich. Ein weiteres Beispiel für den Einsatz von *object-fit* sind bildschirmfüllende Videos – *object-fit* lässt sich bei allen ersetzten Elementen benutzen, und dazu gehören auch Videos.

Passende Bildunterschriften

Dank RWD müssen Bilder und ähnliche Elemente flexibel sein. Schwierig wird es, wenn man eine Bildunterschrift hat, die sich an das flexible Bild anpassen soll – und genau an die-



Mit *object-position* lässt sich ein Element platzieren (Bild 5)

ser Stelle kommen neue Schlüsselwörter ins Spiel, die bei Breitenangaben geschrieben werden.

Sehen wir uns das Beispiel an. Wir haben ein *figure*-Element mit unserem flexiblen Bild und einer Bildbeschriftung:

```
<figure>
  
  <figcaption>Bildbeschriftung, die auch etwas länger
    gehen kann ...</figcaption>
</figure>
```

Normalerweise wird das *figcaption*-Element so breit, wie es der Text erfordert. Wir wollen aber, dass die Beschriftung bei Bedarf umbricht, damit sie nur so breit ist wie das Bild. Hier kommt die neue Angabe *min-content* ins Spiel, die wir für *figure* definieren:

```
figure {
  width: -moz-min-content;
  width: -webkit-min-content;
  width: min-content;
}
```

Dank *max-width: min-content* wird das *figure*-Element nur so breit wie das schmalste Kindelement – in diesem Fall das Bild. Damit haben wir eine Bildbeschriftung, die sich an das flexible Bild anpasst (Bild 6).

Alle modernen Browser unterstützen diese neue Angabe, ausgenommen der Internet Explorer. Um diesen mit ins Boot zu holen, können Sie den Code durch ein umfassendes Element erweitern und für dieses die Grid-Darstellung im IE aktivieren – in diesem Kontext verstehen die Microsoft-Browser *min-content*:

```
<div class="container">
  <figure>
    
    <figcaption>
      Bildbeschriftung, die auch
      etwas länger gehen kann
    ...</figcaption>
  </figure>
</div>
```

Zusätzlich zur Angabe von *min-content* für das *figure*-Element aktivieren Sie die Gridlayout-Darstellung für den Internet Explorer:

```
.container {
  display: -ms-grid;
  -ms-grid-columns: min-content;
}
```

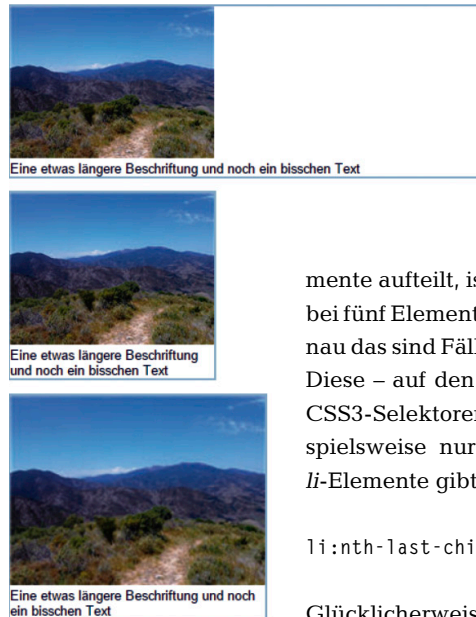
Die neue Angabe *min-content* stammt aus der Spezifikation »CSS Intrinsic & Extrinsic Sizing Module Level 3«. Neben dem Wert *min-content* sind weitere Werte vorgesehen:

- Bei *max-content* wird das Elternelement so breit/hoch wie das größte Kindelement.
- Mit *fit-content* können Sie das aus CSS 2.1 bekannte Shrink-to-fit-Verhalten explizit aktivieren. Shrink-to-fit greift beispielsweise bei gefloateten Elementen ohne Breitenangabe: Sie werden so schmal wie möglich.
- *fill*: Damit füllt ein Element sein Elternelement aus. Diese Eigenschaft hieß in der Spezifikation ursprünglich *fill-available* und ist unter unterschiedlichen Namen in den Browsern implementiert: Firefox versteht *-moz-available* und Chrome *-webkit-fill-available*.

Apropos Browserunterschiede: Im Firefox werden diese neuen Angaben prinzipiell nur bei *width*, *min-width*, *max-width*, aber nicht bei *height*-Angaben unterstützt.

Kreative Nutzung

Und manchmal sind es nicht neue Eigenschaften, sondern neue Kombinationen bekannter Features, die sich als Problemlöser entpuppen – so etwa die quantitativen Queries. Webseiten haben oft auch flexible Inhalte, das heißt, dass et-



Standardverhalten
(oben), die beiden unteren zeigen *min-content*, das unabhängig von der Bildgröße funktioniert (Bild 6)

wa aus den sechs vorgesehenen Hauptnavigationspunkten auch einmal acht werden können oder dass bei manchen Unterseiten mehr Textboxen benötigt werden als auf anderen. Eine geänderte Anzahl bedeutet meist, dass eine andere Formatierung notwendig ist. Wenn man den verfügbaren Platz auf vier Elemente aufteilt, ist eine Breitenangabe von 25 Prozent richtig, bei fünf Elementen sind es hingegen 20 Prozent et cetera. Genau das sind Fälle für die sogenannten quantitativen Queries. Diese – auf den ersten Blick kryptischen – Angaben nutzen CSS3-Selektoren sehr kreativ. Wenn eine Formatierung beispielsweise nur dann gelten soll, wenn es mehr als vier *li*-Elemente gibt, schreiben Sie:

li:nth-last-child(n+5), li:nth-last-child(n+5) ~ li { }

Glücklicherweise müssen Sie den Code nicht selbst erstellen, sondern können sich von dem Online-Tool www.quantityqueries.com bei der Erstellung helfen lassen.

Glücklicherweise müssen Sie den Code nicht selbst erstellen, sondern können sich von dem Online-Tool www.quantityqueries.com bei der Erstellung helfen lassen.

Wenig bekannte Optionen

Und wiederum in anderen Fällen sind die möglichen Optionen weniger bekannt – so bei *background-position*. Klassischerweise geben Sie bei *background-position* Schlüsselwörter an wie *top*, *bottom* et cetera, oder Sie schreiben zwei Längenangaben, mit denen Sie die linke obere Ecke platzieren:

```
background-position: 20px 50px;
```

Neu in CSS3 ist, dass sich beides kombinieren lässt. Damit können Sie festlegen, dass ein Hintergrundbild 20 px von unten und 10 px von rechts platziert werden soll – und das funktioniert eben auch, wenn Sie nicht wissen, wie hoch das Element überhaupt ist – sehr praktisch:

```
background-position: bottom 20px right 50px;
```

Am Anfang des responsiven Webdesigns ahnte wohl niemand, was die nun geforderte Flexibilität in allen Details bedeutet. Glücklicherweise bekommen wir auch nach und nach die dafür erforderlichen Tools an die Hand. Denn eines haben die im Artikel vorgestellten CSS3-Features gemeinsam: Sie helfen, die Anzeige zu steuern, auch wenn man die genaue aktuelle Breite oder Position der Elemente nicht kennt. ■



Florence Maurice

ist Autorin, Trainerin und Programmiererin in München. Sie schreibt Bücher zu PHP und CSS3 und gibt Trainings per Video. Außerdem bloggt sie zu Webthemen unter:

<http://maurice-web.de/blog>

TONIC: INTERAKTIVE NODE.JS-UMGEBUNG IM BROWSER

Code-Sandkasten

Die Idee von webbasierten Code-Sandkästen ist nicht neu.

Zu den bekanntesten Exemplaren gehören JSbin (<http://jsbin.com>) von Remy Sharp und JSfiddle (jsfiddle.net) von Piotr Zalewa. Beide bieten eine in den Webbrowser integrierte Entwicklungsumgebung, um HTML, CSS und JavaScript direkt im Browser zu editieren und – noch viel wichtiger – die Ergebnisse direkt auszuprobieren. Die so erstellten Projekte sind öffentlich verfügbar und können trivial per Link weitergegeben werden: ideale Voraussetzung für Code-Beispiele und kleine Experimente. Tonic ist ebenfalls ein Code-Sandkasten – aber für serverseitige Programmierung mit Node.js (<https://tonicdev.com>).

Browser-Umgebung

Die meisten Code-Sandkästen präsentieren sich als Browser-Umgebungen und decken demnach nur den clientseitigen Teil ab. Hier unterscheidet sich Tonic von Tools wie JSbin und JSFiddle: Anstelle der Browser-Engine stehen dem Entwickler sämtliche Möglichkeiten der Entwicklung mit Node.js zur Verfügung.

Die Entwickler von Tonic sehen ihr Produkt deshalb auch als einen besseren REPL (Read-Eval-Print Loop, Konsole) für die serverseitige JavaScript-Umgebung mit Node.js. Man kann sogar direkt mittels *require* Node.js-Module laden. Ein

separater Installationsschritt mit dem Node Package Manager ist unnötig. Man kann einfach drauflosprogrammieren, als ob bereits alle der mehr als 200.000 NPM-Module installiert wären.

Anmeldung

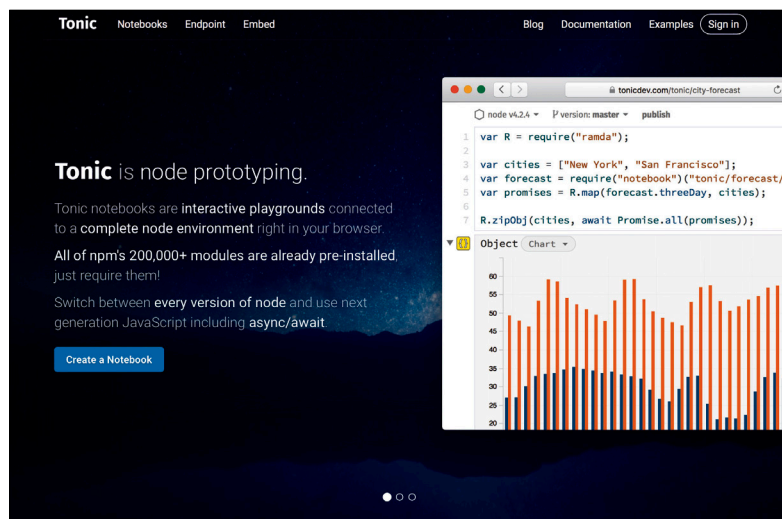
Die Anmeldung über die Website (Bild 1) ist bislang kostenlos, und es sind noch keine Details über eine Kommerzialisierung des Projekts bekannt. Als Besucher der Seite kann man den Anmeldeschritt sogar überspringen. Wenn man jedoch auch später noch Zugriff auf die erstellten Dokumente haben möchte, muss man Benutzername, Passwort und die eigene E-Mail-Adresse eingeben.

Ganz gleich ob angemeldet oder übersprungen – es kann direkt losgehen: Der Nutzer sieht ein Notebook mit dem Titel *Welcome*. Alle Projekte werden bei Tonic als Notebooks erstellt. Diese können einerseits herkömmlichen Text enthalten, aber auch sogenannte Code-Zellen mit ausführbarem JavaScript-Code.

In der Kopfzeile eines Notebooks kann man die Version der Node.js-Engine wählen, mit welcher



Foto: Sentavio / Shutterstock



Website: Die Webseite von Tonic, auf der man sich anmelden kann (Bild 1)

der Code ausgeführt werden soll, und ebenso zwischen unterschiedlichen Versionsständen des Notizbuchs wählen.

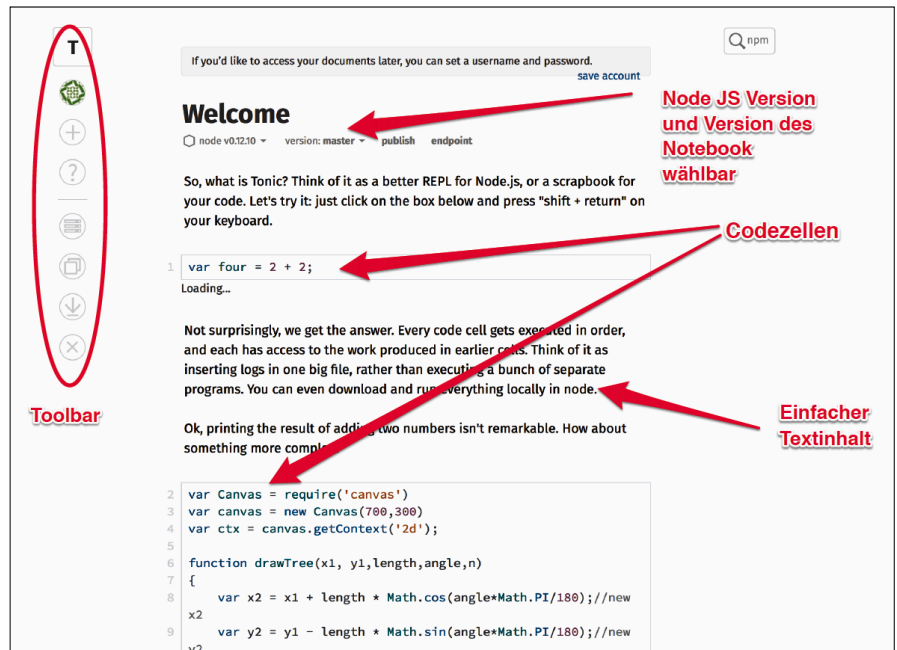
Das Notizbuch

Im Unterschied zu JSBin oder JSFiddle folgt Tonic weniger den typischen Metaphern einer Entwicklungsumgebung. Stattdessen präsentiert es sich in Form eines Arbeitsbuchs oder Notizbuchs – sehr ähnlich, wie man es von Programmen wie Mathematica kennt. Das erlaubt einen Programmierstil, der sehr gut für Code-Dokumentation mit Codebeispielen geeignet ist.

Es handelt sich gewissermaßen um eine interaktive Form des Literate Programming. Bild 2 zeigt die einzelnen Bereiche des Notizbuchs: Direkt unter dem Titel steht die gerade verwendete Version der Node.js-Engine und die Version des Notizbuchs.

Code und Text-Zellen

Ein Klick auf den Link *publish* öffnet einen Dialog, mit dem man ein Release, also eine Freigabe seines Dokuments, veröffentlichen kann (Bild 3). Auf diese Weise erhält das Dokument einen sprechenden Namen und eine eindeutige Versionskennung. Über den zweiten Link – *endpoint* – gelangt man zum URL eines für das Notizbuch definierten HTTP-Endpunkts. Solange aber kein solcher Endpunkt definiert ist, wird lediglich eine Fehlermeldung ausgegeben. Hier wären die Tonic-Entwickler eigentlich besser beraten, diesen Link auszublenden, solange es keinen Sinn ergibt, ihn zu nutzen.



Anatomie eines Tonic-Notizbuchs (Bild 2)

Der Inhalt des Notizbuchs besteht aus eine Reihe von Zellen – jeweils entweder einer Code-Zelle oder einer Text-Zelle. Die Text-Zellen bieten nicht viel mehr als einfachen Text ohne Formatierungen. Immerhin werden URLs als Verweise klickbar gemacht.

Insofern ist die Textdarstellung in Tonic bislang nicht mehr als eine visuell besser lesbare Darstellung von Quellcode-Kommentaren. In den meisten Fällen dürfte das auch reichen, aber vielleicht wären Zwischenüberschriften eine sinnvolle Ergänzung.

Ganz anders die Code-Zellen. Dahinter steckt jeweils eine Instanz des Browser-Code-Editors CodeMirror. Damit funktionieren auch Dinge wie automatische Einrückung oder Syntax-Highlighting.

Beispiel: Haltestellen der Deutschen Bahn

Der Nutzen eines interaktiven Node.js-Sandkastens wie Tonic lässt sich am besten anhand eines konkreten Programmbeispiels demonstrieren. Auf diese Weise kann man zeigen, wie ein Programm in einem Tonic-Notizbuch Schritt für Schritt entsteht.

Der erste Schritt ist dabei, ein neues Notizbuch anzulegen. Dazu klickt man zuerst auf das Plus-Symbol in der Toolbar. Es öffnet sich anschließend ein neues Notizbuch mit dem Titel *untitled notebook*. Mit einem Klick in den Titel wird dieser editierbar und kann auf eine passendere Bezeichnung wie *Deutsche Bahn: Haltestellen* geändert werden. Danach kann man gleich noch unter dem Titel die neueste Node.js-Version auswählen.

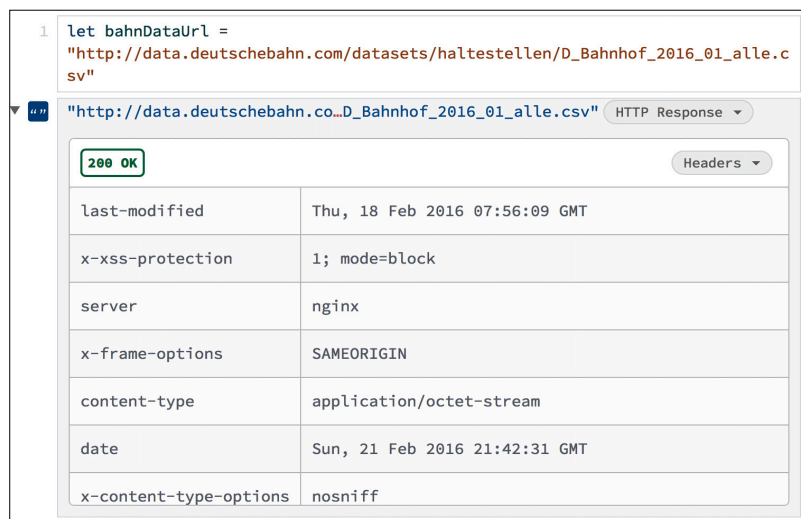
Der neue Name des Notizbuchs verrät schon ein Indiz über den Ausgangspunkt des Beispiels: Im Rahmen ihres Open-Data-Portals veröffentlicht die Deutsche Bahn Daten in maschinenlesbarer Form. Offen lizenziert und kostenlos liegen dort umfangreiche Datensätze direkt abrufbar zur Verfü- ►

Veröffentlichung: Über dieses Formular kann man das Notizbuch veröffentlichen (Bild 3)

gung. Einer dieser Datensätze ist eine CSV-Datei mit den Daten zu den Haltestellen. Den zugehörigen Download-URL kann man in einer Variablendeklaration hinterlegen:

```
let haltestellenURL = "http://data.deutschebahn.com/datasets/haltestellen/D_Bahnhof_2016_01_alle.csv"
```

Unterhalb der Code-Zelle weist Tonic darauf hin, dass man mit der Tastenkombination [Umschalt Eingabe] den Code ausführen kann. **Bild 4** zeigt das Ergebnis der Auswertung: Tonic visualisiert den Wert der letzten Anweisung einer Code-Zelle. In diesem Fall ist das eigentlich eine Zeichenkette mit dem URL – doch Tonic ist so intelligent, dass es das URL-Format erkennt und eine Visualisierung einer HTTP-Anfrage an diesen URL anzeigt. Es gab einen HTTP-Statuscode von **200 OK**, und auch die Kopfzeilen, die der Server in der Antwort mitschickt, werden gezeigt. Über die Auswahlbox mit der Beschriftung *HTTP Response* kann man die Aus-



Auswertung: So wird ein URL ausgewertet (**Bild 4**)

gabe auch auf *Full Text* umstellen – dann wird tatsächlich nur der Inhalt der Zeichenkette ausgegeben.

In der nächsten Code-Zelle sollen eine Reihe von Bibliotheken eingebunden werden, mit denen die CSV-Daten abgerufen und weiterverarbeitet werden können. Man hat in Tonic nahezu auf jede Bibliothek in NPM Zugriff – es reicht vollkommen, das Paket mit *require* zu laden:

```
let request = require('request-promise');
let {Converter} = require('csvtojson');
let _ = require("lodash/fp");
```

Schon das *let* im Beispiel mit dem Daten-URL zeigte, dass Tonic hier tatsächlich schon Syntax und Features aus ES2015 unterstützt. Beim Laden der Bibliotheken kann man sich somit auch die sehr nützliche Destrukturierungs-Syntax zunutze machen. Auf diese Weise wird der Converter gezielt an einen Namen gebunden.

Der Converter aus *csvtojson* ist ein Hilfsobjekt, mit dem man CSV-Daten nach JSON konvertieren kann. Dazu benötigt man eine Instanz davon, und mit der Methode *fromString()* kann man dann als Zeichenkette vorliegende CSV-Daten in ein JavaScript-Objekt konvertieren. Dazu kann man sich in einer weiteren Code-Zelle eine Hilfsfunktion *convertCSV()* definieren:

```
let converter = new Converter({
  delimiter: ";"
});
function convertCSV(csvString) {
  return new Promise((resolve,reject)=>{
    return converter.fromString(csvString,
      (err,result)=>!err? resolve(result) : reject(err));
  });
}
```

fromString() ist typisch für ein asynchrones API in Node.js: Die Funktion erhält als letzten Parameter eine Callback-Funktion, welche die Parameter *err* (für Fehler) und *result* für das berechnete Ergebnis enthält. Die Hilfsfunktion soll statt des Callback-Stils jedoch ein Promise zurückgeben, das mit dem fertig konvertierten Objekt erfüllt wird.

Callback-Stil-Funktionen

Hat man häufiger mit solchen Umwandlungen von Callback-Stil-Funktionen in Promise-Funktionen zu tun, so kann man auch das NPM-Paket *promisify-node* dafür nutzen. Ändern Sie dazu den Inhalt der Code-Zelle folgendermaßen ab:

```
let converter = new Converter({
  delimiter: ";"
});
let promisify = require('promisify-node');
converter.fromString = promisify
(converter.fromString);
function convertCSV(csvString) {
  return converter.fromString(csvString);
}
```

Wie man sieht, kann man in Tonic Code-Zellen jederzeit wieder verändern. Das Notizbuch aktualisiert sich daraufhin wieder in korrekter Weise.

Async/Await inklusive

Doch warum muss die Funktion denn unbedingt einen Promise benutzen? Ganz einfach: Tonic unterstützt bereits die als Vorschlag für ECMAScript eingereichte Syntaxerweiterung *Async/Await*. Damit kann man asynchronen Code, der Promises benutzt, besonders einfach schreiben. In einer neuen Code-Zelle wird dies gleich ausgenutzt, indem zuerst mit der bereits eingebundenen Funktion *request* die Daten des URL

abgerufen und im Anschluss mit `convertCSV` nach JSON konvertiert werden:

```
let data = await
request(haltestellenUrl);
let json = await convertCSV(data);
```

Bild 5 zeigt, wie Tonic das konvertierte Ergebnis anzeigt. Das Ergebnis ist ein Array aus Objekten, und in der Darstellung können die einzelnen Objekte aufgeklappt und anschließend deren Eigenschaften betrachtet werden.

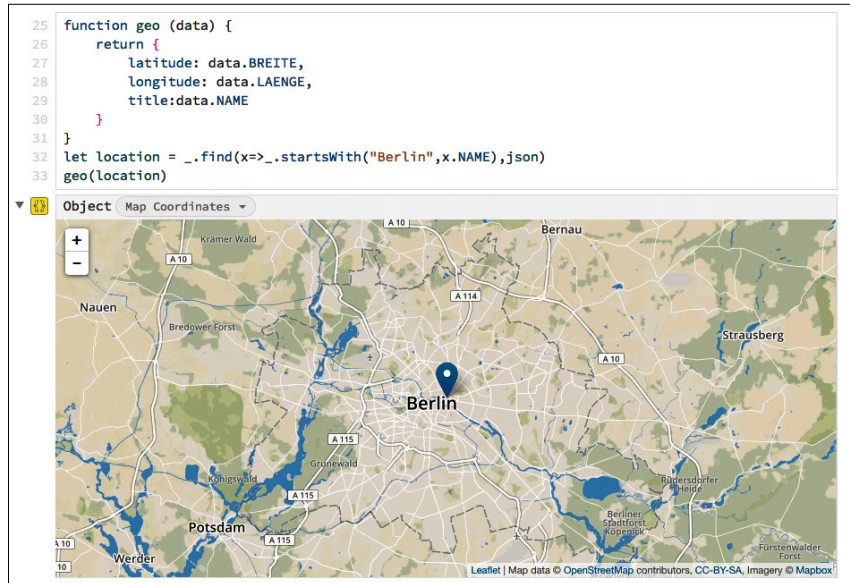
Bei einem derartig großen Array dauert die Ausgabe eine beträchtliche Zeit; immerhin erzeugt das Webinterface von Tonic hier einen flachen Baum aus 6598 Elementen. Da Tonic allerdings stets das Ergebnis der letzten Anweisung zeigt, kann man die Anzeige folgendermaßen beschleunigen:

```
let data = await request(haltestellenUrl);
let json = await convertCSV(data);
_.take(10, json)
```

Auf diese Weise wird nach wie vor die Variable `json` korrekt befüllt, aber die Anzeige ihres Inhalts ist auf die ersten 10 Elemente beschränkt.

Mit Geo-Daten arbeiten

Ein Blick in die konvertierten Daten zeigt, dass über die beiden Eigenschaften `BREITE` und `LAENGE` Geo-Daten repräsentiert sind. Mit einer Hilfsfunktion `geo` kann man diese Daten in eine Form umwandeln, die Tonics intelligenter Werte-Betrachter erkennt und als Kartenansicht darstellt (**Bild 6**):



Kartendarstellung: Tonics intelligenter Werte-Betrachter stellt die Daten in einer Kartenansicht dar (**Bild 6**)

```
function geo (data) {
  return {
    latitude: data.BREITE,
    longitude: data.LAENGE,
    title: data.NAME
  }
}

let location = _.find(x=>_.startsWith(
  "Berlin", x.NAME), json)
geo(location)
```

Man kann auch eine ganze Liste von Orten so darstellen. Der folgende Code zeigt die 20 nächsten Haltestellen um den in `location` gespeicherten Ort:

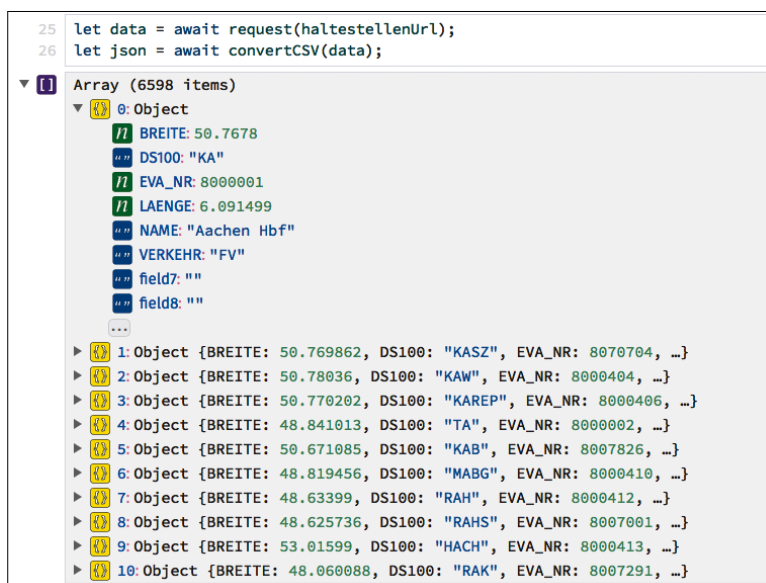
```
let distance = _.curry((a,b) => {
  let dlat = b.latitude - a.latitude;
  let dlong = b.longitude - a.longitude;
  return Math.sqrt(dlat*dlat + dlong*dlong);
})
```

Die Funktion `distance` berechnet die Distanz zwischen zwei übergebenen Geo-Punkten. Mit der Funktion `_.curry` wird sie auch mit nur einem Parameter aufrufbar gemacht. Das Ergebnis ist dann eine Funktion mit dem zweiten Parameter.

Damit lässt sich eine Funktion erzeugen, mit der man die Distanz zu einem gegebenen Ort berechnen kann:

```
let dist_to_location = distance(location);
dist_to_location(geo(json[0]))
```

Auf diese Weise lässt sich das Array aus Haltestellen nach der Distanz zu `location` sortieren: ►



Objektdaten: Hier werden die Objektdaten dargestellt (**Bild 5**)

```
let sortedGeo = _.compose(_.sortBy(
  distance(geo(location))),
  _.map(geo));
_.take(20, sortedGeo(json))
```

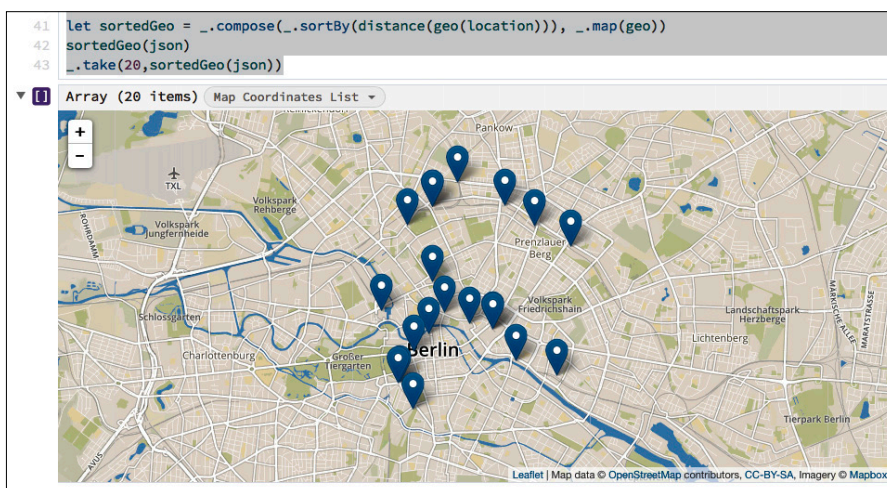
Mit *sortedGeo* wird eine Funktion definiert, die zuerst auf ihr Argument die Funktion *_.map* mit der Abbildungsfunktion *geo* anwendet. Auf das solcherart in Geo-Punkte konvertierte Array wird dann eine Funktion angewendet, welche die Punkte mit *sortBy* nach Distanz zu *location* sortiert. Mit *_.take(20, sortedGeo(json))* werden dann die ersten 20 Elemente aus dem sortierten Array entnommen. **Bild 7** zeigt die Kartendarstellung mit den so gesammelten Haltestellen.

Endpunkte definieren

Ein noch sehr neues, aber ebenso nützliches wie mächtiges Feature ist die Möglichkeit, in einem Tonic-Notizbuch HTTP-Endpunkte zu definieren. Die einfachste Möglichkeit sieht so aus:

```
exports.tonicEndpoint = (req, res) => {
  res.end("Hello, World!");
}
```

Man definiert also einfach auf dem *exports*-Objekt des Notizbuchs eine Funktion mit dem Namen *tonicEndpoint*. Diese Funktion wird bei HTTP-Zugriffen auf den Endpunkt-URL des Notizbuchs mit dem Request-Objekt *req* und dem Response-Objekt *res* aufgerufen. Der zugehörigen Endpunkt-



Orte: Darstellung einer Liste von Orten (Bild 7)

URL kann man ermitteln, indem man im Kopf des Notizbuchs auf den Link *endpoint* klickt. Eine weitere Möglichkeit ist der Einsatz des Node.js-Frameworks *express*. Allerdings ist dazu ein wenig Vorarbeit notwendig:

```
let notebook = require('notebook');
let createApp = notebook
("tonic/express-endpoint/1.0.0");
let app = createApp(exports);
```

Das NPM-Modul *notebook* wird von Tonic selbst bereitgestellt und ermöglicht es, die Inhalte anderer Notizbücher zu laden. Jedes Notizbuch verhält sich dabei wie ein eigenständiges Node.js-Modul – es kann mit *require* Module laden und es exportiert Objekte und Funktionen über sein *exports*-Objekt.

Das Notizbuch *tonic/express-endpoint/1.0.0* enthält Code, der einen Tonic-Endpunkt so mit einer Express-App verknüpft, dass alle eingehenden HTTP-Anfragen an Express weitergeleitet werden.

Die Implementierung dieser Verknüpfung ist natürlich – ebenso wie alle Tonic-Notizbücher – öffentlich einsehbar. Dazu muss man einfach nur den URL <https://tonicdev.com/tonic/express-endpoint/1.0.0> in einem Browserfenster öffnen. Der zugehörige Code ist auch nur eine knappe Bildschirmseite lang.

Nachdem man eine solche mit einem *tonicEndpoint* verknüpfte Express-App hat, kann man Express ansonsten wie gewohnt benutzen:

```
let express = require('express')
let apiRouter = express.Router();
apiRouter.get("/haltestellen", (req,res) => {
  let n = parseInt(req.query.n) || 10;
  res.json(_.take(n,json)); }
app.use("/api", apiRouter);
```

Damit steht unter dem Notizbuch-Endpunkt-URL unter der Erweiterung der Route */api/haltestellen* ein Endpunkt zur

Listing 1: Beispielprogramm

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tonic</title>
    <script src="https://embed.tonicdev.com"
      data-element-id="code-beispiel-1">
    </script>
  </head>
  <body>
    <div id="code-beispiel-1">
      function fac (n) {
        return n==0? 1 : n*fac(n-1);
      }
      fac(5)
    </div>
  </body>
</html>
```



```
function fac (n) {
  return n===0? 1 : n*fac(n-1);
}

fac(5)
```

Powered by Tonic help ▶ run

```
function fib (n) {
  return n < 1 ? 0
    : n <= 2 ? 1
    : fib(n-1) + fib(n-2);
}

fib(7)
```

Powered by Tonic help ▶ run

Widgets: Tonic als eingebettete Widgets (Bild 8)

Verfügung, mit dem die Haltestellen-Daten als JSON abgerufen werden können.

Tonic-Notizbücher sind grundsätzlich öffentlich sichtbar. Selbst ein nicht per Publish veröffentlichtes Notizbuch ist sichtbar, wenn man dessen URL mit der eindeutigen ID kennt. Allerdings kann man nur Notizbücher verändern, die man selbst erstellt hat. Dennoch ist offensichtlich, dass dieser Dienst insbesondere zum öffentlichen Austausch von Quellcodebeispielen für Node.js gedacht ist. Dazu bietet Tonic auch eine weitere Möglichkeit: Man kann es auf der eigenen Website einbinden. Um ein bestehendes HTML-Element in ein Tonic-Element umzuwandeln, kann man das folgende Skriptelement in der Website einbinden:

```
<script src=https://embed.tonicdev.com data-element-id=
"mein-element"></script>
```

Mit dem Attribut *data-element-id* wird die ID eines Elements auf der Seite referenziert. Dies kann beispielsweise wie in [Listing 1](#) aussehen. [Bild 8](#) zeigt, wie das eingebettete Tonic-Widget aussieht. Ein Nachteil dieser Lösung ist, dass man so für jedes ersetzte Element eine eindeutige ID festlegen muss. Es wäre oft angenehmer, wenn man einfach alle Bereiche mit einer Klasse auszeichnen könnte. Das lässt sich durch das JavaScript-API lösen ([Listing 2](#)).

Fazit

Tonic beeindruckt mit einigen Features wie der intelligenten Darstellung von Ergebniswerten, der direkten Verfügbarkeit nahezu aller NPM-Pakete oder der Möglichkeit, mit Endpunkten in Notizbüchern schnell Microservices zu implementieren. Allerdings gibt es auch ein paar Schattenseiten: Die Seite ist manchmal sehr langsam. Die Oberfläche sieht gut aus und ist weitgehend gut durchdacht, bietet aber noch viele Möglichkeiten, Details zu verbessern. Ab und zu verhält sich Code seltsam, und mangels guter Debugging-Werkzeuge ist das, was wirklich passiert, dann schwierig nachzuvollziehen.

Durch die sehr gut gelösten Einbettungsmöglichkeiten ist Tonic tatsächlich ein nützlicher Dienst für Node.js-spezifische Code-Beispiele und Dokumentation. Mittlerweile hat sogar die offizielle NPM-Seite zu jedem Paket einen Verweis auf

Listing 2: Lösung mit JavaScript-API

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tonic</title>
    <script src="https://embed.tonicdev.com">
    </script>
  </head>
  <body>
    <div class="tonic">
      function fac (n) {
        return n===0? 1 : n*fac(n-1);
      }
      fac(5)
    </div>
    <div class="tonic">
      function fib (n) {
        return n < 1 ? 0
          : n <= 2 ? 1
          : fib(n-1) + fib(n-2);
      }
      fib(7)
    </div>

    <script>
      function createNotebook (el) {
        var code = el.textContent;
        el.innerHTML = "";
        Tonic.createNotebook({
          element: el,
          source: code
        });
      }
      [].forEach.call(document.querySelectorAll(
        ".tonic"), createNotebook);
    </script>
  </body>
</html>
```

Tonic. So kann man direkt von der Beschreibung eines NPM-Pakets zu einem Tonic-Notizbuch springen, in dem die Bibliothek geladen wird. ■



Jochen H. Schmidt

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

MODULARISIERUNG IN JAVASCRIPT

Modularer Quelltext

Wie man in JavaScript Quelltext in Module zusammenfassen kann.

Gerade in großen Projekten ist es ratsam, den Quelltext in gewisser Weise zu strukturieren und in einzelne Module zusammenzufassen. Einige Sprachen wie Python unterstützen Module bereits nativ. In JavaScript dagegen fehlte ein solcher Support bis vor Kurzem. Erst mit Version ES6 (beziehungsweise ES2015) sind Module ein Bestandteil des Standards und damit der Sprache.

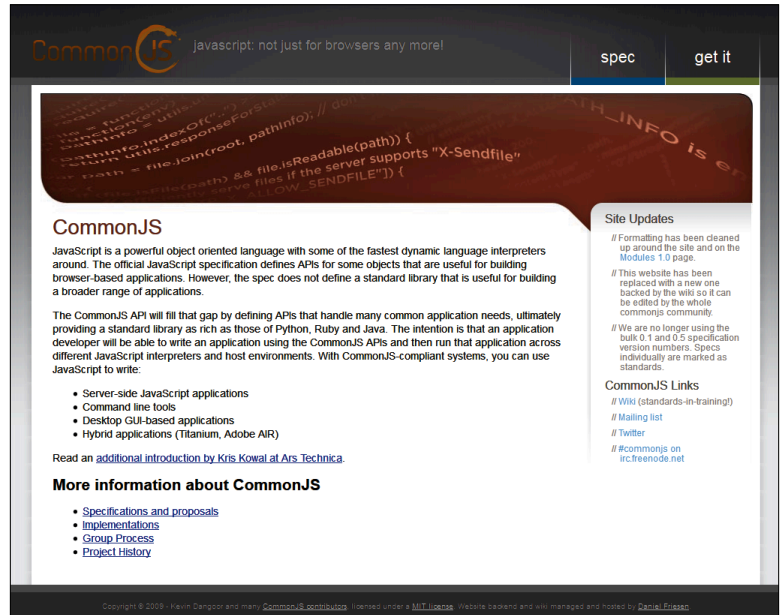
Da dies aber relativ lange eben nicht der Fall war, haben sich im Lauf der Zeit verschiedene Entwurfsmuster und Techniken herausgebildet, um Quelltext modular aufzubauen und in wiederverwendbare Module zusammenzufassen.

Im Zuge dessen haben sich darüber hinaus zwei Quasi-Standards entwickelt, die ebenfalls die Modularisierung des Quelltexts als Ziel haben. Zum einen das sogenannte CommonJS, das in erster Linie Module auf Serverseite adressiert, beispielsweise für die Verwendung unter Node.js (Bild 1). Zum anderen das sogenannte AMD (für Asynchronous Module Definition), das dem Erstellen von Modulen auf Clientseite dient, sprich von eben solchen Modulen, die im Browser verwendet werden.

Im Folgenden möchte ich Ihnen diese verschiedenen Konzepte und Techniken, aber auch den neuen nativen Modulsupport vorstellen.

Das klassische Module-Entwurfsmuster

Doch alles der Reihe nach. Lange Zeit musste man sich als JavaScript-Entwickler nämlich selbst behelfen, wenn man seinen Quelltext in irgendeiner Art strukturieren wollte. Das Emulieren von Modulen basiert dabei auf der Verwendung



CommonJS adressiert in erster Linie Module auf Serverseite (Bild 1)

des Namespace-Entwurfsmusters sowie der Kapselung von Daten über Closures und IIFEs (Immediately-Invoked Function Expressions).

Die Idee dabei ist folgende: Mit Hilfe einer IIFE wird ein eigener Sichtbarkeitsbereich geschaffen, der dafür sorgt, dass alle Eigenschaften und Funktionen des Moduls von außen nicht zugänglich sind. Die IIFE gibt zudem ein Objekt zurück, welches das öffentliche API darstellt und Methoden und Eigenschaften enthält, die durch das Modul bereitgestellt werden (Exportobjekt). Die Methoden dieses Objekts haben dabei Zugriff auf alle Eigenschaften und Funktionen des Moduls.

Das Listing 1 zeigt ein einfaches Beispiel für diese Technik. Gezeigt ist hier das Modul *Module*, das zwei Variablen *x* und *y* enthält, die nur innerhalb des Moduls sichtbar sind. Die IIFE gibt ein Exportobjekt zurück, das wiederum die Methoden *getX()* und *getY()* enthält, zwei einfache Getter-Methoden für die beiden zuvor genannten (privaten) Variablen.

Von außerhalb des Moduls lässt sich nun nur auf die Methoden des Exportobjekts zugreifen, sprich im

Listing 1: Modul-Entwurfsmuster als Kombination aus Closure und IIFE

```
let Module = Module || (function (function (getY: function() {
() {                                     return y;
// private Variable                       }
let x = 5;                                }
let y = 4711;                             }));
// öffentliches API                       console.log(Module.getX()); // 5
return {                                  console.log(Module.getY()); // 4711
// öffentliche Funktion
getX: function() {                       Module.x = 888;
return x;                                Module.y = 888;
},                                         console.log(Module.getX()); // 5
                                           console.log(Module.getY()); // 4711
```

vorliegenden Fall auf die Methoden `getX()` und `getY()`. Versucht man, wie im [Listing 1](#) zu sehen ist, die privaten Eigenschaften (sprich die Variablen `x` und `y`) zu überschreiben, so hat dies keine Auswirkungen (tatsächlich handelt es sich bei `x` und `y` um keine Eigenschaften von `Module`, bei `Module.x` und `Module.y` dagegen schon).

Kleines Detail am Rande: Die *ODER*-Verknüpfung zu Beginn des Moduls verhindert, dass ein Modul mehrfach initialisiert wird. Falls die Variable `Module` bereits existiert, wird diese auch verwendet, nur falls nicht, wird die IIFE aufgerufen und das Modul einmalig initialisiert.

Das Revealing-Module-Entwurfsmuster

Etwas unschön an der gezeigten Implementierung ist noch, dass man – wenn man innerhalb einer Methode eines Moduls andere Methoden des gleichen Moduls aufrufen möchte – diesen Methodenaufrufen immer den Namen des Moduls voranstellen muss.

In [Listing 2](#) beispielsweise wurde das Modul um eine Methode `print()` erweitert, welche ihrerseits die beiden Methoden `getX()` und `getY()` aufruft. Beide Aufrufe funktionieren nur, wenn sie auf dem Objekt `Module` aufgerufen werden, sprich wenn man `Module.getX()` und `Module.getY()` schreibt.

Um diese Unbequemlichkeit zu umgehen, bedient man sich gerne einer Technik, welche im Allgemeinen als Revealing Module beziehungsweise Revealing Module Pattern bekannt ist.

Die Idee dabei ist, dass eine Methode, die exportiert werden soll, nicht innerhalb des Exportobjekts selbst definiert wird, sondern außerhalb davon. Innerhalb des Exportobjekts referenziert man lediglich die entsprechende Methode.

Listing 2: Klassisches Modul-Entwurfsmuster

```
let Module = Module || (function () {
  let x = 5;
  let y = 4711;
  return {
    getX: function() {
      return x;
    },
    getY: function() {
      return y;
    },
    // Zugriff auf eigene Methoden
    print: function() {
      console.log(
        'x: ' +
        Module.getX() +
        ', y: ' +
        Module.getY()
      )
    }
  }
})();
```

Listing 3: Revealing-Module-Entwurfsmuster

```
let Module = Module || (function () {
  let x = 5;
  let y = 4711;
  function getX() {
    return x;
  }
  function getY() {
    return y;
  }
  function print() {
    console.log(
      'x: ' +
      getX() +
      ', y: ' +
      getY()
    )
  }
  // Rückgabeobjekt enthält nur Verweise
  return {
    getX: getX,
    getY: getY,
    print: print
  }
})();
Module.print();
```

[Listing 3](#) zeigt, wie man dank dieser Technik die Methoden `getX()` und `getY()` direkt innerhalb der Methode `print()` aufrufen kann, ohne dabei den Modulnamen voranzustellen.

Importieren von Modulen

Um ein Modul innerhalb eines anderen Moduls zu importieren, übergibt man das zu importierende Modul dem anderen einfach als Argument der IIFE. In [Listing 4](#) beispielsweise sind zwei Module zu sehen: das von eben bereits bekannte Modul `Module` (jetzt mit Namen `ModuleA`) sowie das neue Modul `ModuleB`. Dadurch, dass Letzteres dem anderen Modul als Argument übergeben wird, kann innerhalb der Methode `print()` auf die Methode `printHelloWorld()` von `ModuleB` zugegriffen werden. Möchte man mehrere Module importieren, lässt sich das Ganze natürlich fortsetzen und es können einfach entsprechend mehrere Argumente übergeben werden.

Module Augmentation

Möchte man den Quelltext für ein Modul auf mehrere Quelltextdateien verteilen, kann man sich eines kleinen Tricks bedienen, einer Technik, die als Module Augmentation bezeichnet wird. Und zwar übergibt man dazu einfach das Modul selbst der IIFE als Argument, fügt Eigenschaften und Funktionen hinzu und verwendet das Modul schließlich selbst als Rückgabewert ([Listing 5](#)).

Module Augmentation gibt es in zwei Varianten: Loose Augmentation sowie Tight Augmentation. Bei der Loose ►

Listing 4: Importieren eines Moduls

```

let ModuleB = ModuleB || (function () {
  function printHelloWorld() {
    console.log('Hello World');
  }
  return {
    printHelloWorld: printHelloWorld
  }
})();

let ModuleA = ModuleA || (function (moduleB) {
  // Innerhalb des Moduls stehen ...
  // ... Abhängigkeiten über das entsprechende ...
  // ... Argument zur Verfügung.
  let x = 5;
  let y = 4711;
  function getX() {
    return x;
  }
  function getY() {
    return y;
  }
  function print() {
    moduleB.printHelloWorld();
    console.log(
      'x: ' +
      getX() +
      ', y: ' +
      getY()
    )
  }
  return {
    getX: getX,
    getY: getY,
    print: print
  }
  // Abhängigkeit wird als Argument übergeben
})(ModuleB);
ModuleA.print();

```

Augmentation wird – für den Fall, dass das Modul noch gar nicht erzeugt wurde, sprich quasi die erste Quelltextdatei geladen wurde – ein leeres Objekt als Argument übergeben. Bei der Tight Augmentation dagegen wird davon ausgegangen, dass es das Modul bereits gibt (sprich mindestens eine andere Quelltextdatei bereits geladen wurde). Zu sehen ist dies im Beispiel in dem Abschnitt unterhalb *Datei 2*: Hier wird nur *ModuleA* als Argument übergeben. Im Beispiel ergibt dies insofern Sinn, weil die Methode *print()* (die in der *Datei 2* definiert ist), von den Methoden *getX()* und *getY()* abhängt (die wiederum in *Datei 1* definiert sind).

Vorteil der Loose Augmentation ist, dass sich Skripts, die zum Modul beitragen, asynchron laden lassen. Es spielt keine Rolle, welches Skript als Erstes geladen wird, da im Zweifelsfall das Modul neu erstellt wird. Bei der Tight Augmentation ist das nicht der Fall. Hier setzt die Anwendung eine bestimmte Reihenfolge beim Laden voraus. Vorteil dabei ist, dass man sich innerhalb einer Datei darauf verlassen kann, dass andere Dateien bereits geladen wurden und entsprechend dort zum Modul hinzugefügte Eigenschaften und Methoden existieren (beispielsweise die Methoden *getX()* und *getY()*, die in der Methode *print()* verwendet werden).

Listing 5: Module Augmentation

```

// Datei 1
let ModuleA = ModuleA || (function (moduleA) {
  // Eigenschaften und Funktionen hinzufügen
  let x = 5;
  let y = 4711;
  moduleA.getX = function() {
    return x;
  }
  moduleA.getY = function() {
    return y;
  }
  // Modul selbst als Rückgabewert
  return moduleA;
  // Loose Augmentation
})(ModuleA || {});

// Datei 2

let ModuleA = (function (moduleA) {
  // Eigenschaften und Funktionen hinzufügen
  moduleA.print = function() {
    console.log(
      'x: ' +
      moduleA.getX() +
      ', y: ' +
      moduleA.getY()
    )
  }
  // Modul selbst als Rückgabewert
  return moduleA;
  // Tight Augmentation
})(ModuleA);

// Datei 3
ModuleA.print();

```

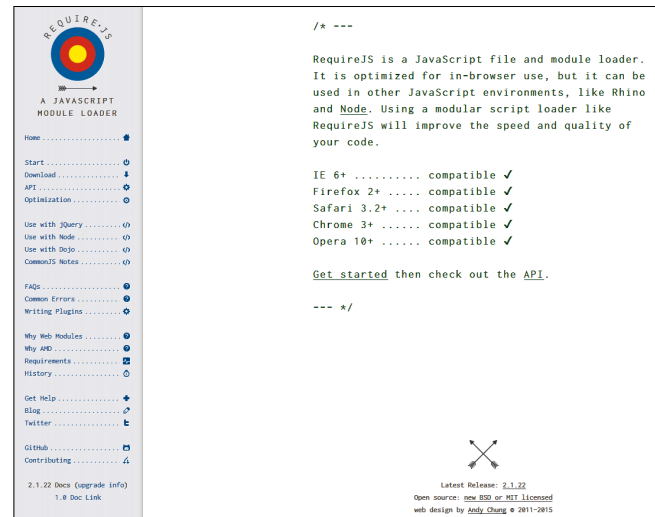

Unabhängig davon, welche Art der Module Augmentation man verwendet, besteht ein gemeinsamer Nachteil: Funktionen, die in einer Quelltextdatei für ein Modul definiert werden, haben keinen Zugriff auf private Variablen, die in einer anderen Quelltextdatei für das gleiche Modul definiert wurden (beispielsweise könnte innerhalb der Methode *print()* nicht auf die Variablen *x* und *y* direkt zugegriffen werden).

AMD ermöglicht das asynchrone Laden von Modulen und adressiert damit Anwendungen, die im Browser laufen. AMD wird beispielsweise von RequireJS (<http://requirejs.org>) implementiert (Bild 2). Listing 6 zeigt die Definition eines Moduls in AMD, wobei man bei genauerem Hinsehen das Module-Entwurfsmuster wiedererkennen kann. Für das Registrieren eines Moduls steht die Methode *define()* zur Verfügung, welche die Modulfunktion als Argument erwartet. Optional kann als erstes Argument der Name des Moduls definiert werden, sowie als zweites Argument eine Auflistung der Module, von denen das aktuelle Modul abhängig ist. Diese stehen dann innerhalb des Moduls als Argumente zur Verfügung.

Für das Einbinden von Modulen definiert AMD die Funktion *require()*. Wie bei *define()* werden die Namen der einzubindenden Module dabei in Form eines Arrays übergeben. Die entsprechenden Module stehen dann innerhalb des Codes zur Verfügung:

```
require(['ModuleA'], function(moduleA) {
    moduleA.print();
});
```

Im Gegensatz zu AMD richtet sich CommonJS (genauer gesagt: dessen Modulspezifikation (<http://wiki.commonjs.org/>



Asynchronous Module Definition (AMD) wird beispielsweise von RequireJS implementiert (Bild 2)

wiki/Modules/1.1) an die Verwendung von Modulen auf Serverseite (Bild 3). In Listing 7 sehen Sie die im Vergleich zu AMD etwas kompaktere Definition eines Moduls in CommonJS. Es fällt sofort auf, dass der Quelltext nicht wie bei AMD in einer eigenen IIFE gekapselt ist. CommonJS geht nämlich vom Prinzip her so vor, jede Quelltextdatei als ein eigenes Modul mit eigenem Gültigkeitsbereich zu betrachten. Innerhalb eines solchen Gültigkeitsbereichs stehen im Wesentlichen folgende drei verschiedene Komponenten zur Verfügung:

- Über die Methode *require()* ist es möglich, abhängige Module zu laden. Übergeben wird der Methode der Name ►

Listing 6: Moduldefinition mit AMD

```
define(
    'ModuleB',    // Names des Moduls
    [],          // Zu importierende Module
    function () { // liefert das Modul zurück

        let module = {
            printHelloWorld: function() {
                console.log('Hello World');
            }
        }
        return module;
    }
);

define(
    'ModuleA',    // Names des Moduls
    ['ModuleB'],  // Zu importierende Module
    function (moduleB) { // liefert das Modul zurück

        let x = 5;
        let y = 4711;

        let module = {
            getX: function() {
                return x;
            },
            getY: function() {
                return y;
            },
            print: function() {
                moduleB.printHelloWorld();
                console.log(
                    'x: ' +
                    this.getX() +
                    ', y: ' +
                    this.getY()
                )
            }
        }
        return module;
    }
);
```



Die Modulspezifikation von CommonJS richtet sich an die Verwendung von Modulen auf Serverseite (Bild 3)

des jeweiligen Moduls beziehungsweise dessen Pfad. Im Beispiel lädt der Aufruf

```
let moduleA = require('ModuleA')
```

die Datei *ModuleA.js* beziehungsweise das dort definierte Modul in die Variable *moduleA*.

- An dem Objekt *exports* werden die Komponenten hinzugefügt, die das Modul nach außen hin bereitstellen soll (dies entspricht in etwa dem Exportobjekt aus dem Module Entwurfsmuster). Im Beispiel werden in *ModuleA* die Methode *print()* und in *ModuleB* die Methode *printHelloWorld()* exportiert.
- Das Objekt *module* (im Listing nicht verwendet) stellt zusätzliche Metainformationen für ein Modul zur Verfügung, beispielsweise die Modul-ID oder den URL, unter dem ein Modul gefunden werden kann.

ES2015-Module

Seit ECMAScript 6 stellt der Standard ein natives Modulsystem für JavaScript zur Verfügung. Bei der Spezifikation der entsprechenden Modulkomponente des Standards wurde dabei ein Kompromiss zwischen AMD und CommonJS eingegangen (Bild 4).

So ist zum einen die Schreibweise ähnlich kompakt wie bei CommonJS (wenn nicht noch kompakter). Zum anderen gibt es die Möglichkeit, Module sowohl synchron (wie bei CommonJS) als auch asynchron (wie bei AMD) zu laden. Folglich können ES6-Module sowohl auf Serverseite als auch auf Clientseite zum Einsatz kommen.

Der ES6-Modul-Standard besteht dabei aus zwei Teilen: zum einen aus einer deklarativen Syntax für die Definition von Modulen (sprich den Export von Komponenten) sowie den Import

Listing 7: Moduldefinition mit CommonJS

```
// Datei ModuleA.js
let moduleB = require('ModuleB');
let x = 5;
let y = 4711;
function getX() {
    return x;
}
function getY() {
    return y;
}
function print() {
    moduleB.printHelloWorld();
    console.log(
        'x: ' +
        getX() +
        ', y: ' +
        getY()
    );
}
exports.print = print;

// Datei ModuleB.js
function printHelloWorld() {
    console.log('Hello World');
}
exports.printHelloWorld = printHelloWorld;

// Datei main.js
let moduleA = require('ModuleA');
moduleA.print();
```

von Modulen, zum anderen aus einem API, mit dem es beispielsweise möglich ist, Programmmodule bei Bedarf dynamisch zu laden.

Das Prinzip von ES6-Modulen ist ähnlich wie auch bei CommonJS: Ein Modul ist nichts anderes als eine gewöhnliche Skriptdatei, in der Objekte und Funktionen, die das Modul nach außen hin bereitstellen soll, exportiert werden. Dieser Export von Komponenten kann auf verschiedene Arten

Tabelle 1: Vergleich der verschiedenen Modulsysteme

	Syntax	Synchron	Asynchron	Client	Server
Module Entwurfsmuster	leicht komplex	Tight Augmentation	Loose Augmentation	ja	ja
CommonJS	kompakt	ja	nein	nein (teilweise mit Browserify)	ja
AMD	komplex	nein	ja	ja	nein
ES6-Module	kompakt	ja	ja	ja	ja

geschehen: Entweder gibt man die zu exportierenden Komponenten explizit an, dann spricht man von einem benannten Export (**Listing 8**), oder man gibt eine einzelne Default-Komponente an, die quasi den Hauptteil des Moduls repräsentiert (**Listing 9**).

Der Vorteil von Default-Exporten ist, dass Sie, wenn Sie das entsprechende Modul in ein anderes Modul importieren, nicht wissen müssen, wie die exportierte Komponente innerhalb des Moduls heißt.

Import-Optionen

Für den Import stehen Ihnen verschiedene Möglichkeiten zur Verfügung. Angenommen, das in **Listing 8** definierte Modul ist in der Datei *logger.js* gespeichert. Dann können einzelne Komponenten dieses Moduls direkt wie folgt unter Angabe des jeweiligen Komponentennamens importiert werden:

```
import {info, debug, warn, error} from 'logger'
```

Dabei werden die Komponenten an lokale Variablen im aktuellen Modul gebunden:

```
info(4711);
// Aufruf der Funktion info des Moduls 'logger'
```

Dies ist vergleichbar mit dem statischen Import von Methoden in Java. Ein Nachteil hierbei ist jedoch, dass schnell Na-

Listing 8: Inline-Export von Modulen

```
// Datei: logger.js
export function info(meldung) {
  console.log(meldung);
}
export function debug(meldung) {
  console.debug(meldung);
}
export function warn(meldung) {
  console.warn(meldung);
}
export function error(meldung) {
  console.warn(meldung);
}
```

Listing 9: Default-Exportkomponente

```
// Datei: Album.js
export default class Album {
  constructor(interpret, titel, jahr) {
    this.interpret = interpret;
    this.titel = titel;
    this.jahr = jahr;
  }
}
```

Folglich können Sie die importierten Komponenten unter neuem Namen ansprechen:

```
e(4711);
// Aufruf der Funktion error des Moduls 'logger'
```

Wie die Beispiele des Artikel gezeigt haben, gibt es in JavaScript verschiedene Lösungen zur Modularisierung von Quelltext (**Tabelle 1**).

Zum einen besteht die Möglichkeit, Module über das Module-Entwurfsmuster zu emulieren, zum anderen stehen mit den beiden Modulsystemen CommonJS und AMD sowohl Lösungen für die Serverseite als auch für die Clientseite zur Verfügung. Native Module dagegen sind erst seit ES6 beziehungsweise ES2015 im Sprachumfang enthalten. ■



ECMAScript 6 stellt ein natives Modulsystem für JavaScript zur Verfügung (**Bild 4**)

meniskonflikte auftreten können. Nämlich genau dann, wenn gleichnamige Komponenten aus verschiedenen Modulen importiert werden.

Abhilfe schaffen hier zwei andere Import-Optionen: So können einzelne Komponenten eines Moduls unter einem anderen Namen (Alias) importiert werden, oder direkt das gesamte Modul:

```
import {
  info as i, debug as d, warn as w, error as e
} from 'logger'
```



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>

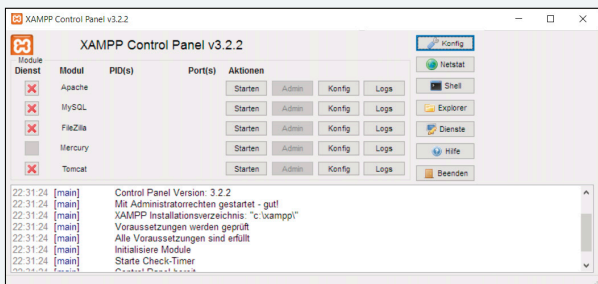
ÜBERBLICK

CD-Highlights

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

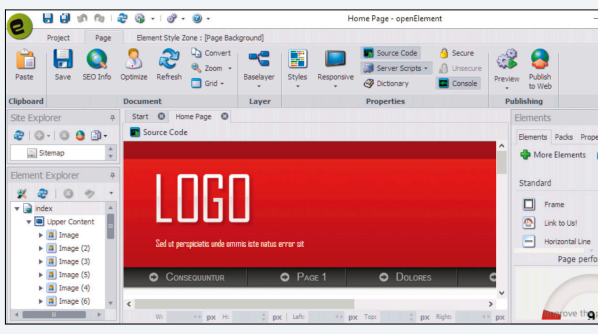
XAMPP 7.0.4 für Windows

So mancher wird schon die Erfahrung gemacht haben: Ein Apache-Webserver installiert sich nicht so leicht, Systemkenntnisse zum Einspielen der Pakete sind erforderlich. Noch schwieriger wird es, wenn weitere Pakete wie eine Datenbank oder Skriptsprachen dazukommen. Die Lösung ist XAMPP – eine kostenlose, leicht zu installierende Apache-Distribution, die unter anderem MariaDB, PHP und Perl enthält. XAMPP enthält zusätzlich Werkzeuge wie den FTP-Server FileZilla, den Mailserver Mercury, phpMyAdmin, Webalizer und OpenSSL. Zur Bequemlichkeit der Entwickler ist XAMPP nach der Installation so konfiguriert, dass alle Funktionen eingeschaltet sind.



openElement 1.53 R6

openElement ist ein leistungsstarkes und intuitiv zu nutzendes Web-Authoring-Programm für kreative Profis. Es unterstützt Sie mit allen Tools für eine standardkonforme Website. Die Software unterstützt neueste Technologien wie HTML5, CSS3 und jQuery. Sie können professionell gestaltete Vorlagen nutzen oder Projekte von Grund auf neu aufbauen. Der von openElement generierte Code ist zu 100 Prozent W3C-konform, SEO-ready und zeigt Webseiten korrekt mit jedem Browser.



Weitere Highlights:

BlueGriffon 1.8

Der Wysiwyg-Editor hält sich an die W3C-Webstandards und erstellt zu HTML 4, HTML 5, XHTML 1, XHTML 1.1 und XHTML 5 kompatible Seiten. BlueGriffon unterstützt CSS 2.1 sowie 3 und basiert auf Techniken, die auch Firefox nutzt, beispielsweise die HTML-Rendering-Engine Gecko.

Notepad++ 6.9

Der Editor Notepad++ hat deutlich mehr zu bieten als das Windows-eigene Notepad. Wollen Sie in Notepad++ etwa zwei Dokumente gleichzeitig editieren, lassen sich beide Texte in einer Split-Screen-Ansicht darstellen. Beim Editieren von HTML-Seiten oder Quelltexten bietet Notepad++ eine Syntax-Hervorhebung mit Code-Collapsing.

PhraseExpress 11.0.132

Die Textbausteinverwaltung erspart die Eingabe häufig benötigter Texte und beschleunigt damit die Schreibarbeit. Textbausteine lassen sich inklusive Word-, HTML- und RTF-Schriftformatierung und Bildern in einer Ordnerstruktur speichern und per Autotext oder Hotkey einfügen. Der Zwischenablagenspeicher sichert die vorherigen Inhalte der Zwischenablage, bevor sie von neuen Inhalten überschrieben werden, und bietet Zugriff auf die Historie.

Synwrite 6.20.2195

Ein vielseitiger Quelltext-Editor, der sich vor allem an Software-Entwickler und Webdesigner richtet. Synwrite unterstützt das Arbeiten im Quelltext durch das automatische Hervorheben der Syntax von über 80 Sprachen von ADA bis XML. Der Übersicht dient neben der üblichen Baumstruktur eine Minimap; für wiederkehrende Aufgaben gibt es einen Makrorekorder. Dazu gibt es einen integrierten FTP-Client, eine Codevorschau und eine Rechtschreibprüfung.

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

CONSTRAINT VALIDATION API

Native Validierung


Das Constraint Validation API vereinfacht die Validierung von Formulareingaben.

Noch vor nicht allzu langer Zeit war die Validierung von Formularfeldern, also der Check auf die Korrektheit der User-Eingaben, nur mit JavaScript möglich. Mittlerweile bietet HTML schon von Haus aus gewisse Möglichkeiten zur Überprüfung von Formularfeldern an, was mittlerweile auch von allen gängigen Browsern unterstützt wird (Bild 1).

In Listing 1 sehen Sie hierzu als Beispiel ein einfaches Registrierungsformular, das entsprechende HTML-Attribute verwendet, um die native Validierung anzustoßen. Die Felder zur Eingabe von Nutzernamen und E-Mail-Adresse sind beispielsweise über das Attribut *required* als Pflichtfelder gekennzeichnet, Letztere durch das *type*-Attribut (mit Wert *email*) zusätzlich als Feld, in das nur E-Mail-Adressen eingegeben werden können.

Das Feld zur Eingabe der Homepage erlaubt nur die Eingabe von URLs (gekennzeichnet durch den Wert *url* im *type*-Attribut) und das Feld zur Eingabe des Alters nur die Eingabe von Zahlenwerten (Wert *number*) in einem bestimmten Wertebereich, der durch die beiden Attribute *min* und *max* definiert wurde.

Ohne eine einzige Zeile JavaScript haben Sie für Standardfälle wie diese bereits eine funktionierende Validierung. Lässt man nun beispielsweise das Feld zur Eingabe der

Form validation  - LS Global 75.98% + 19.7% = 95.68%

Method of setting required fields and field types without requiring JavaScript. This includes preventing forms from being submitted when appropriate, the `checkValidity()` method as well as support for the `:invalid`, `:valid`, and `:required` CSS pseudo-classes.

Current aligned Usage relative Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8			45					4.3	
9			46					4.4	
10		43	47			8.4		4.4.4	
11	13	44	48	9	34	9.2	8	47	47
	14	45	49	9.1	35	9.3			
		46	50		36				
		47	51						

Das Constraint Validation API wird von den meisten Browsern unterstützt (Bild 1)

E-Mail-Adresse leer und versucht, das Formular abzuschicken, erzeugt der Browser eine entsprechende Fehlermeldung (die allerdings sehr allgemein gehalten ist) und wendet die CSS-Regeln an, die für die CSS-Pseudoklasse *:invalid* definiert wurden (Listing 2, Bild 2, Bild 3).

Für Formularelemente mit gültigen Werten dagegen werden entsprechend die CSS-Regeln für die CSS-Pseudoklasse *:valid* angewendet (im Beispiel das Feld zur Eingabe des Nutzernamens).

In Bild 4 sehen Sie zum Vergleich noch das Formular mit gültiger E-Mail-Adresse. Würde man nun die Submit-Schaltfläche betätigen, würde die Validierung keinen Fehler melden und das Formular erfolgreich abgeschickt.

Listing 1: Beispiel für die native HTML5-Validierung

```
<form id="register" name="register" method="post"
action="register.html">
  <div class="form-group">
    <label for="username">Nutzername:</label>
    <input type="text" id="username" name="username"
      required class="form-control" >
  </div>
  <div class="form-group">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email"
      required class="form-control" >
  </div>
  <div class="form-group">
```

```
<label for="url">Website:</label>
    <input type="url" id="website" name="website"
      class="form-control" >
  </div>
  <div class="form-group">
    <label for="age">Alter:</label>
    <input type="number" id="age" name="age" min="18"
      max="99" value="18" class="form-control" >
  </div>
  <button type="submit" id="submit" name="submit"
    class="btn btn-primary btn-block">Anmelden
  </button>
</form>
```

Tabelle 1: Native Validierung

Eigenschaft	Beschreibung
<code>willValidate</code>	Liefert die Angabe darüber, ob die Validierung für ein Formularelement aktiviert werden soll oder nicht. Standardmäßig ist diese Eigenschaft auf <i>true</i> gesetzt. Wird dagegen ein Formularelement über das <i>disabled</i> -Attribut deaktiviert, enthält diese Eigenschaft den Wert <i>false</i> . Für Nicht-Formularelemente wie beispielsweise <code><div></code> -Elemente liefert die Eigenschaft den Wert <i>undefined</i> .
<code>validity</code>	Enthält ein Objekt vom Typ <i>ValidityState</i> , welches wiederum detaillierte Informationen über die Gültigkeit der in das entsprechende Formularelement eingegebenen Daten enthält. Welche dies sind, sehen Sie in Tabelle 2.
<code>validationMessage</code>	Enthält die Validierungsmeldung, die der Browser anzeigt, für den Fall, dass die eingegebenen Daten ungültig sind. Je nach Browser unterscheidet sich diese Meldung. Beispielsweise liefern Chrome und Opera, wenn ein Pflichtfeld nicht ausgefüllt wurde, den Wert <i>Füllen Sie dieses Feld aus</i> , Firefox (schon etwas freundlicher) den Wert <i>Bitte füllen Sie dieses Feld aus</i> , und Safari schlicht (und weniger freundlich) den Wert <i>Wert fehlt</i> .

Listing 2: CSS-Pseudoklassen

```
input:required:invalid, input:focus:invalid {
    border: thick solid red;
}
input:required:valid {
    border: thick solid lightgreen;
}
```

Allerdings ist die Validierung ohne JavaScript nicht sonderlich flexibel. Möchten Sie beispielsweise überprüfen, ob der eingegebene Nutzernamen schon vergeben ist, können Sie

dies mit HTML alleine nicht erreichen. Wahrscheinlich wäre hierzu eine Serveranfrage notwendig, die Sie nur mit Hilfe von JavaScript (beziehungsweise Ajax) durchführen könnten. Auch um Beziehungen zwischen einzelnen Formularelementen zu validieren, zum Beispiel, ob zwei Eingabefelder den gleichen Wert enthalten, reicht HTML alleine nicht aus. Und bezüglich der Fehlermeldungen und deren Aussehen ist man mit reinem HTML ebenfalls stark eingeschränkt.

Doch es gibt Abhilfe: Dank des sogenannten Constraint Validation API (<https://html.spec.whatwg.org/multipage/forms.html#the-constraint-validation-api>) lässt sich die native Validierung von HTML5 nämlich auch per JavaScript steuern beziehungsweise auf Validierungsinformationen zugreifen und dabei deutlich flexibler validieren. Dazu definiert das Constraint Validation API verschiedene Eigenschaften und Methoden, die für Formulare und Formularelemente zur Verfügung stehen (Tabelle 1).

Native HTML5-Validierung in JavaScript

So gibt die Eigenschaft `willValidate` Auskunft darüber, ob die Validierung für ein Formularelement aktiviert ist oder nicht, die Eigenschaft `validationMessage` enthält im Fall eines Validierungsfehlers eine entsprechende Fehlermeldung, und über die Eigenschaft `validity` lassen sich Rückschlüsse auf die Fehlerursache ziehen. Hinterlegt ist dieser Eigenschaft ein Objekt mit verschiedenen Eigenschaften, die jeweils boolesche Werte enthalten (Tabelle 2), beispielsweise `valueMissing` für fehlende Eingaben, `tooShort` oder `tooLong` für Werte, welche die Mindestlänge unterschreiten oder überschreiten, oder `rangeUnderflow` und `rangeOverflow` für Werte, die außerhalb eines definierten Wertebereichs liegen.

Ein Beispiel für die Verwendung der nativen Validierung in JavaScript sehen Sie in Listing 3. Für das Textfeld zur Eingabe der E-Mail-Adresse wurde hier ein Event Listener für das `change`-Event registriert, innerhalb dessen dann einige der oben genannten Eigenschaften ausgegeben werden.

Das Beispiel lässt sich etwas praxisnäher gestalten, um beispielsweise statt der Standardausgabe des Browsers Fehlermeldungen unterhalb des jeweiligen Eingabefeldes anzuzeigen (Bild 5). Listing 4 zeigt dazu den entsprechenden JavaScript-Code. Abhängig davon, ob das Eingabefeld einen ►

Native Validierung in HTML5: Solche Meldungen erscheinen (Bild 2)

Je nach Fehlerursache gibt es unterschiedliche Meldungen für ein Formularfeld (Bild 3)

Das Formular mit gültigen Werten präsentiert sich ohne Meldungen (Bild 4)

Listing 3: Ausgabe der Validierungsdaten

```
function init() {
  let emailElement = document.
    getElementById('email');
  emailElement.addEventListener('change',
    validateEmail);
}

function validateEmail(e) {
  let event = (e ? e : window.event);
  // Event
  let emailElement = (event.target
    // Zielelement
    ? event.target
    : event.srcElement);
  console.log(emailElement.willValidate);
  // true
  console.log(emailElement.validity);
  // ValidityState: ...
  console.log(emailElement.validity.valueMissing);
  // ... Wert vorhanden?
  console.log(emailElement.validity.valid);
  // ... Wert gültig?
  console.log(emailElement.validationMessage);
  // Validierungsmeldung
}

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

gültigen Wert enthält (zu erkennen an der Eigenschaft *emailElement.validity.valid*), wird hier die Validierungsmeldung im entsprechenden Containerelement angezeigt oder nicht.

Neben den genannten Eigenschaften verfügen Formulare und Formularelemente dank des Constraint Validation API

Listing 4: Anzeigen der Validierungsmeldung

```
function validateEmail(e) {
  ...

  let errorContainer = document.getElementById(
    'email-error');
  let messageContainer =
    errorContainer.querySelector('.error-message');
  if(!emailElement.validity.valid) {
    messageContainer.textContent =
      emailElement.validationMessage;
    errorContainer.style.display = 'block';
  } else {
    messageContainer.textContent = '';
    errorContainer.style.display = 'none';
  }
}
```

auch über einige neue Methoden (Tabelle 3). Über die Methode *checkValidity()* kann beispielsweise überprüft werden, ob ein einzelnes Formularelement oder ein ganzes Formular gültig ist. Folglich kann diese Methode auf beidem aufgerufen werden. Die Methode *reportValidity()* sorgt dafür, dass zusätzlich auch im Fall eines Fehlers die Validierungsmeldung (im Standard-Look-and-Feel des jeweiligen Browsers) angezeigt wird, und die Methode *setCustomValidity()* schließlich ermöglicht darüber hinaus, eigene Validierungsmeldungen zu definieren.

Erweitertes Formular

Ein praxisnahes Beispiel zur Verwendung von *setCustomValidity()* sehen Sie in Listing 5. Das zu validierende Formular wurde dabei um ein zweites Textfeld erweitert, in dem Nutzer ihre E-Mail-Adresse ein zweites Mal eingeben sollen,

Nutzername: Max Mustermann

Email: max.mustermann

Website: http://www.javascripthandbuch.de

Alter: 44

Anmelden

Anstelle der Standardausgabe für die Validierungsmeldung wird diese unterhalb des Eingabefelds angezeigt (Bild 5)

Nutzername: Max Mustermann

Email: max.mustermann@javascripthandbuch.de

Email Kontrolle: max.mustermann@philipackermann.de

Website: http://www.javascripthandbuch.de

Alter: 44

Anmelden

Beispiel einer Validierung, welche mit HTML alleine nicht umgesetzt werden kann (Bild 6)

Nutzername: Max Mustermann

Email: max.mustermann@javascripthandbuch.de

Email Kontrolle: max.mustermann@javascripthandbuch.de

Website: http://www.javascripthandbuch.de

Alter: 44

Anmelden

Nur wenn beide E-Mail-Felder den gleichen Wert enthalten, ist die Validierung erfolgreich (Bild 7)

Listing 5: Überprüfen, ob zwei Eingabefelder den gleichen Wert enthalten

```

function init() {
    let emailElement = document.getElementById('email');
    let emailElement2 =
        document.getElementById('email2');
    emailElement.addEventListener('change',
        validateEmail);
    emailElement2.addEventListener('change',
        validateEmail);
}

function validateEmail(e) {
    let emailElement = document.getElementById('email');
    let emailElement2 =
        document.getElementById('email2');
    if (emailElement.value !== emailElement2.value) {
        emailElement.setCustomValidity('E-Mails müssen
            übereinstimmen. ');
        emailElement2.setCustomValidity('E-Mails müssen
            übereinstimmen. ');
    } else {
        emailElement.setCustomValidity('');
        emailElement2.setCustomValidity('');
    }

    let errorContainer = document.getElementById
        ('email-error');
    let messageContainer =
        errorContainer.querySelector('.error-message');

    if(!emailElement.validity.valid) {
        messageContainer.textContent =
            emailElement.validationMessage;
        errorContainer.style.display = 'block';
    } else {
        messageContainer.textContent = '';
        errorContainer.style.display = 'none';
    }

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

```

um der Eingabe einer zwar formal gültigen, aber falschen E-Mail-Adresse entgegenzuwirken. Die Funktion *validateEmail()* wird an beiden Eingabefeldern als Event Listener für das *change*-Event registriert und prüft, ob der Inhalt beider Eingabefelder gleich ist.

Ist dies nicht der Fall, wird über *setCustomValidity()* eine entsprechende Fehlermeldung gesetzt (ruft man diese Methode dagegen mit einer leeren Zeichenkette auf, sagt man damit, dass die Validierung erfolgreich war). Das Ergebnis

des Programms sehen Sie in **Bild 6** (Validierung fehlgeschlagen) und **Bild 7** (Validierung erfolgreich).

Möchte man die native Validierung mit JavaScript nicht wie bisher gezeigt direkt bei Änderungen am jeweiligen Eingabefeld anstoßen, sondern erst, wenn ein Formular abgeschickt beziehungsweise die entsprechende Submit-Schaltfläche gedrückt wurde, muss man, wie in **Listing 6** zu sehen, die Eigenschaft *noValidate* auf *true* setzen. Denn standardmäßig wird die native Validierung durchgeführt, bevor das ►

Tabelle 2: Übersicht der Eigenschaften bezüglich der Validierung von Formularen

Eigenschaft	Beschreibung
<i>valid</i>	Liefert die Angabe darüber, ob ein Eingabefeld einen Fehler enthält oder nicht.
<i>valueMissing</i>	Liefert die Angabe darüber, ob ein Eingabefeld ein Pflichtfeld ist, aber kein Wert enthalten ist.
<i>typeMismatch</i>	Liefert die Angabe darüber, ob der geforderte Typ eines Eingabefeldes (zum Beispiel eine E-Mail-Adresse) von dem eingegebenen Wert nicht erfüllt wird.
<i>patternMismatch</i>	Liefert die Angabe darüber, ob der eingegebene Wert dem definierten Muster entspricht oder nicht.
<i>tooLong</i>	Liefert die Angabe darüber, ob der eingegebene Wert zu lang ist.
<i>tooShort</i>	Liefert die Angabe darüber, ob der eingegebene Wert zu kurz ist.
<i>rangeUnderflow</i>	Liefert die Angabe darüber, ob der eingegebene Wert unterhalb des definierten Wertebereichs liegt.
<i>rangeOverflow</i>	Liefert die Angabe darüber, ob der eingegebene Wert oberhalb des definierten Wertebereichs liegt.
<i>stepMismatch</i>	Liefert die Angabe darüber, ob der eingegebene Wert der Definition des <i>step</i> -Attributs entspricht.
<i>badInput</i>	Liefert die Angabe darüber, ob der eingegebene Wert ungültig ist.
<i>customError</i>	Liefert die Angabe darüber, ob der eingegebene Wert einen benutzerdefinierten Fehler auslöst.

Listing 6: Validieren aller Formularfelder innerhalb eines Formulars

```

function init() {
    let registerForm =
    document.getElementById('register');
    registerForm.noValidate = true;
    registerForm.addEventListener('submit',
    validateForm);
}

document.addEventListener('DOMContentLoaded', init);

function validateForm(e) {
    let event = (e ? e : window.event); // Event
    let form = (event.target // Zielelement
    ? event.target
    : event.srcElement);
    let formIsValid = true;
    // Gültigkeit des Formulars
    let formElements =
    // Formularelemente, die ...
    form.querySelectorAll(
    // ... native Validierung ...
    'input, textarea, select');
    // ... unterstützen.
    for (let i = 0; i < formElements.length; i++) {
        let formElement = formElements[i];
        if (formElement.willValidate !== 'undefined') {
            formElement.checkValidity();
        } else {
            // Browser unterstützt nicht
            // native HTML5-Validierung
        }

        if (!field.validity.valid) {
            // Falls Wert nicht gültig ...
            formIsValid = false;
            // ... Formulardaten nicht valide.
        }
    }

    if (!formIsValid) {
        // Falls Formulardaten nicht valide ...
        if (event.preventDefault()) {
            event.preventDefault();
            // ... Standardaktionen verhindern.
        }
    }
    return formIsValid;
}

```

Submit-Event ausgelöst wird. Das bedeutet, dass erst die Standardausgaben des Browsers (inklusive der Standardvalidierungsmeldungen) angezeigt werden, bevor man per JavaScript überhaupt die Möglichkeit hat, eigene Validierungen durchzuführen und angepasste Ausgaben zu erzeugen. Setzt man die Eigenschaft `noValidate` dagegen wie im Bei-

spiel auf `true`, so deaktiviert man die Standardausgaben des Browsers. Der restliche Code im Beispiel sorgt dann dafür, dass alle Formularelemente des Formulars auf Gültigkeit überprüft werden.

Fazit

Für die Validierung von Formulareingaben stehen Ihnen die nativen HTML5-Validierungen in Form des Constraint Validation API auch in JavaScript zur Verfügung. Neben dem Durchführen von Standardvalidierungen ist es mit diesem API auch möglich, eigene Validierungen zu implementieren.

Im Gegensatz zur reinen nativen HTML5-Validierung bietet das API eine deutlich höhere Flexibilität, nicht zuletzt auch im Hinblick auf die Validierungsmeldungen, die relativ einfach den eigenen Anforderungen angepasst werden können. Und das Gute: Das API wird von mehr oder weniger allen aktuellen Browsern unterstützt. ■

Tabelle 3: Methoden bezüglich der nativen Validierung

Methode	Beschreibung
<code>checkValidity()</code>	Diese Methode kann auf einzelnen Formularelementen oder auf Formularen aufgerufen werden. Im ersten Fall wird überprüft, ob das jeweilige Formularelement einen gültigen Wert enthält, im letzteren Fall wird überprüft, ob alle Formularelemente des jeweiligen Formulars gültige Werte enthalten. In beiden Fällen ist das Ergebnis ein boolescher Wert.
<code>reportValidity()</code>	Auch diese Methode kann wie die Methode <code>checkValidity()</code> auf einzelnen Formularelementen oder auf Formularen aufgerufen werden. Im Unterschied dazu wird aber nicht nur auf Gültigkeit der Formularelemente beziehungsweise des Formulars geprüft, sondern auch eventuelle Validierungsmeldungen werden (in ihrem Standardaussehen) im Browser angezeigt.
<code>setCustomValidity()</code>	Über diese Methode ist es möglich, eine angepasste Validierungsmeldung zu verfassen.



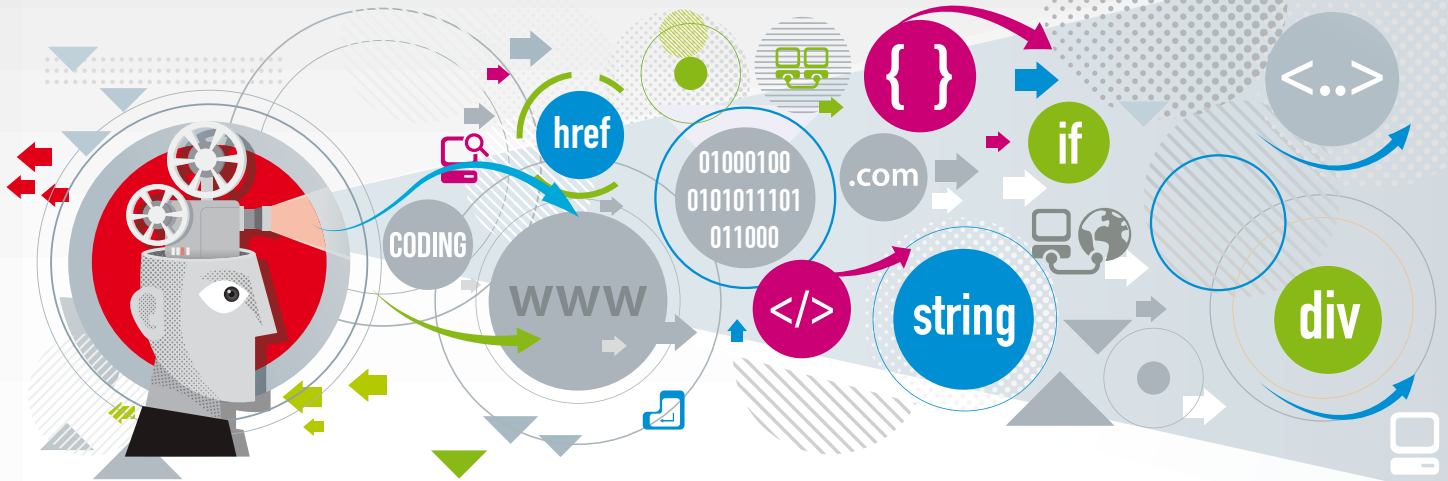
Philip Ackermann

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>



Developer Week 2016

20.-23. Juni 2016,
Messe Nürnberg



Mit Code
DWX16wmd
€ 149,-
sparen!

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**
Job-Shuttle: Tech-on-Tour | Community Night |
DevSessions | Büchercafé u. v. m.
- **Themen (u.a.)**
Architektur | Testing | IoT | DevOps | Angular2 |
Industrie 4.0 | Cloud/Server | UI/UX | Tools
- **10 parallele Workshops am 23.06.2016**
Ganztägig | Praxisorientiert | Vertiefend |
Aktive Teilnahme
- **Fachmesse**
Mehr als 50 Partner präsentieren sich in der
Fachmesse vom 20.-22. Juni 2016.

developer-week.de



DeveloperWeek

Aussteller & Sponsoren:

adorsys



brainLight.
LIFE IN BALANCE

FR
Fast Reports
Reporting must be Fast!

GU
GFU Cyrus AG



MAXIMAGO

MATHEMA

SDX

SHS VIVEON



TEXTCONTROL

hello, technology

UID

Veranstalter:

developer
media

Neue
Mediengesellschaft
Ulm mbH

Präsentiert von:

dotnetpro

web & mobile
DEVELOPER

Grußwort des Wirtschaftsreferenten der Stadt Nürnberg



Dr. Michael Fraas,
Wirtschaftsreferent der Stadt
Nürnberg

Ein herzliches Willkommen allen Konferenzteilnehmerinnen und -teilnehmern auf der Developer Week 2016 in Nürnberg. Zum vierten Mal trifft sich die Entwicklerszene in Nürnberg, um über neue und zukunftsweisende Technologien und digitale Trends zu diskutieren. Der beachtliche Zuwachs der größten unabhängigen Softwareentwick-

ler-Konferenz Europas bei Ausstellern und Besuchern bestätigt die Attraktivität des Themas und des Veranstaltungsortes.

Die Halbmillionenstadt Nürnberg ist das Herzstück einer Europäischen Metropolregion, die mit ihren 3,5 Millionen Menschen zu den zehn großen Wirtschaftszentren Deutschlands gehört. Nürnberg ist High Tech-, Industrie- und Dienstleistungsstandort. Der Standort entwickelt sich zu einer Innovations-Hauptstadt für Zukunftstechnologien, Forschung und Entwicklung.

Mit seinem starken Technologieprofil in Information und Kommunikation versteht sich Nürnberg als digitale Metropole. Mit mehr als 100.000 Beschäftigten ist die Informations- und Kommunikationstechnologie eine sehr bedeutende Branche in der Metropolregion Nürnberg. Knapp 10 % der Beschäftigten in Nürnberg arbeiten im IKT-Sektor – damit belegt Nürnberg einen Spitzenplatz unter den 20 größten deutschen Städten.

Für Ihren Aufenthalt in Nürnberg wünsche ich Ihnen interessante und anregende Diskussionen sowie viele neue inspirierende und ertragbringende Kontakte.

Ich hoffe, Sie finden neben dem vielfältigen Kongressprogramm ein wenig Zeit, Nürnberg näher kennenzulernen. Auch abseits des Kongressgeschehens bietet Nürnberg seinen Gästen viel Sehenswertes.

Ihnen Allen wünsche ich erfolgreiche Konferenztage und einen angenehmen Aufenthalt in Nürnberg.



Dr. Michael Fraas

Wirtschaftsreferent der Stadt Nürnberg

Conference Chairs:



Tilman Börner,
Chefredakteur,
dotnetpro

Tilman Börner ist seit 2003 Chefredakteur der dotnetpro. Er programmiert seit der Schulzeit in Basic, dBASE, Turbo Pascal, Visual Basic, Fortran, C, Delphi, PHP und C#.



Dr. Markus Stäuble,
freier Berater
und Autor

Dr. Markus Stäuble ist passionierter Informatiker, Conference Chair der Developer Week, Fachautor und Programmleiter Make beim Franzis Verlag. Neben Make beschäftigt er sich intensiv mit dem Thema Mobile und hat zu dessen Auswirkungen auf die Arbeitswelt promoviert.

Advisory Board & Track Chairs:



Gregor Biswanger
Cross-Plattform



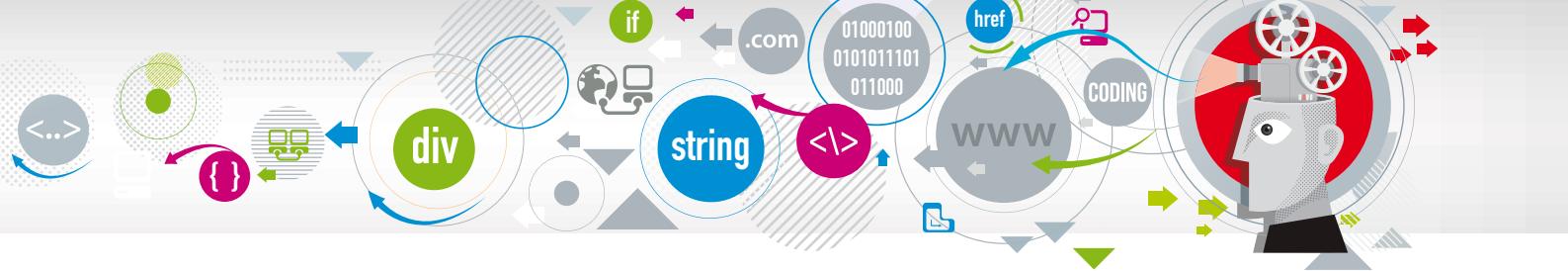
Robert Eichenseer
Cloud/Server



Robert Misch
Softskills



Björn Schotte
Agile



Programm-Übersicht Developer Week 2016

Montag, 20.06.2016	Dienstag, 21.06.2016	Mittwoch, 22.06.2016	Donnerstag, 23.06.2016
Konferenz	Konferenz	Konferenz	Workshops
<ul style="list-style-type: none"> Architektur Datenzugriff Best Practices Frontend DevOps Projektpraxis 	<ul style="list-style-type: none"> .NET Allgemein Web Funktional Entity Framework ALM Cloud/Server 	<ul style="list-style-type: none"> Softwarequalität DevTest Pattern Tools Softskills Agile 	<ul style="list-style-type: none"> Radikale Objektorientierung für die agile Softwareproduktion Entwickeln für die Cloud Parallelprogrammierung mit der TPL Architektur für .NET-Projekte Von den Anforderungen zum Code - Flüssige Softwareentwicklung Angular 2: Einstieg in die komponentenbasierte Entwicklung von Single-Page-Anwendungen Web Components Schnelleinstieg UWP Apps unter Windows 10 Team Foundation Server 2015: Ohne Stress zu häufigen Releases Modernes JavaScript mit ECMAScript 2015
<ul style="list-style-type: none"> Industry 4.0 JavaScript DevOps Projektpraxis 	<ul style="list-style-type: none"> Angular 2 Performance ALM Cloud/Server 	<ul style="list-style-type: none"> UI/UX Better Coding Softskills Agile 	
<ul style="list-style-type: none"> Cross-Plattform Testing DevOps Projektpraxis 	<ul style="list-style-type: none"> Better Apps Datenbanken ALM Cloud/Server 	<ul style="list-style-type: none"> Responsive Mobile Architekturen Softskills Agile 	



Jan Fellien
ALM,
Performance



Dr. Ronald Hartwig



Johannes Hoppe
Angular 2,
JavaScript



André Krämer
Datenzugriff



Patrick Lobacher
Industry 4.0,
Responsive



Hendrik Lösch
Best Practices,
Design Pattern



Ulrike Stirnweiß
Frontend



David Thömmes
UI/UX



David Tielke
Architektur,
Softwarequalität



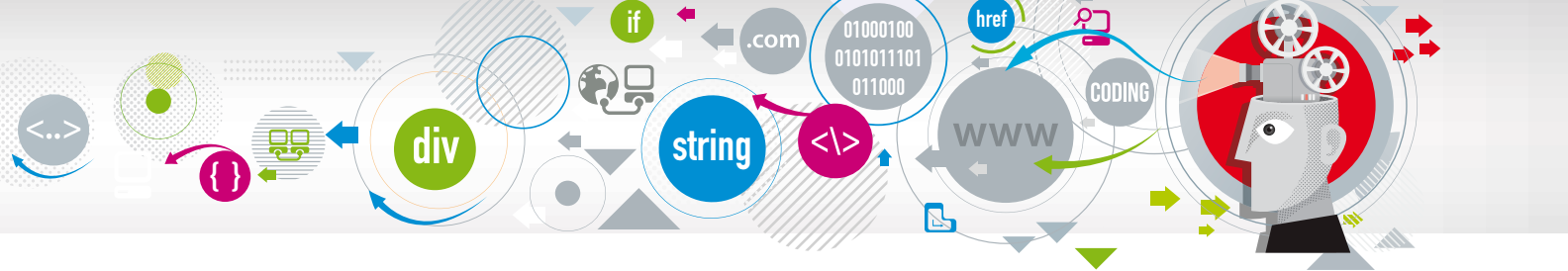
Boris Wehrle
DevOps,
DevTest



Holger Wendel
Better Coding,
Tools

Programm – Tag 1: Montag, 20. Juni 2016

	Architektur	Datenzugriff	Best Practices	Frontend	DevOps
09.00 - 10.00	 Keynote: Managing for Happiness Jürgen Appelo				
10.00 - 10.30	Kaffeepause				
10.30 - 11.30	Composite Components Architecture David Tielke	Performance trotz Entity Framework André Krämer	Was ist eigentlich eine Unit Hendrik Lösch	Datenbindung Deluxe – Deep Dive in das Binding von WPF Christian Giesswein	DevOps – Entwickelst du noch oder lieferst du schon? Thomas Schissler
11.30 - 11.45	Raumwechsel				
11.45 - 12.45	Flow Based Development – was war – was ist – was wird Stefan Dirschnabel	SQL Server 2016 für Entwickler Constantin Klein	KISS, im Prinzip ganz einfach Jan Fellien	The WPF First Aid Kit David Würfel	Entwurf und Bereitstellung von Software für DevOps-Organisationen Michael Kaufmann
12.45 - 14.15	Mittagspause				
14.15 - 15.15	Etwas zum Anfassen Veikko Krypczyk	C# + SQL = Big Data Sascha Dittmann	Legacy Code. Und jetzt? Daniel Marbach	One Code to rule 'em all – Die Windows Universal Platform Marco Richardson	Wenn jede Sekunde zählt – Unternehmen werden zu Software Firmen und Zeit ist hier bares Geld Christian Karl-Heinz Nink
15.15 - 15.30	Raumwechsel				
15.30 - 16.30	Modulares UI – MVVM mit Prism 6 Christian Giesswein	In-Memory Data Grids – supercomputing for the rest of us Ralph Winzinger	Automatisierte Erkennung der Top .NET und Web Performance Fehler in der CI Andreas Grabner	Von A wie Accelerometer bis X wie XmlLite – Das Windows 10 SDK für UWP Apps Alexander Witkowski	Monitoring 2.0: Alles im Lot? Nico Orschel Marc Müller
16.30 - 17.00	Kaffeepause				
17.00 - 18.00	Event Storming Marco Heimeshoff	Daten vor Hackern schützen N.N.	The Impact of Gherkin Albert Schlotter, Maik Devrient	Beyond XAML and HTML David C. Thömmes	Metrik gesteuertes DevOps und Continuous Delivery Andreas Grabner
ab 18.00 #nightone					
	Am ersten Abend der Developer Week gibt es ab 18:00 Uhr neben Speis und Trank auch ein buntes Programm für alle Konferenzbesucher. Diese Programmpunkte sind aktuell geplant: 19.00 Thementische, 20.00 Xamarin-Night, ab 20.00 Ant.Me – C# spielerisch (Hendrik Lösch)				





Projektpraxis	Industry 4.0	JavaScript	Cross-Plattform	Testing	
Keynote: Managing for Happiness <i>Jurgen Appelo</i>					09.00 – 10.00
Kaffeepause					10.00 – 10.30
Visualisierung von Software-Entwicklungsarbeit mit einem flexiblen Dashboard <i>Christian Ringler</i>	Smart Development in the Industry 4.0 – Best Practices and Prospects <i>Alexander Schulze</i>	ECMAScript 2015/2016: Neues aus der JavaScript-Welt <i>Marius Schulz</i>	Xamarin oder Cordova – das ist hier die Frage! <i>Ulrike Stirnweiß, Nina Hauer</i>	UI Testing Strategies <i>Daniel Kersting</i>	10.30 – 11.30
Raumwechsel					11.30 – 11.45
Von Programmierer bis Projektleiter: Eine Sprache sprechen <i>Kurt Salman</i>	IoT Ultimate Edition <i>Damir Dobric</i>	State of Node.js <i>Sebastian Springer</i>	Aber schnell! Mehr Speed für Cross-Plattform-HTML5-Anwendungen <i>Gregor Biswanger</i>	Layout-Testing: Was geht? Was bringt's? Wer braucht's? <i>Christiane Helmchen, Bianca Niestroj</i>	11.45 – 12.45
Mittagspause					12.45 – 14.15
Agile Projekte ohne Scrum <i>Ina Einemann, Ulf Mewe</i>	Innovationsmotoren für IoT <i>Patrick Lobacher</i>	Einführung in React.js <i>N.N.</i>	Daten-Synchronisation für Apps – Offline und ohne Cloud <i>Christian Liebel, Marco Frodl</i>	Practical Mutation Testing <i>Sven Ruppert</i>	14.15 – 15.15
Raumwechsel					15.15 – 15.30
Wettbewerbsvorteile durch den richtigen Umgang mit technischen Schulden <i>André Kolell</i>	Industrie 4.0-Lösungen für Ihre Kunden und sich selber liefern lassen <i>Lars Roith</i>	Testing unchained <i>Dominik Ehrenberg</i>	Ionic Apps mit Cloud Backend Services – Ein Dreamteam <i>Simon Martin Reimler</i>	Webtests automatisieren <i>N.N.</i>	15.30 – 16.30
Kaffeepause					16.30 – 17.00
Work just simply smarter – für bessere Projekte und zufriedene Beteiligte <i>Stephan Kaps</i>	(UML) Modell-basiert Code generieren und debuggen – ja es funktioniert wirklich! <i>Daniel Sigl</i>	Domain-Driven Design mit JavaScript <i>Gregor Biswanger</i>	Introduction to NativeScript <i>Sebastian Witalec, Johannes Hoppe</i>	Test-Gap-Analyse: Erfahrungen aus drei Jahren Praxiseinsatz <i>Dennis Pagano</i>	17.00 – 18.00
#nightone					ab 18.00
Am ersten Abend der Developer Week gibt es ab 18:00 Uhr neben Speis und Trank auch ein buntes Programm für alle Konferenzbesucher. Diese Programmpunkte sind aktuell geplant: 19.00 Thementische, 20.00 Xamarin-Night, ab 20.00 Ant.Me – C# spielerisch (Hendrik Lösch)					

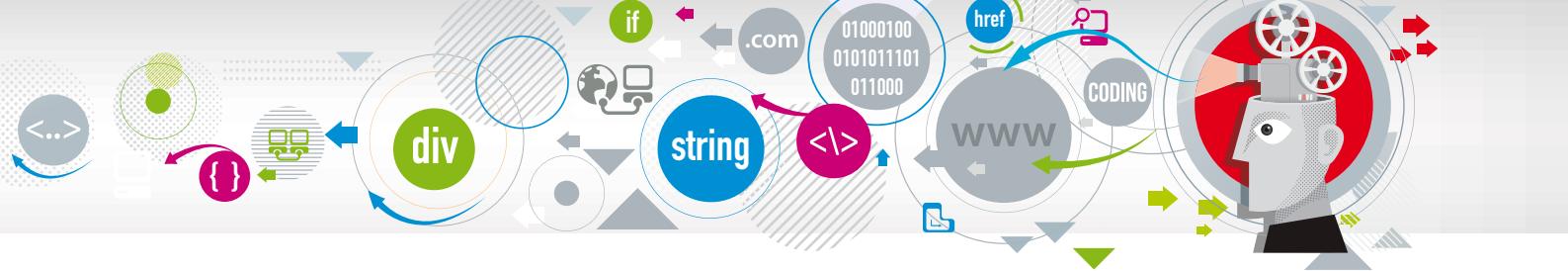
Programmänderung vorbehalten



DeveloperWeek

Programm – Tag 2: Dienstag, 21. Juni 2016

	.NET Allgemein	Web	Funktional	Entity Framework	ALM
09.00 – 10.00	Visual Studio 2015 Geheimtipps <i>Manuel Meyer</i>	Web APIs mit ASP.NET MVC 6 <i>Manfred Steyer</i>	Einführung in die funktionale Programmierung mit F# <i>Carsten König</i>	Entity Framework hinter den Kulissen <i>André Krämer</i>	Docker 101 <i>Lars Kumbier</i>
10.00 – 10.30	Kaffeepause				
10.30 – 11.30	akka.net – Einführung in das Actor Model mit .NET <i>Ralph Waldenmaier</i>	NancyFX: Das Framework für die Web-Entwicklung <i>N.N.</i>	DDD mit Funktionalen Sprachen <i>Marco Heimeshoff</i>	Plattformunabhängiger Datenzugriff mit Entity Framework Core 1.0 <i>Dr. Holger Schwichtenberg</i>	Mit fünf Schritten effizienter Software entwickeln <i>Karsten Kempe</i>
11.30 – 11.45	Raumwechsel				
11.45 – 12.45	Continuous Delivery for .NET Projects at adidas <i>Paul Vassu, Daniel Eichten</i> 	Enabling Plugins in your web application with MEF <i>Don Wibier</i> 	Event Sourcing funktional <i>Carsten König</i>	Datenbasierte Services mit Entity Framework und Co. <i>Manfred Steyer</i>	Build und Release Automation mit TFS 2015 / VSTS <i>Marc Müller, Nico Orschel</i>
12.45 – 14.15	Mittagspause				
14.15 – 16.15 DevSessions	.NET Core <i>David Tielke</i>	Mit SignalR Datenverbindungen zwischen Server und Client herstellen <i>N.N.</i>	Funktionale und asynchrone Pipelines mit C# <i>Daniel Marbach</i>	Entity Framework: Tipps & Tricks <i>Dr. Holger Schwichtenberg</i>	Qualität von Anfang an - Best Practices zur Integration von agilem Testen aus realen Projekten <i>Holger Hofmeister</i>
16.15 – 16.45	Kaffeepause				
16.45 – 18.45 DevSessions	WPF Troubleshooting in Visual Studio 2015 <i>Manuel Meyer</i>	Datenlieferant: GraphQL mit .NET nutzen <i>Philip Jander</i>	Systematische Entwicklung mit funktionaler Programmierung <i>Michael Sperber</i>	Entity Framework und WPF <i>Christian Giesswein</i>	Schnellere Build-Prozesse mit IncrediBuild <i>Robin Sedlaczek</i>
ab 19.00	#community-night				
	Mitmachen und Spaß haben erwünscht! Am Abend des 21.06.2016 öffnet die DWX ab 19:00 Uhr ihre Türen und lädt alle Entwickler und Freunde ein zur #communitynight mit abwechslungsreichem Programm. Auch selbst mitmachen ist erwünscht. Das ausführliche Programm finden Sie auf der Webseite.				

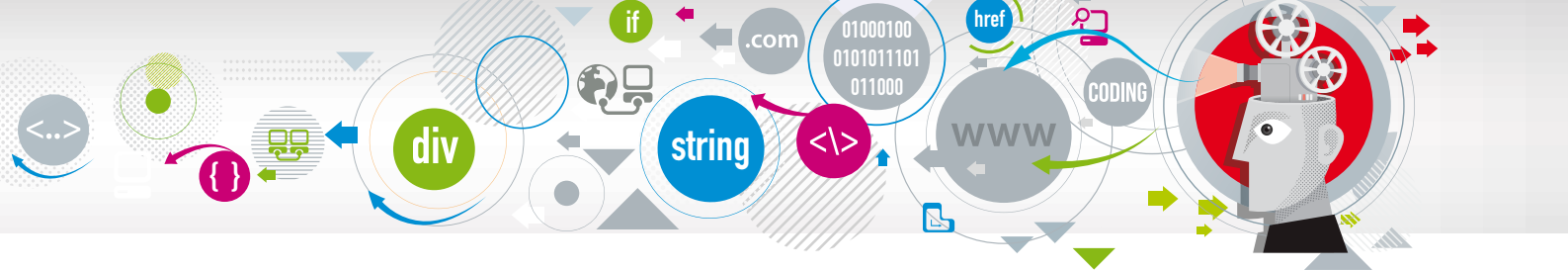


Cloud/Server	Angular 2	Performance	Better Apps	Datenbanken	
„Backend vNext“ mit Microservices und Actor Programming Model <i>Damir Dobric</i>	Speed-up AngularJS – Hochperformante Webanwendungen bauen <i>Timo Korinth</i>	HTTP/2: Die neue Generation für Performance <i>Sascha Schumann</i>	Mobile App-Entwicklung im Team: Tools und Techniken <i>Robert Virkus</i>	o3db – Datenbank-anwendungen fürs Web in Scheme <i>Michael Köhne</i>	09.00 – 10.00
Kaffeepause					10.00 – 10.30
Connecting local data and the cloud – Hybride Lösungen mit Microsoft Azure <i>Sebastian Achatz</i>	Live-Coding mit Angular 2 <i>Johannes Hoppe</i>	Design the Priority: Performance und UX <i>Peter Rozek</i>	Mobile Friendly: Fluch oder Segen? <i>Thomas Kaiser, Paul Dougherty</i>	Automatisierte Datenbankkänderungen – Vergleich der Open-Source-Tools Flyway und LiquiBase <i>Stephan Kaps</i>	10.30 – 11.30
Raumwechsel					11.30 – 11.45
Azure: ein Überblick über die Möglichkeiten der Cloud <i>N.N.</i>	Angular 2 Change Detection Explained <i>Pascal Precht</i> 	JavaScript Performance Analyse <i>Sebastian Springer</i>	Ist Ihre App sicher? <i>Kerry W. Lothrop</i>	Database Lifecycle Management – die nächste Stufe für Ihre Datenbank-entwicklung <i>Constantin Klein</i>	11.45 – 12.45
Mittagspause					12.45 – 14.15
In wenigen Minuten zur Serverlandschaft für meine Apps <i>Sascha Dittmann</i>	Schnellstart mit Angular 2 <i>Johannes Hoppe, Gregor Woiwode</i>	Performance & Load Testing mit Visual Studio richtig gemacht ... <i>Nico Orschel, Marc Müller</i>	Produktionsdaten in der App effizient darstellen <i>N.N.</i>	SQL Server – Sicherheitkonzepte <i>Thorsten Kansy</i>	14.15 – 16.15 DevSessions
Kaffeepause					16.15 – 16.45
Containers, Service Fabric and Azure – Globally scalable, available architectures in Cloud Environments <i>Robert Eichenseer</i>	Angular 2 - Upgrade <i>Manfred Steyer</i>	Reaktive Web-Anwendungen mit RxJS <i>Michael Menzel</i>	Performancetuning von Apps <i>N.N.</i>	NoSQL mit PostgreSQL <i>Stephan Hochdörfer</i>	16.45 – 18.45 DevSessions
#community-night					ab 19.00
Mitmachen und Spaß haben erwünscht! Am Abend des 21.06.2016 öffnet die DWX ab 19:00 Uhr ihre Türen und lädt alle Entwickler und Freunde ein zur #communitynight mit abwechslungsreichem Programm. Auch selbst mitmachen ist erwünscht. Das ausführliche Programm finden Sie auf der Webseite.					

Programmänderung vorbehalten

Programm – Tag 3: Mittwoch, 22. Juni 2016

	Softwarequalität	DevTest	Pattern	Tools	Softskills
09.00 – 10.00	Softwarequalität <i>David Tielke</i>	Eigenschafts-basierendes Testen <i>Carsten König</i>	Lose gekoppelt wie nie: DI vs. IoC <i>Hendrik Lösch</i>	Visual Studio 2015 Extensions leichtgemacht <i>Christian Giesswein</i>	Innovativ, kreativ und begeisternd – aber wie? <i>Anja Schwarz</i>
10.00 – 10.30	Kaffeepause				
10.30 – 11.30	[Aus-]Kommentiert <i>Michael Wiedeking</i>	Testmanagement in der agilen Transition <i>Kay Grebenstein</i>	Async/Await. Die Würfel sind gefallen! <i>Daniel Marbach</i>	Zehn kostenfreie Visual Studio-Extensions, die Sie kennen sollten <i>Dr. Holger Schwichtenberg</i>	Von Ärzten, Piloten und Flugzeugen: Mehr Effizienz in der Softwareentwicklung <i>Christian Robert</i>
11.30 – 11.45	Raumwechsel				
11.45 – 12.45	Die S.O.L.I.D.-Prinzipien für C#-Entwickler <i>Thomas Claudius Huber</i>	Unit Testing Reifegrade <i>Frank Sons</i>	Microservices patterns with Spring Cloud <i>Paul Vassu, Daniel Eichten</i> 	Ein Blick in meinen Werkzeugkasten für Desktop-anwendungen <i>Hendrik Lösch</i>	Einmal mit Profis arbeiten <i>Martin Walter</i>
12.45 – 14.15	Mittagspause				
14.15 – 15.15	.NET Native <i>David Tielke</i>	Testend Entwickeln – Entwickelnd Testen <i>Martin Uhlig, Michael Thiele</i>	Practical Proxy DeepDive <i>Sven Ruppert</i>	AMP – Eine Bibliothek für einfachen Zugriff auf die GPU <i>Bernd Marquardt</i>	Auf der Suche nach dem perfekten Mitarbeiter oder vom T-Shape zum Team-Shape <i>Ulf Mewe</i>
15.15 – 15.30	Raumwechsel				
15.30 – 16.30	Planlos mit Plan – wie erhöhe ich die Verlässlichkeit der Planung in der Softwareentwicklung? <i>Frank Düsterbeck</i>	Entwicklerproduktivität mit ALM Rangers Solutions für TFS & Visual Studio Real Live Scenario <i>Johannes Cosmin Dumitru</i>	Digitale Transformation <i>Golo Roden</i>	Office 365 – Die USB-Plattform von Microsoft <i>Alexander Tews</i>	Mit Selbstorganisation zur besseren Software <i>N.N.</i>
16.30 – 17.00	Kaffeepause				
17.00 – 18.00	 Abschluss-Keynote: JavaScript, The Web, Mobile, and the Rise of the New Virtual Machine <i>Scott Hanselman</i>				



Agile	UI/UX	Better Coding	Responsive	Mobile Architekturen	
Agil und Konzern – Das kannst du schon so machen, aber...? <i>Dominik Ehrenberg, Sebastian Bauer</i>	User Experience Design und Usability <i>Brandstifter David C. Thömmes</i>	Continuous Quality – Gleichbleibende Qualitätsstandards durch (voll-)automatisierte Releaseprozesse sicherstellen <i>Thomas Rümmler</i>	Full Responsive <i>Sven Wolfermann</i>	Native Mobile Apps mit NativeScript und Angular 2.0 <i>Johannes Hoppe, Sebastian Witalec</i>	09.00 – 10.00
Kaffeepause					10.00 – 10.30
Alles wird gut – wir machen jetzt Scrum! <i>Julia Schmidt</i>	„Was würde Einstein sagen?“ – Personas als Werkzeug zur Verbesserung der Gebrauchstauglichkeit von interaktiven Systemen <i>Armin Reuter</i>	Softwareentwicklung ohne Abhängigkeiten <i>Stefan Lieser</i>	Atomic Design – Die Einheit von Frontend und Design im RWD-Zeitalter <i>Patrick Lobacher</i>	MVVM und FRP: Perfekte Architekturen für komplexe mobile Anwendungen? <i>Dr. Wolfram Schroers</i>	10.30 – 11.30
Raumwechsel					11.30 – 11.45
Gerade genug Architektur vorneweg – Zur eigenen Architektur-Vision in 16 Minuten <i>Roland Mast</i>	Turn your ideas into reality – Prototyping in der Praxis <i>Ulrike Stirnweiß, Nina Hauer</i>	Demystifying automated testing in enterprise projects <i>Denis Biondic</i> 	Responsive Design: Der Inhalt entscheidet, nicht die Auflösung. <i>Daniel Kersting</i>	Mobile Anwendungen synchronisieren mit Immutability <i>Michael Sperber</i>	11.45 – 12.45
Mittagspause					12.45 – 14.15
Agile Methoden in verteilten Teams – Was hat sich bewährt, was nicht? <i>Thomas Rümmler</i>	Sie sind Software Engineer? Dann sind Sie UX Designer! <i>Alexander Keller</i>	Asynchrone Programmierung 101 <i>Golo Roden</i>	Mobile First <i>N.N.</i>	Architekturen für das Mobile Enterprise <i>Thomas Künneth</i>	14.15 – 15.15
Raumwechsel					15.15 – 15.30
Lessons learned: Sprung von einer kleinen Firma in ein stark wachsendes Unternehmen <i>Sebastian Hoitz</i>	WPF und Parallelprogrammierung <i>Bernd Marquardt</i>	Refresh: Migration einer Legacy-Anwendung <i>Veikko Krypczyk</i>	Responsive Web Design for Developers <i>Don Wibier</i> 	Advanced Mobile Cross-Platform Architecture <i>Jens Becker</i>	15.30 – 16.30
Kaffeepause					16.30 – 17.00
 Abschluss-Keynote: JavaScript, The Web, Mobile, and the Rise of the New Virtual Machine <i>Scott Hanselman</i>					17.00 – 18.00

Programmänderung vorbehalten



DeveloperWeek

Workshops – Tag 4: Donnerstag, 23. Juni 2016

Workshop 1

Radikale Objektorientierung für die agile Softwareproduktion

Workshopleiter: Ralf Westphal

Workshopdauer: 09.00-17.00 Uhr

Die Objektorientierung hatte versprochen, die Entwicklung von Softwaresystemen zu verbessern. Aber Technical debt, brownfield, legacy code belasten die meisten Projekte – allerdings nicht trotz sondern eher wegen ihrer Objektorientierung. Im Workshop lernen Sie eine radikale Objektorientierung kennen. Radikal, weil sie zurückgeht an die Wurzeln. Dort findet sich nämlich das zentrale Merkmal für Objektorientierung. Messaging ist der Schlüssel zu vielen Tugenden guten Softwaredesigns wie Entkopplung, Testbarkeit, Kapselung, klaren Verantwortlichkeiten und sogar agilem Vorgehen. Als Teilnehmer lernen Sie die wesentlichen Prinzipien radikaler Objektorientierung kennen und wenden sie in mehreren Beispielprojekten an. Und Sie erfahren, wie ein Umgang mit Anforderungen aussehen muss, um die radikale Objektorientierung nahtlos in agiles Vorgehen einzubetten.



Workshop 2

Entwickeln für die Cloud

Workshopleiter: Golo Roden

Workshopdauer: 09.00-17.00 Uhr

Die Cloud ist die Zukunft, das steht außer Frage. Doch der Weg ist steinig. Sie benötigen REST-APIs, Streaming und Echtzeitdaten. Das alles skalierbar, verlässlich und effizient. Für welche Technologie entscheiden Sie sich? In diesem Workshop lernen Sie die Plattformen Node.js und io.js kennen, die JavaScript auf dem Server ausführen und so Ihren Aufwand verringern und Ihre Entwicklung beschleunigen.



Workshop 3

Parallelprogrammierung mit der TPL (Task Parallel Library)

Workshopleiter: Bernd Marquardt

Workshopdauer: 09.00-17.00 Uhr

Da die Taktfrequenzen unserer Prozessoren nicht ansteigen, ist die Parallelprogrammierung fast „zur Pflicht“ geworden. Hier helfen moderne Parallel-Bibliotheken damit unsere Anwendungen durch die Parallelisierung nicht unendlich kompliziert werden. Sie lernen die Schleifenparallelisierung, die Task-orientierte Programmierung, die parallelen Container-Klassen, die Fehlerbehandlung und viele andere Themen aus der TPL kennen. Es wird auch auf die „unangenehmen“ Eigenschaften der Parallelprogrammierung eingegangen. Hierzu gehören u.a. Data Races und Dead Locks. Viele Live-Beispiele, an denen das neue Wissen direkt beobachtet werden kann, lockern den Workshop auf.



Workshop 4

Architektur für .NET-Projekte

Workshopleiter: David Tielke

Workshopdauer: 09.00-17.00 Uhr

Ein Architekt sollte in jedem großen Softwareprojekt unter anderem für die Struktur der Anwendung sorgen und diese an den nichtfunktionalen Anforderungen ausrichten. Soweit so gut. Leider ist nicht jedes Projekt von solcher Größe und bei einem kleinen Team existiert nur selten ein dedizierter Softwarearchitekt. Zu oft wird die Notwendigkeit einer ordentlichen Softwarearchitektur in solchen Projekten nicht gesehen, oft mit fatalen Folgen: auf lange Sicht haben solche Projekte Probleme bei der Wartbarkeit, Weiterentwickelbarkeit und vielen anderen Eigenschaften. Wir schauen in diesem Workshop, wie Sie diese Aufgabe ohne Architekturzertifikat meistern.



Workshop 5

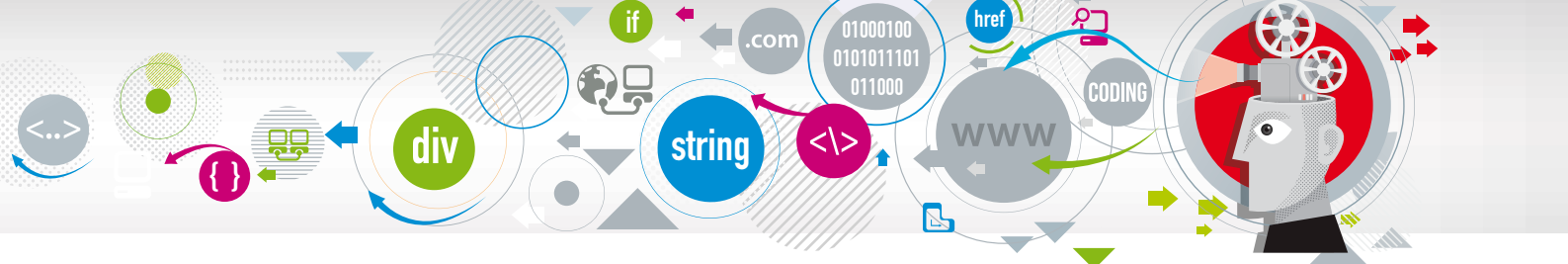
Von den Anforderungen zum Code – Flüssige Softwareentwicklung

Workshopleiter: Stefan Lieser

Workshopdauer: 09.00-17.00 Uhr

Softwareentwickler werden mit Anforderungen konfrontiert und müssen Code produzieren. Doch wie geht man als Softwareentwickler dabei vor? Der Workshop zeigt anhand zahlreicher Beispiele, wie Anforderungen so zerlegt werden, dass konkrete handhabbare Einheiten entstehen, die anschließend entworfen und umgesetzt werden können. Sie lernen, wie Anforderungen in konkrete umsetzbare Einheiten zerlegt werden. Wie Sie eine Lösung für die Anforderungen entwerfen und wie Sie die entworfene Lösung in Quellcode übersetzen. Ziel dieser Vorgehensweise ist ein flüssiger Produktionsprozess, der es erlaubt, Software als Einzelner oder auch im Team von mehreren Entwicklern zu produzieren. Die Werte der CCD-Initiative (Evolvierbarkeit & Korrektheit) werden in den Blick genommen.





Workshop 6

Angular 2: Einstieg in die komponentenbasierte Entwicklung von Single-Page-Anwendungen

Workshopleiter: Johannes Hoppe, Gregor Woiwode

Workshopdauer: 09.00-17.00 Uhr

AngularJS ist das populärste Framework für Single-Page-Anwendungen. Mit Angular 2 mischt Google die Webentwicklung gehörig auf. Das neue Framework setzt auf die Programmiersprache TypeScript, komponentenbasierte Entwicklung, eine neue Template-Syntax und ein verbessertes Tooling. Kurzum: alles ist neu! Unter Anleitung der beiden Angular-Experten lernen Sie folgende Themen kennen: Typescript, Setup einer Angular 2 App, Template-Syntax, Komponenten, Pipes & Direktiven, Navigation mit dem neuem Component-Router, Integration von Polymer & Kendo UI, Tooling, Unit-Test und Oberflächentests. Zum Ende des Workshops haben Sie die Grundlagen und fortgeschrittene Themen von Angular 2 gemeistert. Als Teilnehmer benötigen Sie einen Laptop mit vorinstallierter Software (Chrome, Atom Editor, Node.js) oder Virtual Box.



Johannes Hoppe



Gregor Woiwode

Workshop 7

Web Components

Workshopleiter: Peter Kröner

Workshopdauer: 09.00-17.00 Uhr

Das Webseiten-Modul der Zukunft ist kein jQuery-Plugin mehr, sondern ein selbstdefiniertes HTML-Element. Web Components werden die Entwicklung von Web-Frontend revolutionieren – und mit den richtigen Tools kann die Revolution bereits heute beginnen. Dieser Workshop gibt einen Einblick in die Webstandards hinter Web Components und zeigt Ihnen, wie Sie bequem eigene HTML-Elemente erfinden können. Außerdem führt der Workshop in Googles Polymer-Library für Web Components ein und gibt Ihnen in einem großen Praxisteil die Gelegenheit, erste Hands-On-Erfahrung mit Web Components zu sammeln. Für den Workshop werden ein Laptop mit modernen Browsern (Chrome, Firefox), einem beliebigen Code-Editor und einem beliebigen lokal installierten Webserver benötigt.



Workshop 8

Schnelleinstieg UWP Apps unter Windows 10

Workshopleiter: Alexander Witkowski

Workshopdauer: 09.00-17.00 Uhr

Der Workshop gibt einen Einstieg in die Entwicklung von Apps für Windows 10. Nur eine App mit nur einem SDK läuft auf den unterschiedlichsten Plattformen. Vom Raspberry Pi über das Telefon, das Tablet bis zum Desktop PC und der Xbox. Begonnen bei den notwendigen Werkzeugen wird, neben der Entwicklung selbst, ein Blick auf die Richtlinien und das Verteilen der Anwendungen über den Store geworfen. Auch der Zugriff auf Sensoren, das Verwenden einer Datenbank und das MVVM Pattern kommen im Workshop zum Einsatz. Am Ende des Tages gehen Sie mit einer Universal Windows Plattform App nach Hause und wissen, wie Sie diese weiterentwickeln und in den Store bringen.



Workshop 9

Team Foundation Server 2015: Ohne Stress zu häufigen Releases

Workshopleiter: Neno Loje

Workshopdauer: 09.00-17.00 Uhr

In diesem interaktiven Workshop lernen Sie, wie Sie mit dem Team Foundation Server (TFS) den Prozess von der Anforderung, über Implementierung, Test bis hin zur Auslieferung effizient und schlank gestalten. Um häufige Releases zu ermöglichen bietet der TFS vielfältige Möglichkeiten der Automatisierung. In einer End-to-End-Demo wird auf die verschiedenen Aspekte eingegangen, um zeitnah auf Kunden und Marktanforderungen reagieren zu können. Freuen Sie sich auf einen lehrreichen und unterhaltsamen Tag zusammen mit dem TFS-Experten & MVP Neno Loje. Und: Bringen Sie gerne Ihre Fragen mit!



Workshop 10

Modernes JavaScript mit ECMAScript 2015

Workshopleiter: Marius Schulz

Workshopdauer: 09.00-17.00 Uhr

Mit ECMAScript 2015, früher ECMAScript 6, wurde JavaScript nach vielen Jahren erheblich ausgebaut. Die umfangreichste Erweiterung ist die Einführung eines nativen Modulsystems, mit dem sich auch größere Anwendungen sauber strukturieren lassen. Des Weiteren erleichtern diverse kleinere Features die Arbeit mit JavaScript. In dem Workshop werden u.a. die folgenden Sprachkonstrukte von ECMAScript 2015 behandelt: Natives Modulsystem, Konstanten und Block-Scoping, Arrow-Funktionen und lexikalisches „this“-Binding, Destrukturierung von Objekten und Arrays, Default-Werte für Parameter, Template-Strings und String-Interpolation, Rest- und Spread-Operator, Klassen und Vererbung, Iteratoren und Iteration. Zusätzlich lernen Sie einen Transpiler einzusetzen, der die neuen Sprachkonstrukte so umschreiben kann, dass sie von allen gängigen Browsern verstanden werden.



Jetzt Ticket sichern: developer-week.de/anmeldung

<p>○ Kombi-Ticket DWX & Workshop 20.-23. Juni 2016 Ticketpreis 1.750,- € zzgl. MwSt. (statt 1.899,- € zzgl. MwSt.)</p>	<p>○ 3-Tages-Ticket 20.-22. Juni 2016 Ticketpreis 1.250,- € zzgl. MwSt. (statt 1.399,- € zzgl. MwSt.)</p>	<p>○ 2-Tages-Ticket 20./21.06.2016 od. 21./22.06.2016 Ticketpreis 850,- € zzgl. MwSt. (statt 999,- € zzgl. MwSt.)</p>	<p>○ 1-Tages-Ticket 20.06.2016, 21.06.2016 oder 22.06.2016 Ticketpreis 650,- € zzgl. MwSt. (statt 799,- € zzgl. MwSt.)</p>	<p>○ Workshop-Ticket 23.06.2016 Ticketpreis 650,- € zzgl. MwSt. (statt 799,- € zzgl. MwSt.)</p>
--	---	---	--	---



Die Referenten der Developer Week

Achatz, Sebastian
Appelo, Jurgen, Jojo Ventures BV
Bauer, Sebastian, Inovex GmbH
Becker, Jens, Incloud GmbH
Biondic, Denis, complement AG
Biswanger, Gregor, CleverSocial.de
Devrient, Maik, DATEV eG
Dirschnabel, Stefan, Codelution
Dittmann, Sascha, Microsoft Deutschland GmbH
Dobric, Damir, Daenet GmbH
Dougherty, Paul, Forecheck LLC
Dumitru, Johannes C.
Düsterbeck, Frank, HEC GmbH
Ehrenberg, Dominik, Infineon Technologies AG
Eichenseer, Robert, Microsoft Corp.
Eichten, Daniel, Adidas-Salomon AG
Einemann, Ina, HEC GmbH
Fellien, Jan, devCrowd GmbH
Frodl, Marco, Thinkecture AG
Giesswein, Christian, Giesswein Creativ GmbH
Grabner, Andreas, Dynatrace, Austria
Greibenstein, Kay, Saxonia Systems AG
Hanselman, Scott, Microsoft Corp.
Hauer, Nina, complement AG
Heimeshoff, Marco, Heimeshoff IT
Helmchen, Christiane, punkt.de GmbH
Hochdörfer, Stephan, bitExpert AG
Hofmeister, Holger, complement AG
Hoitz, Sebastian, komola GmbH
Hoppe, Johannes, HAUS HOPPE - ITS
Huber, Thomas, Trivadis AG
Jander, Philip, Jander IT
Kaiser, Thomas, cyberpromote GmbH
Kansy, Thorsten,
DotNetConsulting by Thorsten Kansy
Kaps, Stephan, Bundesversicherungsamt
Kaufmann, Michael,
Alegri International Service GmbH
Keller, Alexander, Centigrade GmbH
Kempe, Karsten, KWP Projekt-Vertriebs GmbH
Kersting, Daniel, User Interface Design GmbH

Klein, Constantin, Freudenberg IT GmbH & Co. KG
Köhne, Michael, o3db.com
Kolell, André,
FFG Finanzcheck Finanzportale GmbH
König, Carsten, Wiegand-Glas GmbH
Korinth, Timo, MAXIMAGO GmbH
Krämer, André, André Krämer - Software,
Training & Consulting
Kröner, Peter, Brainfire Design
Krypczyk, Veikko, internationale Berufsakademie
Kumbier, Lars, Kumbier IT Consulting
Künneht, Thomas, MATHEMA Software GmbH
Liebel, Christian, Thinkecture AG
Lieser, Stefan
Lobacher, Patrick, +Pluswerk AG
Loje, Neno, NenoLoje.de
Lösch, Hendrik, Saxonia Systems AG
Lothrop, Kerry W., Zühlke Engineering GmbH
Marbach, Daniel, tracelight GmbH
Marquardt, Bernd, Bernd Marquardt
Mast, Roland, Sybit GmbH
Menzel, Michael, Senacor Technologies AG
Mewe, Ulf, HEC GmbH
Meyer, Manuel, Trivadis AG
Müller, Marc, 4tecture GmbH
Niestroj, Bianca, punkt.de GmbH
Nink, Christian, New Relic International Ltd.
Orschel, Nico, AIT GmbH & Co. KG
Pagano, Dennis, CQSE GmbH
Precht, Pascal, thoughttram GmbH
Reimler, Simon Martin,
arvato Bertelsmann SE & Co. KGaA
Reuter, Armin, Heinrich & Reuter Solutions GmbH
Richardson, Marco, Microsoft Deutschland GmbH
Ringler, Christian,
GfK Retail and Technology GmbH
Robert, Christian, SapientNitro
Roden, Golo,
the native web ug (haftungsbeschränkt)
Roith, Lars, AIT GmbH & Co. KG
Rozek, Peter, Ex.io germany GmbH

Rümmeler, Thomas, AIT GmbH & Co. KG
Ruppert, Sven, Macros Reply GmbH
Salman, Kurt, CoMo Solutions GmbH
Schissler, Thomas, artiso solutions GmbH
Schlotter, Albert, DATEV eG
Schmidt, Julia, BERATUNG JUDITH ANDRESEN
Schroers, Dr. Wolfram,
Numerik & Analyse Schroers
Schulz, Marius, 69 Grad GmbH
Schulze, Alexander, Inntrade GmbH
Schumann, Sascha, Myra Security GmbH
Schwarz, Anja, SOPHIST GmbH
Schwichtenberg, Dr. Holger, www.IT-Visions.de
Sedlaczek, Robin, Fairmas GmbH
Sieg, Daniel, LieberLieber Software GmbH
Sons, Frank, code-qualiy.de
Sperber, Michael, Active Group GmbH
Springer, Sebastian, MaibornWolff GmbH
Steyer, Manfred, www.IT-Visions.de
Stirnweiß, Ulrike, complement AG
Tews, Alexander, complement AG
Thiele, Michael, Saxonia Systems AG
Thömmes, David C., Shapefield UG
Tielke, David, david-tielke.de
Uhlig, Martin, Saxonia Systems AG
Vassu, Paul, Adidas Group
Virkus, Robert, Enough Software GmbH & Co.KG
Waldenmaier, Ralph,
PROGRESS SOFTWARE GmbH
Walter, Martin, Deutsche Welle
Westphal, Ralf, One Man Think Tank
Wibier, Don, Developer Express Inc.
Wiedeking, Michael, MATHEMA Software GmbH
Winzinger, Ralph, Senacor Technologies AG
Witalec, Sebastian, Telerik
Witkowski, Alexander, Marktjagd GmbH
Woiwode, Gregor, heco GmbH
Wolfermann, Sven, maddesigns
Würfel, David, Centigrade GmbH

Kooperationspartner (Stand: 15.03.2016):

Veranstalter:



GOOGLE CHART TOOLS

Vielfältige Diagramme

Mit den Google Chart Tools lassen sich vielfältige Diagramme in HTML-Seiten einbauen.

Auf vielen Webseiten werden zur Veranschaulichung eines Sachverhalts Daten als Diagramme dargestellt. Google Chart Tools bietet Ihnen dafür eine Sammlung von verschiedenen Diagrammen wie Torten-, Balken-, Linien- oder auch Länderdiagramme. Auf einfache Weise können die Diagramme mit Ihren Daten gefüllt und für Ihre Internetseite angepasst werden.

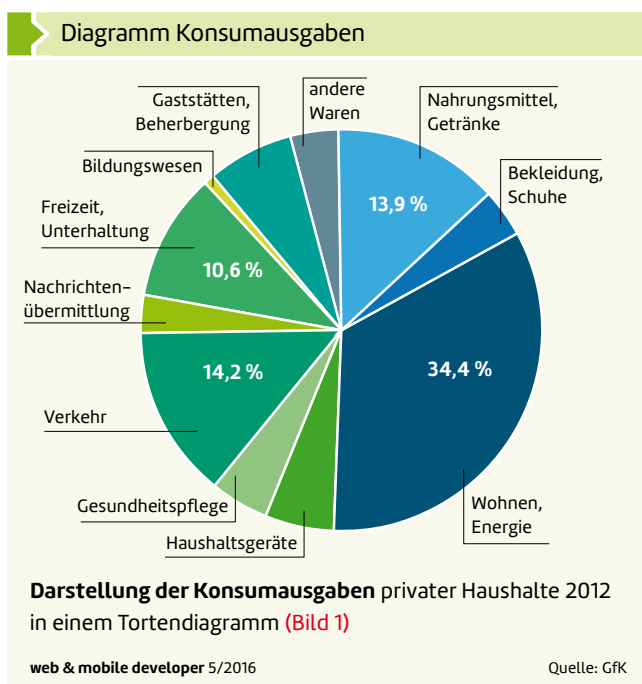
Um die Diagramme in Ihre Seite einbauen zu können, benötigen Sie lediglich einige Librarys von Google, die mit Hilfe von JavaScript in die HTML-Seite eingefügt werden. Die Diagramme werden mit Hilfe von HTML5/SVG dargestellt und bieten somit eine browserübergreifende Kompatibilität. Sie können auf diese Weise mit allen gängigen aktuellen Browsern dargestellt werden.

Einfaches Beispiel

In einem einfachen Beispiel werden die Konsumausgaben privater Haushalte in einem Tortendiagramm dargestellt. Den Sourcecode dazu können Sie sich in [Listing 1](#) ansehen, die Darstellung im Browser in [Bild 1](#).

Im ersten Schritt werden die benötigten Librarys geladen. Es handelt sich dabei um folgende drei Librarys:

- das Google JSAPI API,
- die Google Visualization Library,
- die eigentliche Bibliothek für das Diagramm (das Package).



Im ersten Schritt wird das JSAPI geladen, im nächsten die Visualization Library mit dem Package *corechart*. Dieses Package beinhaltet die gebräuchlichsten Diagramme wie Torten-, Balken- oder Liniendiagramme. Nach dem Laden der Bibliotheken wird die Funktion *google.setOnLoadCallback(drawChart)* aufgerufen, durch die die Funktion *drawChart* aufgerufen wird, in der anschließend das Diagramm erstellt wird.

In dieser Funktion werden zunächst die Daten für das Diagramm bereitgestellt. Dies erfolgt mit Hilfe der *google.visualization.DataTable()*. Es werden die Tabellenspalten für die Art der Ausgaben und den Betrag in Euro erstellt und anschließend mit Werten gefüllt. Der nächste Schritt, die Definition der *options*, ist für eine funktionierende Grafik nicht notwendig, durch ihn können Sie aber das Diagramm an Ihre Anforderungen anpassen.

Im Beispiel wird hier definiert, dass die Legende links von der Grafik steht und dass das Diagramm dreidimensional dargestellt wird. Der Titel wird festgelegt, und es werden Angaben zur Höhe und Breite des Diagramms gemacht. Schließlich werden noch die Farben der einzelnen Tortenstücke definiert.

Im letzten Schritt muss das Diagramm gezeichnet werden. Dazu wird zunächst eine neue Instanz des Diagrammtyps erstellt, der gezeichnet werden soll, in diesem Fall ein PieChart. Daran wird die Funktion *draw* aufgerufen, der die Daten und Optionen als Parameter übergeben werden. Eine letzte Sache fehlt noch zum erfolgreichen Zeichnen des Diagramms: In der HTML-Seite muss ein Element erstellt werden, das als Behälter des Diagramms dient und das über eine ID angesprochen werden kann. In der Regel wird hierzu ein `<div>` verwendet, hier das `<div id="chart_div"></div>`.

Benutzer-Interaktion

Es ist auch sehr einfach möglich, interaktives Verhalten einzubauen. Einige typische Events, die von einem Diagramm ausgelöst werden sind folgende:

- **ready:** Wird nach dem Zeichnen des Diagramms ausgelöst.
- **select:** Wird beim Klick auf ein Diagrammteil, im Beispiel ein Tortenstück, ausgelöst.
- **error:** Wird geworfen wenn die eingegebenen Daten nicht angezeigt werden können.
- **onmouseover/onmouseout:** Wird beim Bewegen der Maus über einem Element oder aus einem heraus ausgelöst.

Um im obigen Beispiel eine Reaktion auf das Klicken auf ein Tortenstück auszulösen, ist lediglich folgender Code nötig:

```
function selectHandler() {
    var selectedItem = chart.getSelection()[0];
```

```

if (selectedItem) {
    var ausgabenArt = data.getValue
    (selectedItem.row, 0);
    var ausgaben = data.getValue(selectedItem.row, 1);
    alert('Die Ausgaben für ' + ausgabenArt +
    ' betragen ' + ausgaben + ' Euro');
}
}
google.visualization.events.addListener
(chart, 'select', selectHandler);

```

Mit der Funktion *selectHandler* wird ein Handler definiert, der für das ausgewählte Element ein Fenster mit den Ausgaben in der Einheit Euro aufpoppen lässt. Dieser Handler wird seinerseits durch einen Event Listener, der auf *select* hört, aufgerufen.

Spezialisierte Packages

Neben dem Package *corechart*, das die gängigen Diagrammarten enthält, gibt es noch weitere spezialisierte Packages für jeweils eine spezielle Diagrammart. Im Folgenden wird in diesem Artikel noch die Diagrammart *timeline*, mit Hilfe derer ein Zeitstrahl dargestellt werden kann, und die Diagramm-

art *geochart* zur Darstellung von Daten auf geografischen Flächen vorgestellt.

In diesem Beispiel wird ein Zeitstrahl erstellt, der die Regierungszeiten der Bundeskanzler von Konrad Adenauer bis Angela Merkel darstellt (Bild 2).

Die Struktur der HTML-Datei ist die gleiche wie im ersten Beispiel. Im *<head>* werden zunächst die Librarys geladen. Hier besteht der einzige Unterschied darin, dass ein anderes Package geladen wird:

```

google.load('visualization', '1.0',
{'packages':['timeline']});

```

Anschließend wird in der Funktion *drawChart* wieder eine *dataTable* erstellt und Spalten werden hinzugefügt:

```

var dataTable = new google.visualization.DataTable();
dataTable.addColumn({ type: 'string', id: 'Term' });
dataTable.addColumn({ type: 'date', id: 'Start' });
dataTable.addColumn({ type: 'date', id: 'End' });

```

Die Unterschiede zum Tortenbeispiel liegen in der Anzahl und dem Datentyp der Spalten. Da die Spalten zwei und drei

Listing 1: Tortendiagramm

```

<html>
<head>
    <script type="text/javascript"
    src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

        google.load('visualization', '1.0',
        {'packages':['corechart']});
        google.setOnLoadCallback(drawChart);

        function drawChart()
        {
            var data = new google.visualization.Data
            Table();
            data.addColumn('string', 'Art der Ausgaben');
            data.addColumn('number', 'Betrag in Euro');
            data.addRows([
                ['Nahrungsmittel, Getränke, Tabakwaren',
                321],
                ['Bekleidung, Schuhe', 106],
                ['Wohnen, Energie, Wohnungsinstandhaltung',
                796],
                ['Innenausstattung,
                Haushaltsgüter, -gegenstände', 128],
                ['Gesundheitspflege', 96],
                ['Verkehr', 329],
                ['Nachrichtenübermittlung', 57],
                ['Freizeit, Unterhaltung, Kultur', 245],
                ['Bildungswesen', 16],

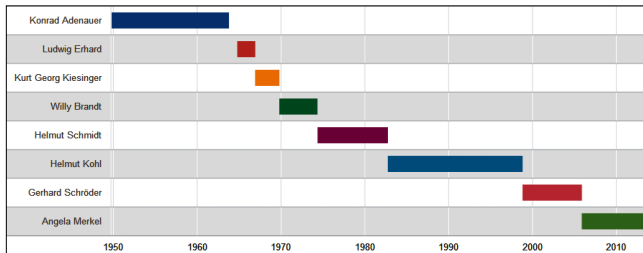
                ['Beherbergungs- und
                Gaststättendienstleistungen', 127],
                ['andere Waren und Dienstleistungen', 90]
            ]);

            var options = {
                'legend':'left',
                'title':'Konsumausgaben privater Haushalte
                2012 (Quelle: Statistisches Bundesamt)',
                'is3D':true,
                'width':1300,
                'height':500,
                'colors': ['#000066', '#0000FF', '#006600',
                '#00FF33', '#660066', '#9900FF', '#CC0000',
                '#FF6633', '#FFCC00', '#FFFF00', '#FF0099']
            }

            var chart = new google.visualization.
            PieChart(document.getElementById('chart_div'));
            chart.draw(data, options);
        }
    </script>
</head>

<body>
    <div id="chart_div">
    </div>
</body>
</html>

```



Darstellung der Regierungszeiten der Bundeskanzler in einem Zeitstrahl (Bild 2)

ein Datum enthalten, Start und Ende der Regierungszeit, muss hier der Datentyp *date* gewählt werden. Anschließend können mit der Funktion *addRows* die Datensätze eingefügt werden:

```
dataTable.addRows([
  ['Konrad Adenauer', new Date(1949, 8, 15 ),
    new Date(1963, 9, 16 ) ],
  ...
  ['Angela Merkel', new Date(2005, 10, 22 ), new Date() ],
]);
```

In diesem Beispiel wurde auf eine Anpassung durch *options* verzichtet. Daher fehlt nur noch das Zeichnen des Diagramms und das Einbinden in ein `<div>`:

```
var chart = new google.visualization.Timeline(
  document.getElementById('timeline'));
chart.draw(dataTable);
...
<body>
  <div id="timeline" style="width: 900px;
    height: 500px;"></div>
</body>
```

Im letzten Beispiel wird die Arbeitslosenquote anhand eines Länderdiagramms dargestellt, bei dem die Farben für die einzelnen Bundesländer von Gelb (niedrigster Wert) bis Rot (höchster Wert) reichen (Bild 3). Der Aufbau der HTML-Seite ist auch hier gleich, daher wird nur auf die Unterschiede zum ersten Beispiel näher eingegangen. Für ein Länderdiagramm muss das Package *geochart* geladen werden. Die *DataTable* hat zwei Spalten, einen *string* für das Bundesland und eine *number* für die Arbeitslosenquote:

```
data.addRows([
  ["Baden-Württemberg", 4],
  ...
  [{f: 'Brandenburg', v: 'DE-BB'}, 9.6],
  ...
  ["Thüringen", 8]
]);
```

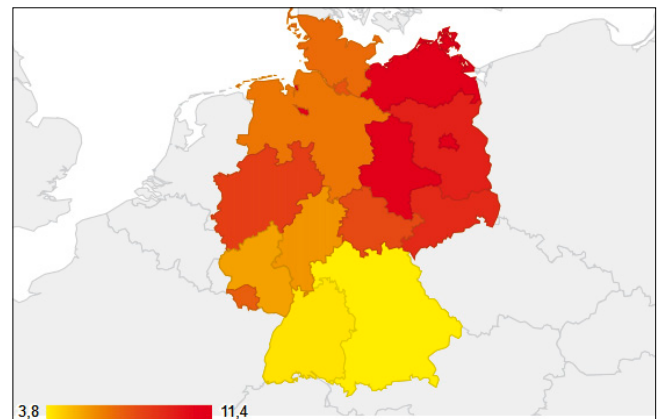
Die abweichende Darstellung für Brandenburg ist durch einen Fehler in den Google Chart Tools bedingt. Durch diese

Art der Angabe ist im Diagramm aber kein Fehler mehr zu sehen; ansonsten würde Brandenburg nicht erkannt und dementsprechend auch nicht farblich markiert werden.

Die *options* sind in diesem Beispiel wichtig, da hier der Schlüssel für Deutschland und die Einteilung in Bundesländer angegeben werden:

```
var options = {
  "title": "Arbeitslosenquote in Deutschland",
  region: 'DE',
  'resolution': 'provinces',
  colorAxis: {colors: ['yellow', 'red']},
  "width": 595,
  "height": 400};
```

Für Deutschland steht die *region*: 'DE' und für die Darstellung in Bundesländer die *resolution*: 'provinces'. Hier wird auch die Farbgebung in Gelb bis Rot angegeben. Das anschließende



Darstellung der Arbeitslosenquote anhand eines Länderdiagramms (Bild 3)

de Zeichnen des Diagramms und Einbinden in die HTML-Seite erfolgt wie in den vorigen Beispielen.

Fazit

Um einfache Diagramme in Ihre HTML-Seite einzubauen, haben Sie mit den Google Chart Tools ein leicht zu bedienendes Werkzeug in der Hand. Für nähere Informationen besuchen Sie die Developer-Seite von Google unter <https://developers.google.com/chart/interactive/docs/index>. ■



Bettina Zimmermann

ist Software-Entwicklerin im Java-Umfeld und seit 2002 in der IT-Branche tätig. Sie schreibt regelmäßig Artikel zu IT-Themen in verschiedenen Fachzeitschriften.

MVC IN DER IOS-ENTWICKLUNG

Model-View-Controller

Das MVC-Pattern gehört zu den wichtigsten Patterns in der iOS-Entwicklung.

App-Entwicklung ist ein spannendes und zugleich auch sehr kreatives Feld. Für so gut wie jede Problemstellung gibt es nicht nur die eine Lösung; oftmals führen viele verschiedene Wege zum Ziel. Doch wo es viele verschiedene Lösungsansätze gibt, sind nicht alle auch tatsächlich gut und fügen sich sauber in die Programmierung für eine spezifische Plattform ein.

Aus diesem Grund gibt es in der Softwareentwicklung allgemein die verschiedensten Design-Patterns, die beschreiben und festlegen, wie Code strukturiert und aufgebaut wird. Zu diesen Patterns gehört auch MVC (kurz für Model-View-Controller). Es spielt unter anderem in der iOS-Entwicklung eine essenzielle Rolle, nicht zuletzt aufgrund der Tatsache, dass Apple – der Hausherr dieser Plattform – selbst in seinen grundlegenden Frameworks auf MVC setzt und somit die gesamte Codebasis in der iOS-Entwicklung im Hinblick auf MVC entwickelt und designt wurde (Bild 1).

Warum MVC so wichtig ist

Bevor ich nun damit beginne, MVC in Bezug auf die iOS-Entwicklung im Detail vorzustellen, möchte ich kurz vorausschicken, warum MVC eigentlich so wichtig für uns iOS-Entwickler ist und warum sich jeder App-Entwickler damit im Detail auseinandersetzen sollte.

Wie im vorigen Abschnitt kurz erwähnt, sind Patterns wie MVC dazu gedacht, eine grundlegende gemeinsame Struktur für den eigenen Code zu definieren. Halten wir uns an derartige Patterns, hat das zur Folge, dass sich auch andere Entwickler, die ebenfalls mit diesen Patterns vertraut sind

und diese selbst anwenden, sich schneller im Code zurechtfinden und leichter nachvollziehen können, wie und warum etwas funktioniert und implementiert ist. Umgekehrt wird ebenfalls ein Schuh daraus, nämlich dann, wenn Entwickler derartige Design-Patterns ignorieren und ihren Code so schreiben und strukturieren, wie sie es selbst für richtig und sinnvoll halten (und dabei die Eigen- und Besonderheiten der jeweiligen Plattform außer Acht lassen). Das führt schnell zu unübersichtlichem Spaghetti-Code und erschwert es anderen Entwicklern, mit einem solchen Code zurechtzukommen.

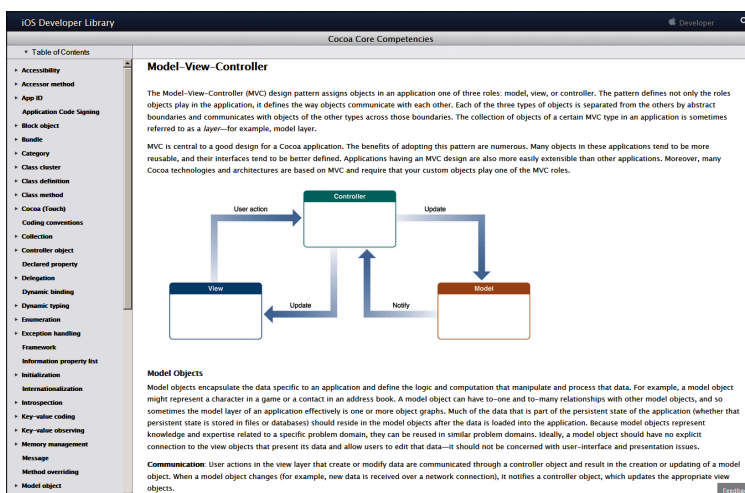
Sie sehen also schon: Patterns in der Software-Entwicklung sind wichtig. Und Apple setzt in iOS stark auf MVC, gerade auch bei all den hauseigenen Frameworks, mit denen iOS-Entwickler tagtäglich zu tun haben (Bild 2). Nur wer die Prinzipien und die Funktionsweise von MVC grundlegend versteht, begreift auch, wie bestimmte Funktionen in der iOS-Entwicklung arbeiten, und ist darüber überhaupt erst in der Lage, selbst sauberen und gut strukturierten Code für iOS zu schreiben.

Model-View-Controller

Nach diesen Vorworten aber nun ohne Umschweife weiter mit den Grundlagen des MVC-Pattern. Wie bereits beschrieben, stehen die drei Buchstaben für Model, View und Controller. Damit werden die Klassen eines jeden iOS-Projekts bezeichnet, die sich idealerweise genau einem dieser drei Elemente zuordnen lassen.

Als Model werden dabei die eigentliche Logik und die Datenhaltung einer App bezeichnet. Model-Klassen regeln Zugriffe auf das Dateisystem oder auf Datenbanken und beschreiben, wie die interne Logik einer App aussieht. Bei einer Adressbuch-App würde das Model beispielsweise beschreiben, welche Informationen zu einem Kontakt gespeichert werden können und wie die Kontakte intern in der App verwaltet werden.

Eine View ist das genaue Gegenstück zum Model. Während das Model sich um die nach außen hin nicht sichtbare Logik kümmert, dienen die Views zur Darstellung von Elementen auf der Oberfläche. Schaltflächen, Textansichten, Labels; all das sind Views, die wir in der iOS-Entwicklung verwenden können. Dabei zeichnet Views aus, dass diese keinerlei Information über die zugrunde liegende Logik einer App (also das Model) besitzen. Views regeln einzig und allein, wie etwas aussieht, nicht aber, welche Informationen sie anzeigen. Nehmen



MVC in der Entwickler-Dokumentation von Apple (Bild 1)

wir als Beispiel ein Label. Ein Label legt einfach nur fest, wie ein Text aussieht: Welche Schriftfarbe und Schriftart werden verwendet? Wie groß ist das Label? Das sind Dinge, die ein Label definiert. Welcher Text angezeigt wird, ist dem Label dabei egal.

Diese Information ist von Label zu Label unterschiedlich. Diese strikte Trennung erlaubt es uns, Labels immer und immer wieder in unterschiedlichen Projekten zu verwenden, ohne für jeden Inhalt auch eine eigene Label-Klasse erstellen zu müssen. Genau hier zeigt sich bereits der große Vorteil und der Schwerpunkt, der das MVC-Pattern ausmacht, doch dazu gleich mehr.

Bleibt zu guter Letzt noch der Controller. Der Controller kann dabei als Bindeglied zwischen View und Model verstanden werden. Controller bringen diese beiden Elemente zusammen und schließen somit die Lücke zwischen der reinen Logik einer App und deren Oberfläche.

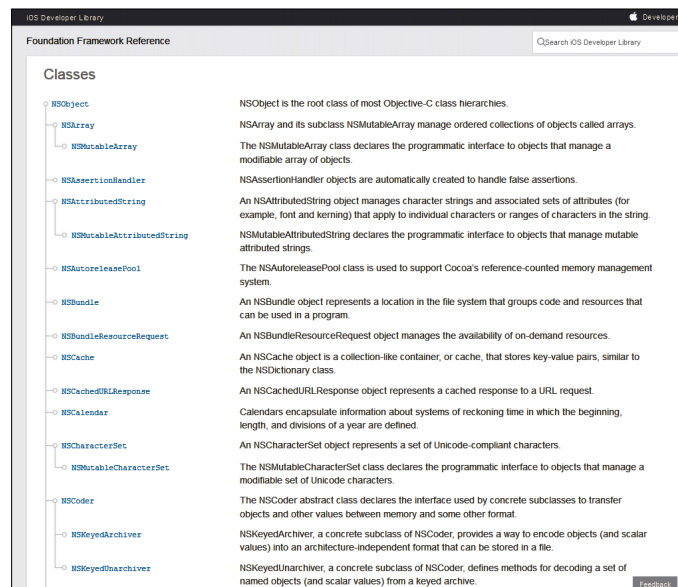
Betrachten wir dabei zur Verdeutlichung noch einmal das bereits genannte Beispiel einer Adressbuch-App: Ein Controller könnte darin alle Kontakte (das Model) auslesen und in einer Tabelle (der View) dem Nutzer anzeigen. Beide für sich betrachtet – ein Kontakt und eine Tabelle – funktionieren unabhängig voneinander und können so auch problemlos in andere Projekte übernommen werden. Kontakte können dann auch in einer App für ein soziales Netzwerk verwendet werden, eine Tabelle kann andere Inhalte anzeigen. Wie eben jenes Zusammenspiel genau aussieht, regelt ein Controller.

Wiederverwendbarkeit dank MVC

Dieses grundlegende Prinzip ist das wichtigste Paradigma bei der Arbeit mit MVC: Nur ein Controller erhält Zugriff auf Model und View. Model-Klassen sollen nur Zugriff auf andere Model-Klassen besitzen, genauso wie Views nur Zugriff auf andere View-Klassen haben sollen.

Diese strikte Trennung erlaubt es, Model- und View-Klassen leicht wiederzuverwenden und in verschiedenen Projekten für unterschiedliche Zwecke einzusetzen. Genau so ist es schließlich auch bei den verschiedenen View- und Model-Klassen, die Teil des Foundation- oder UIKit-Frameworks von Apple sind. Man stelle sich vor, man müsste für jeden Button eine eigene Klasse erstellen, in der festgelegt ist, was der Button tut.

Dank MVC ist das nicht nötig. Ein Button ist damit eine View, und was ein Button nun genau tut, legt ein zugeordneter Controller fest. Soll an einer anderen Stelle eine andere Aktion von einem Button aus ausgeführt werden, so wird ein



Auch das Foundation- und UIKit-Framework von Apple arbeitet nach dem MVC-Prinzip (Bild 2)

zweiter, passender Controller erstellt, der die gewünschte Aktion festlegt und implementiert; eine neue separate Button-Klasse ist dafür nicht notwendig, es kann dieselbe Button-Ansicht wie zuvor verwendet werden, und dennoch ist die durchzuführende Aktion eine andere.

Während somit die Model- und View-Klassen nach dem MVC-Pattern die Elemente sind, die problemlos in verschiedenen Projekten wiederverwendet werden können, sind die Controller der Teil des MVC-Codes, der am wenigsten an anderer Stelle wiederverwendet werden

kann. Das liegt daran, dass Controller sehr spezifischen Code enthalten und festlegen, wie Model und View verwendet werden. Das ist aber auch nicht weiter tragisch und ist durch das Pattern bedingt.

Kommunikation im MVC-Pattern

Nun haben wir sehr viel über das MVC-Pattern gelesen und erfahren und wissen, wofür es gut ist, wo seine spezifischen Vorteile liegen und warum es in der iOS-Entwicklung so wichtig ist. Bevor es nun an die Anwendung von MVC in der iOS-Programmierung geht, bleibt noch eine letzte wichtige Frage zu klären: Wie kommunizieren denn Controller-Klassen mit Model und View? Schließlich wissen wir jetzt, dass Model und View niemals direkt miteinander kommunizieren sollen und stattdessen hier der Controller als Bindeglied fungiert. Doch wie tut er das? Wie teilt eine View einem Controller mit, dass sie betätigt wurde? Und wie kann das Model den Controller beispielsweise darüber informieren, dass es neue Datensätze aus dem Internet geladen hat?

Hier kommen die verschiedenen Kommunikationswege des MVC-Patterns ins Spiel. Derer gibt es typischerweise vier:

- Eine View informiert ihren Controller über eine Aktion, die mit oder auf ihr ausgeführt wurde.
- Ein Model informiert seinen Controller über mögliche Änderungen in der Datenhaltung.
- Ein Controller hat Zugriff auf seine View und kann diese aktualisieren.
- Ein Controller hat Zugriff auf sein Model und kann dieses aktualisieren.

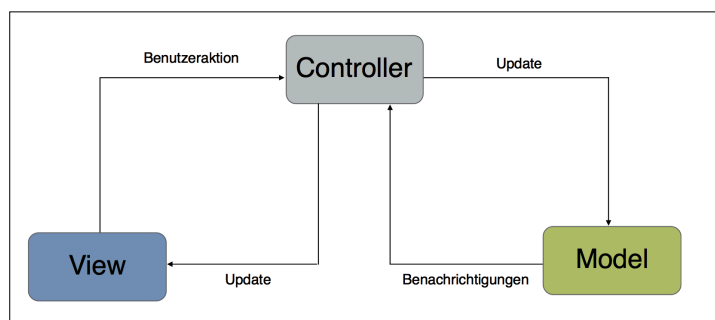
Diese kleine Übersicht zeigt erneut, wie essenziell wichtig der Controller für die Kommunikation zwischen View und Model ist. Ohne ihn würden nach MVC-Pattern die Objekte von View- und Model-Klassen niemals miteinander kommunizieren können. ►

Bild 3 verdeutlicht noch einmal die Kommunikationswege zwischen den drei Elementen Model, View und Controller des MVC-Patterns.

Bei MVC gibt es also nur zwei Arten von Kommunikationswegen: zwischen Controller und Model und zwischen Controller und View. Und genau diese beiden betrachten wir nun einmal im Detail.

Kommunikation zwischen Model und Controller

In der iOS-Entwicklung gibt es zwei Techniken, um die Kommunikation zwischen Model und Controller zu ermöglichen. Ich möchte dabei direkt vorausschicken, dass die genannten Techniken durchaus auch an anderer Stelle im Code eines iOS-Projekts verwendet und implementiert werden können, beispielsweise auch bei der Kommunikation zwischen mehreren verschiedenen Controllern oder zwischen verschiedenen Model-Objekten. Sie sind allerdings unabdingbar und essenziell für eben jene wichtige Kommunikation zwischen Model und Controller und müssen dort in jedem Fall verwendet werden.



Der Controller fungiert als Bindeglied zwischen Model und View (Bild 3)

Die erste Technik beschreibt eine iOS-typische Möglichkeit, Objekte über bestimmte Aktionen zu informieren. Es handelt sich dabei um die sogenannten Notifications. Eine Notification in iOS ist sehr einfach aufgebaut und verfügt mindestens über einen eindeutigen Titel, über den die Notification identifiziert werden kann.

Wird nun auf Controller- oder Model-Seite eine bestimmte Aktion ausgelöst, über die man auch andere Objekte eines iOS-Projekts informieren möchte, so sendet man zusammen mit der entsprechenden Aktion eine Notification. Alle anderen Objekte des Projekts können sich für eben jene Notification registrieren und werden damit informiert, sobald besagte Notification gesendet wurde. Sie haben dann die Chance, eigene weitere Aktionen auszuführen und entsprechend auf das Event zu reagieren, welches die besagte Notification gesendet hat.

Notifications in der Praxis

So weit einmal die Theorie von Notifications. In der Praxis spielen zwei Klassen des Foundation-Frameworks eine essenzielle Rolle bei der Arbeit mit Notifications: *NSNotification* und *NSNotificationCenter*.

Ein *NSNotification*-Objekt entspricht einer im vorherigen Abschnitt beschriebenen Notification. Es besteht aus mindestens einem eindeutigen Titel, über den die Notification im gesamten Projekt identifiziert werden kann. Darüber hinaus kann eine Notification zusätzlich auch aus einem optionalen Objekt und einem optionalen Dictionary mit weiteren Informationen bestehen. Diese Teile werden dann zusammen mit der Notification versendet und können von den Empfängern der Nachricht mit ausgewertet werden.

Das *NSNotificationCenter* ist als Singleton konzipiert, das heißt, dass es in der Regel nur eine Instanz dieser Klasse in einem iOS-Projekt gibt. Über dieses Singleton werden zwei Aufgaben geregelt: Einerseits werden über das Notification Center Notifications verschickt. Möchte also ein Objekt eine Notification senden, so wird diese dem *NSNotificationCenter* übergeben und dieses stellt die Nachricht im Projekt bereit. Auf der anderen Seite registrieren sich Objekte, die über den Eingang bestimmter Notifications informiert werden wollen, ebenfalls beim Singleton der Klasse *NSNotificationCenter* und geben dabei an, welche Methode ausgeführt werden soll, wenn die registrierte Notification tatsächlich gesendet wird.

Notifications erstellen und senden

Betrachten wir zunächst einmal das Erstellen und Senden von Notifications. Für diesen Zweck bringt die Klasse *NSNotification* einen passenden Designated Initializer mit, über den eine vollständig konfigurierte Instanz einer *NSNotification* erzeugt werden kann:

```
init(name name: String, object object: AnyObject?,
userInfo userInfo: [NSObject : AnyObject]?)
```

Der erste Parameter *name* entspricht dabei dem eindeutigen Namen der Notification. Dieser wird auch von allen Objekten, die über das Senden dieser Notification informiert werden möchten, für die zugehörige Registrierung verwendet. *object* ist ein optionaler Parameter, bei dem es sich um ein beliebiges Objekt handeln kann. Dieses Objekt wird gewissermaßen mit der Notification verschickt, was es Empfängern der Notification erlaubt, dieses Objekt auszulesen und für ihre Zwecke zu verwenden.

userInfo zu guter Letzt ist ebenfalls ein optionaler Parameter. Dieses Dictionary kann genutzt werden, um zusammen mit der Notification weitere beliebige Informationen zu verschicken. Es hängt von Fall zu Fall ab, ob es sinnvoll und angebracht ist, ein solches Dictionary zu übergeben oder nicht.

Neben diesem Designated Initializer verfügt die Klasse *NSNotification* noch über einen vereinfachten Convenience Initializer:

```
convenience init(name aName: String, object anObject:
AnyObject?)
```

Dieser nimmt lediglich den eindeutigen Namen und ein optionales Objekt für die Notification entgegen. Beim Initialisieren einer Notification über diesen Convenience Initializer

Listing 1: Erstellen und Senden einer Notification

```
let myNotification = NSNotification(name:
    "MyNotification", object: nil)
NSNotificationCenter.defaultCenter().
    postNotification(myNotification)
```

wird das *userInfo*-Dictionary des Designated Initializers also in jedem Fall auf *nil* gesetzt. Dieser Convenience Initializer lohnt sich also immer dann, wenn das *userInfo*-Dictionary nicht gebraucht wird. Eine simple Notification ohne *userInfo*-Dictionary lässt sich somit etwa wie folgt erzeugen:

```
let myNotification = NSNotification
(name: "MyNotification", object: nil)
```

Bleibt noch die Frage, wie nun eine solche Notification innerhalb einer App gesendet wird. Hier kommt das Singleton der Klasse *NSNotificationCenter* zum Einsatz. Es kann mit Hilfe der Klassenmethode *defaultCenter* ausgelesen werden:

```
class func defaultCenter() -> NSNotificationCenter
```

Bei der Arbeit mit *NSNotificationCenter* wird standardmäßig immer dieses Singleton verwendet, und darüber werden alle weiteren Aktionen des *NSNotificationCenter* ausgeführt. Um nun ein *NSNotification*-Objekt im System zu senden, bringt *NSNotificationCenter* eine passende Instanzmethode mit:

```
func postNotification(_ notification: NSNotification)
```

Die Methode *postNotification*: nimmt das zu sendende *NSNotification*-Objekt entgegen und sendet es umgehend in der gesamten App. Listing 1 zeigt einmal beispielhaft den kompletten Prozess.

Empfangen von Notifications

Wir wissen nun also, wie wir Notifications erstellen und senden können. Bleibt noch die Frage, wie nun ein Objekt, das sich für eine bestimmte Notification interessiert, auf diese »lauschen« und beim Senden eben jener Nachricht eine eigene Aktion durchführen kann.

Essenziell hierbei ist erneut das Singleton der Klasse *NSNotificationCenter*. Die Klasse bringt eine Methode mit, mit deren Hilfe ein sogenannter Observer für eine Notification erstellt wird. Dieser Observer legt fest, welches Objekt bei welcher Notification welche Methode aufrufen soll. Die Deklaration dieser Methode lautet wie folgt:

```
func addObserver(_ observer: AnyObject, selector
aSelector: Selector, name aName: String?,
object anObject: AnyObject?)
```

Interessiert sich nun ein Objekt für eine bestimmte Notification, kann es diese Methode nutzen, um sich selbst für eben

jene Notification zu registrieren. Betrachten wir dabei die Parameter dieser Methode einmal im Detail:

- *observer*: Der Observer ist das Objekt, das über das Eintreffen einer spezifischen Notification informiert werden möchte. Wenn Sie diese Methode in der Klasse des Objekts aufrufen, das selbst über eben jene Notification informiert werden möchte, übergeben Sie hier als Parameter *self*.
- *aSelector*: Der Selector entspricht dem Namen der Methode, die bei Eingang der Notification auf dem als *observer* definierten Objekt aufgerufen werden soll. In Swift werden Observer als einfache Strings definiert. Dabei ist unbedingt darauf zu achten, den korrekten Methodennamen zu schreiben, da der Compiler durch dieses Verfahren keine Chance hat, die Zeichenkette gegen einen vorhandenen Methodennamen zu prüfen. Schleicht sich in der Zeichenkette ein Fehler ein und die entsprechende Methode existiert gar nicht, führt dies zum Absturz der App, sobald die entsprechende Notification gesendet wird.
- *aName*: Dieser Parameter entspricht dem Namen der Notification, über deren Senden der *observer* informiert werden soll. Hierbei mag es vielleicht verwundern, warum ausgerechnet dieser Parameter – der ja gerade als essenziell wichtig erscheint – als optional deklariert ist. Sollte für *aName* *nil* übergeben werden, so bedeutet das einfach, dass die in *aSelector* definierte Methode des *observer* bei jeder gesendeten Notification ausgeführt wird und nicht nur bei der spezifisch als *aName* definierten.
- *anObject*: Möchte man nur über das Senden von Notifications von einem ganz spezifischen Objekt informiert werden, so kann man dieses als *anObject* übergeben. Der aufzurufende Selector des *observer* wird sodann nur ausgeführt, wenn die gewünschte Notification von eben jenem Objekt aus gesendet wurde. Wird die Notification von einem anderen Objekt gesendet, greift der Selector nicht und wird auch nicht ausgeführt. In der Regel wird hier oftmals *nil* übergeben, was bedeutet, dass der Selector des *observer*-Objekts immer ausgeführt werden soll, wenn die gewünschte Notification gesendet wird; ganz gleich von welchem Objekt das Senden ausgeht.

Sollen nun Objekte einer Klasse über das Senden einer bestimmten Notification informiert werden, so müssen diese sich an einer bestimmten Stelle im Code für eben jene Notification mit der genannten Methode registrieren. Typische Stellen für die Registrierung für Notifications sind beispielsweise Initializer oder – bei View-Controllern – die *viewDidLoad*-Methode. Eine einfache beispielhafte Registrierung für eine Notification mit dem Namen *MyNotification* kann wie folgt aussehen:

```
NSNotificationCenter.defaultCenter().addObserver(self,
selector: "doSomething:", name: "MyNotification",
object: nil)
```

In diesem Beispiel wird die Methode *doSomething*: aufgerufen, sobald an irgendeiner beliebigen Stelle im Code die Notification mit dem Namen *MyNotification* gesendet wird. ►

Zu beachten ist dabei der Parameter, den die Methode *doSomething*: erwartet (zu erkennen am Doppelpunkt am Ende des Selectors). Dieser Parameter ist vom Typ *NSNotification*, was bedeutet, dass dieser Methode beim Aufruf das zugehörige *NSNotification*-Objekt übergeben wird, das beim Senden eben jeder Notification verwendet wurde.

Das erlaubt es dieser Methode, weitere Informationen der Notification wie das optional mitgelieferte Objekt oder das *userInfo*-Dictionary auszulesen. Die Klasse *NSNotification* bringt für diesen Zweck drei Read-only-Properties mit, um auf alle Informationen einer Notification zugreifen zu können:

- Den Namen der Notification.
- Das optional mit der Notification mitgelieferte Objekt.
- Das optional mitgelieferte *userInfo*-Dictionary.

Die Beispielmethode *doSomething*: kann in ihrer Implementierung auf all diese Informationen dank ihres Parameters zugreifen und diese auswerten und verwenden:

```
func doSomething(notification: NSNotification) {
    print("Name: \(notification.name)")
    print("Object: \(notification.object)")
    print("User info: \(notification.userInfo)")
}
```

Der Name des Parameters in der Methodendeklaration von *doSomething*: spielt dabei keine Rolle; Sie können jeden beliebigen Namen verwenden (in diesem Beispiel ist der Parametername auf *notification* gesetzt). Das Notification-System von iOS übergibt die zugrunde liegende Notification an eben jenen Parameter, den Sie dann innerhalb der Methode verwenden und auswerten können.

Optional können Sie auch auf die Übergabe eines *NSNotification*-Parameters beim Selector verzichten. In diesem Fall fällt der Doppelpunkt am Ende des Selectors weg, der auf einen Parameter hinweist. Dabei ist zusätzlich noch zu beachten, dass die zugehörige Methode selbst dann ebenfalls über keinem Parameter verfügen darf.

Observer entfernen

Sie kennen nun die wichtigsten Klassen und Methoden sowie die Funktionsweise von Notifications. Zu guter Letzt möchte ich noch auf einen weiteren wichtigen Punkt hinweisen, der nicht umgehend ins Auge springt.

Registriert sich ein Objekt mittels der im vorigen Abschnitt vorgestellten Methode *addObserver:selector:name:object:* für den Empfang einer Notification, so wird diese Information im Singleton des *NSNotificationCenter* gespeichert. Diese Information bleibt auch dann noch vorhanden, wenn das Objekt, das sich für eine Notification registriert hat, gar nicht mehr existiert und aus dem Speicher entfernt wurde. Dieses Verhalten hat aber zur Folge, dass das Notification Center beim Senden eben jener Notification trotzdem versucht, auch das bereits aus dem Speicher entfernte Objekt zu informieren und aufzurufen. Das führt zu einem Absturz der App, schließlich existiert das gewünschte Objekt nicht mehr.

Listing 2: Vollständiges Notification-Beispiel

```
class Person {
    let firstName: String
    let lastName: String
    init(firstName: String, lastName: String) {
        self.firstName = firstName
        self.lastName = lastName
    }
}

class Company {
    var employees = [Person]()
    func addEmployee(employee: Person) {
        employees.append(employee)
        let addedEmployeeNotification =
            NSNotification(name: "AddedEmployee",
                object: employee)NSNotificationCenter.
                defaultCenter().postNotification
                (addedEmployeeNotification)
    }
}

class EmployeeViewController: NSObject {
    override init() {
        super.init()
        NSNotificationCenter.defaultCenter().
            addObserver(self, selector:
                "addedEmployee:", name: "AddedEmployee",
                object: nil)
    }
    func addedEmployee(notification: NSNotification)
    {
        // Reload view.
    }
    deinit {
        NSNotificationCenter.defaultCenter().
            removeObserver(self, name: "AddedEmployee",
                object: nil)
    }
}
```

Daher ist es wichtig, dass Objekte, die sich für Notifications registrieren, sich auch genauso wieder vom Notification Center abmelden, sobald sie sich nicht mehr für den Eingang eben jener Notifications interessieren. Typischerweise ist das spätestens dann der Fall, wenn das betroffene Objekt aus dem Speicher entfernt wird.

Zu diesem Zweck verfügt *NSNotificationCenter* über eine passende Methode zum Entfernen eines Observers:

```
func removeObserver(_ observer: AnyObject, name aName:
    String?, object anObject: AnyObject?)
```

Die drei Parameter kennen wir dabei bereits aus der Methode *addObserver:selector:name:object:*. *observer* ist das Objekt, das sich für eine Notification registriert hat. *aName* ent-

spricht dem Namen der Notification, für die sich der *observer* registriert hat. Und *anObject* schließlich entspricht einem optionalen Objekt, falls man Notifications nur von eben jenem Objekt wünscht.

Diese Methode wird dann äquivalent zur Methode `addObserver(selector:name:object:)` aufgerufen, beispielsweise wie folgt:

```
NSNotificationCenter.defaultCenter().
removeObserver(self, name: "MyNotification", object:
nil)
```

Zum Abschluss des Notification-Themas führe ich noch ein komplettes Beispiel auf, das kurz und knapp die Funktionsweise von Notifications demonstriert und einen typischen Anwendungsfall zeigt ([Listing 2](#)).

Ein paar Worte zur Erklärung des Beispiels: Die Klasse *Company* verschickt Notifications, wann immer ein Mitarbeiter in Form eines *Person*-Objekts über die Methode `addEmployee:` hinzugefügt wird. Diese Notification trägt den Namen *AddedEmployee* und übergibt die hinzugefügte *Person*-Instanz als *Notification*-Objekt. Über ein *userInfo*-Dictionary verfügt die Notification nicht.

Neben diesen *Person*- und *Company*-Klassen enthält das Beispiel noch eine weitere Klasse namens *EmployeeViewController*, bei dem es sich der Einfachheit halber nicht wirklich um einen View-Controller, sondern um eine Subklasse von *NSObject* handelt.

Im Initializer der Klasse wird der Observer für die Notification *AddedEmployee* gesetzt. Geht diese Nachricht ein, wird die Methode `addedEmployee:` aufgerufen, die als Parameter die zugehörige Notification erhält. Diese Methode könnte nun beispielsweise dazu dienen, eine Liste der Mitarbeiter neu zu laden und zu aktualisieren.

Zu guter Letzt wird der gesetzte Observer bei der Deinitialisierung eines Objekts der Klasse *EmployeeViewController* wieder entfernt.

Key-Value Observing

Notifications sind die eine Möglichkeit, eine Kommunikation zwischen Model und Controller herzustellen. Die andere dazugehörige Technik in der iOS-Entwicklung nennt sich Key-Value Observing (kurz KVO).

Während bei Notifications zwei Stellen aktiv werden (die eine sendet eine Nachricht, die andere lauscht darauf), ist das beim Key-Value Observing anders. Hier registriert sich ein Objekt auf eine bestimmte Property eines anderen Objekts und wird informiert, sobald sich eben jene Property ändert. Im Vergleich zu den Notifications fällt somit die explizite Sendestelle weg. Damit Key-Value Observing funktioniert, ist es wichtig, dass alle zugehörigen Klassen von *NSObject* erben, denn diese Klasse aus dem Foundation-Framework bringt die grundlegende Funktionalität für KVO mit. Ohne *NSObject* kann KVO nicht verwendet werden.

Ähnlich wie bei Notifications registriert sich ein Objekt bei KVO, um über Änderungen informiert zu werden und darüber eigene Aktionen durchzuführen.

Dabei definiert nicht *NSObject* selbst die für Key-Value Observing nötigen Methoden. Stattdessen sind diese im Protokoll *NSKeyValueObserving* definiert, zu dem *NSObject* konform ist.

Änderung einer Property

Möchte nun ein Objekt einer Klasse über die Änderungen einer Property eines anderen Objekts informiert werden, so muss sich dieses als Observer registrieren. Dazu dient die im folgenden Beispiel dargestellte Methode des *NSKeyValueObserving*-Protokolls:

```
func addObserver(_ observer: NSObject, forKeyPath
keyPath: String, options options:
NSKeyValueObservingOptions, context context:
UnsafeMutablePointer<Void>)
```

Diese Methode wird auf dem Objekt aufgerufen, von dem eine Property überwacht werden soll. Damit diese Methode auf eben jenem Objekt aufgerufen werden kann, ist es wichtig, dass jenes Objekt in jedem Fall von *NSObject* erbt; nur so steht diese Methode zur Verfügung.

Betrachten wir dabei die einzelnen Parameter einmal im Detail:

- Beim *observer* handelt es sich um das Objekt, das über Änderungen des Objekts, über das die Methode aufgerufen wird, informiert werden möchte.
- *keyPath*: Der *keyPath* entspricht dem Namen der Property, die auf dem Objekt, über das die Methode aufgerufen wird, überwacht werden soll. Diese Property wird als einfacher String übergeben. Dabei können auch mehrere Properties über den *keyPath* miteinander durch Trennung per `.` verketet werden; dazu später mehr.
- *options*: Die Optionen definieren, welche Informationen bei einer Änderung der zu überwachenden Property an den *observer* übergeben werden. Zwei typische Optionen sind `.New` und `.Old`, die festlegen, dass bei einer Änderung der neue Wert (`.New`) beziehungsweise der vorherige Wert (`.Old`) mit übergeben werden.
- *context*: Der Context dient zur Identifizierung einer spezifischen Property-Überwachung. Das liegt daran, dass ein Objekt ja mehrere verschiedene andere Objekte und Properties überwachen könnte. Wenn es nun zu einer Änderung kommt, müssen Sie irgendwie feststellen, welche der überwachten Objekte und Properties sich nun geändert hat, und dazu dient der *context*.

Äquivalent zu der Methode `addObserver(forKeyPath:options:context:)` verfügt das *NSKeyValueObserving*-Protokoll über eine weitere Methode, um gesetzte Observer wieder zu entfernen.

Gesetzte Observer

Das Prinzip ist dabei dasselbe wie bei den Notifications: Gesetzte Observer müssen an geeigneter Stelle auch wieder entfernt werden, damit nicht nach Freigabe der entsprechenden Objekte ein möglicher Aufruf erfolgt, der ins Leere ►

geht und somit zu einem Absturz führt. Die Methode `removeObserver(forKeyPath:context:)` ist folgendermaßen im Protokoll definiert:

```
func removeObserver(_ observer: NSObject, forKeyPath
keyPath: String, context context:
UnsafeMutablePointer<Void>)
```

Die Parameter `observer`, `keyPath` und `context` sind dabei äquivalent zu den gleichnamigen Parametern aus der Methode `addObserver(forKeyPath:options:context:)`.

Der context-Parameter

Wer sich die Deklaration des `context`-Parameters in den Methoden `addObserver(forKeyPath:options:context:)` und `removeObserver(forKeyPath:context:)` etwas genauer angesehen hat, war womöglich vom Typ dieses Parameters etwas irritiert. Dabei handelt es sich nämlich um ein Objekt vom Typ `UnsafeMutablePointer<Void>`. Doch was ist das eigentlich für ein Typ?

Einfach gesagt handelt es sich dabei um einen Verweis auf eine Speicherstelle, die zur eindeutigen Identifizierung des Observers dient. Laut Swift-Dokumentation lässt sich ein solches `context`-Objekt aber sehr einfach erstellen. Es reicht dabei, eine einfache Variable zu erzeugen und dieser einen Integer zuzuweisen.

Damit wird eine Speicherstelle für diese Variable reserviert, und genau diese wird dann als Parameter für `context` verwendet. Dabei ist zu beachten, für `context` die Speicheradresse und nicht das Objekt selbst zu übergeben. Das wird mithilfe des `&`-Operators bewerkstelligt, der dem übergebenen Parameter vorangestellt wird. Wie das konkret aussieht, sehen wir gleich in einem vollständigen Beispiel zu Key-Value Observing.

Bevor wir nun den Einsatz der Methoden des `NSKeyValueObserving`-Protokolls in der Praxis betrachten, sei zuvor noch ein weiteres wichtiges Detail vorangestellt: Properties, die Sie mittels KVO überwachen möchten, müssen zwingend mit dem Schlüsselwort `dynamic` gekennzeichnet werden. Nur dann ist eine Überwachung der entsprechenden Property möglich. Das folgende Beispiel zeigt eine derartige Deklaration:

```
dynamic var propertyToObserve = "Property to observe"
```

Ebenso wichtig ist – wie bereits beschrieben –, dass die Klasse, die über mindestens eine zu überwachende Property verfügt, auch in jedem Fall von der Klasse `NSObject` erbt, damit KVO verwendet werden kann.

Abfangen von Änderungen mittels KVO

Zum Abschluss der KVO-Funktionsweise bleibt noch eine Frage zu klären: Wie wird man über Änderungen der zu überwachenden Properties informiert? Schließlich geben wir im Gegensatz zu Notifications keinerlei Methoden beim Setzen der Observer an, die festlegen, welche Aktionen bei Änderung der zu überwachenden Properties ausgeführt werden.

Hier kommt eine weitere Methode des `NSKeyValueObserving`-Protokolls ins Spiel, die im Folgenden deklariert ist:

```
func observeValueForKeyPath(_ keyPath: String?, ofObject
object: AnyObject?, change change: [String : AnyObject]?,
context context: UnsafeMutablePointer<Void>)
```

Diese Methode muss in den Klassen überschrieben werden, die Properties überwachen und über entsprechende Änderungen informiert werden möchten. Diese Methode ist die zentrale Anlaufstelle, die dann bei Änderung eben jener überwachten Properties automatisch vom System aufgerufen

Listing 3: Vollständiges KVO-Beispiel

```
class Person: NSObject {
    let firstName: String
    let lastName: String
    init(firstName: String, lastName: String) {
        self.firstName = firstName
        self.lastName = lastName
    }
}

class Company: NSObject {
    dynamic var employees = [Person]()
    func addEmployee(employee: Person) {
        employees.append(employee)
    }
}

class EmployeeViewController: NSObject {
    let company = Company()
    private var companyContext = 0
    override init() {
        super.init()
        company.addObserver(self, forKeyPath:
            "employees", options: .New, context:
            &companyContext)
    }
    override func observeValueForKeyPath(keyPath:
        String?, ofObject object: AnyObject?, change:
        [String : AnyObject]?, context:
        UnsafeMutablePointer<Void>) {
        if context == &companyContext {
            // Reload view.
        } else {
            super.observeValueForKeyPath(keyPath,
                ofObject: object, change: change,
                context: context)
        }
    }
    deinit {
        company.removeObserver(self, forKeyPath:
            "employees", context: &companyContext)
    }
}
```

wird. Und darin gilt es, die gewünschten Aktionen durchzuführen, äquivalent zu den definierten Selectors bei Notifications.

Bei Überschreiben dieser Methode ist eine sehr wichtige Sache zu beachten: Diese Methode wird auch von Systemkomponenten von Apple verwendet. Daher gilt es bei Aufruf dieser Methode zu prüfen, ob diese tatsächlich wegen einer von uns überwachten Property aufgerufen wurde oder nicht. In letzterem Fall liegt es dann an uns, dieselbe Methode auf der Superklasse mittels *super* aufzurufen. Tun wir das nicht, kann das zu Systeminstabilitäten bis hin zu Abstürzen der eigenen App führen. Dazu nutzen wir das übergebene *context*-Objekt der Methode *observeValueForKeyPath:ofObject:change:context:*, die wir mit dem *context*-Objekt abgleichen, das wir bei der Registrierung der Observer verwendet haben.

Das *change*-Dictionary enthält darüber hinaus Informationen zur Änderung der überwachten Property. Welche Informationen das Dictionary enthält, ist davon abhängig, welche Optionen beim Observer für den Parameter *options* gesetzt wurden.

KVO in der Praxis

Manches mag bisher noch recht kryptisch klingen, daher betrachten wir einmal ein konkretes Beispiel zu KVO, das alle bisher genannten Aspekte des Key-Value Observing im Detail beleuchtet. Listing 3 zeigt das vorige Beispiel, umgesetzt mit Hilfe von KVO.

Betrachten wir einmal einige Details dieser Implementierung. Die Klasse *EmployeeViewController* definiert nun eine eigene Property *company* vom Typ *Company* und ein Context-Objekt namens *companyContext*; dieses wird für die eindeutige Identifizierung des Observers benötigt. Ziel ist es – wie im vorigen Notifications-Beispiel –, über Änderungen des *employees*-Array informiert zu werden. Zu diesem Zweck registriert sich die Klasse *EmployeeViewController* in ihrem Initializer als Observer für die eigene *company*-Property. Da es sich bei der zu überwachenden Property des *Company*-Typs um *employees* handelt, wird diese als *keyPath*-Parameter in Form eines Strings übergeben. Als Context wird die Property *companyContext* verwendet, deren Adresse mit Hilfe des *&*-Operators übergeben wird. Innerhalb des Deinitializers wird dann noch die passende Methode zum Entfernen des gesetzten Observers aufgerufen.

Um nun mögliche Änderungen der Property *employees* des *company*-Objekts abzufangen, muss innerhalb der Klasse *EmployeeViewController*, welche als Observer fungiert, die Methode *observeValueForKeyPath:ofObject:change:context:* überschrieben werden. Deren Implementierung prüft das übergebene *context*-Objekt gegen die Speicheradresse der *companyContext*-Property. Ist diese Prüfung erfolgreich, wissen wir, dass sich die Property *employees* geändert hat, und wir können entsprechend darauf reagieren (beispielsweise durch Aktualisierung der Ansicht, um alle aktuellen Mitarbeiter anzuzeigen).

Sollte die Prüfung nicht erfolgreich sein, wissen wir ebenso, dass diese Methode aufgrund einer anderen Änderung, die vom System ausging, aufgerufen wurde. In diesem Fall

Listing 4: Beispiel-Klassen zur Nutzung des Key-Path

```
class Person: NSObject {
    let firstName: String
    let lastName: String
    dynamic var tasks: [String] = [String]()
    init(firstName: String, lastName: String) {
        self.firstName = firstName
        self.lastName = lastName
    }
}

class Company: NSObject {
    let name: String
    var ceo: Person
    init(name: String, ceo: Person) {
        self.name = name
        self.ceo = ceo
    }
}

class CompanyViewController: NSObject {
    var company = Company(name: "MyCompany", ceo:
        Person(firstName: "Thomas", lastName:
            "Sillmann"))
    private var companyContext = 0
    override init() {
        super.init()
        company.addObserver(self, forKeyPath:
            "ceo.tasks", options: .New, context:
                &companyContext)
    }
    override func observeValueForKeyPath(keyPath:
        String?, ofObject object: AnyObject?, change:
            [String : AnyObject]?, context:
                UnsafeMutablePointer<Void>) {
        if context == &companyContext {
            // Reload view.
        } else {
            super.observeValueForKeyPath(keyPath,
                ofObject: object, change: change,
                    context: context)
        }
    }
    deinit {
        company.removeObserver(self, forKeyPath:
            "ceo.tasks", context: &companyContext)
    }
}
```

müssen wir sicherstellen, dann die entsprechende Methode der Superklasse mittels *super* aufzurufen.

Ebenfalls beachtet werden muss die Deklaration der zu überwachenden Property *employees* in der Klasse *Company* als *dynamic*, da nur dann eine Überwachung möglich ist.

Zum Abschluss der Vorstellung von KVO möchte ich noch einmal auf den Key-Path eingehen, der den Pfad zur überwachenden Property beschreibt. Im vorigen Beispiel war die- ►

ser Key-Path sehr einfach, da er sich direkt auf die Property *employees* der zu überwachenden Klasse *Company* bezog. Doch mit Hilfe des Key-Path können auch ganze Pfade von Properties aneinandergekettet werden. Listing 4 zeigt dazu ein abgewandeltes Beispiel.

Die Klasse *Company* verfügt nun noch über eine weitere Property *person*, die wiederum die neue Property *tasks* besitzt. Die Klasse *CompanyViewController* überwacht die Property *tasks* und greift dazu über den Key-Path zu, indem sie sich von der Property *ceo* zu *tasks* durchhangelt.

In einem solchen Fall ist es auch ausreichend, wenn die finale zu überwachende Property (also in diesem Beispiel *tasks*) als *dynamic* deklariert wird. Die vorigen Properties des Key-Path müssen nicht zwingend als *dynamic* deklariert werden.

Vor- und Nachteile von Notifications und KVO

Sie haben nun die beiden typischen Techniken zur Kommunikation zwischen Model und Controller in der iOS-Entwicklung kennengelernt. Bleibt die Frage: Wann sollten Sie welche der beiden Techniken am besten einsetzen?

Wann immer Sie sich innerhalb einer Klasse für die Werte von Properties anderer Objekte interessieren, die Sie innerhalb dieser Klasse selbst referenzieren, ist KVO sicherlich zunächst die beste Lösung. Dann nämlich stehen Ihnen alle Informationen und Objekte zur Verfügung, die Sie benötigen, um einen passenden Observer zu registrieren und Änderungen an den gewünschten Properties abzufangen.

Notifications hingegen streuen weiter. Selbst wenn sie keine Referenz auf ein Objekt besitzen, das eine für Sie interessante Notification sendet, können Sie diese dennoch abfangen und entsprechende Aktionen durchführen. Generell ist das Verwenden und Senden von Notifications immer dann sinnvoll, wenn eine Aktion ausgeführt wird, die weitreichenden Einfluss in einem Projekt besitzt. Auch wenn es viele Stellen in einem Projekt gibt, die sich für eine bestimmte Aktion interessieren, ist es einfacher, eine Notification zu senden und sie an den gewünschten Stellen abzufangen, als an jeder benötigten Stelle mittels KVO die zugehörigen Eigenschaften abzufragen.

Zu guter Letzt erlauben es Notifications, weitere Informationen wie ein Objekt und ein User-Info-Dictionary zu senden. Sollte das auch in Ihrem Fall nötig sein, ist sehr wahrscheinlich die Nutzung von Notifications der von KVO vorzuziehen. Außerdem können Notifications an jeder beliebigen Stelle gesendet werden. KVO hingegen erlaubt nur die Überwachung von Properties.

Kommunikation zwischen View und Controller

Nachdem wir nun wissen, wie Model und Controller in einer iOS-App miteinander kommunizieren können, bleibt noch die Frage, welche Möglichkeiten der Kommunikation es typischerweise zwischen View und Controller gibt. Auch hier stehen zwei verschiedene Techniken zur Verfügung: Target-Action und Delegation. Beide Verfahren sind sehr unterschiedlich und haben auch verschiedene Einsatzgebiete.

Betrachten wir dabei zunächst einmal Target-Action. Wie der Name bereits andeutet, gibt es bei dieser Technik zwei

Listing 5: Button-Klasse mit Target-Action

```
class Button: NSObject {
    var target: AnyObject
    var action: Selector
    init(target: AnyObject, action: Selector) {
        self.target = target
        self.action = action
    }
    func performAction() {
        target.performSelectorOnMainThread(action,
            withObject: self, waitUntilDone: true)
    }
}
```

essenzielle Bestandteile. Das Target verweist auf ein Objekt, das eine View zu einem bestimmten Zeitpunkt aufruft, um eine Aktion in Form einer Methode auszuführen; die sogenannte Action. Nehmen wir als Beispiel einen Button, ein typisches View-Element. Wenn ein Button betätigt wird, soll eine bestimmte Aktion ausgeführt werden. Welche Aktion das ist, hängt davon ab, wo der Button definiert ist und was er tun soll; schließlich hat so ziemlich jeder Button eine andere Aktion auszuführen.

Damit ein spezifischer Button nun weiß, was er bei einer Betätigung tun soll, werden ihm zwei Informationen übergeben: Einmal erhält der Button eine Referenz auf ein Objekt (typischerweise ein Controller), der den Code enthält, der bei Betätigung ausgeführt werden soll. Dieser Code, der typischerweise in einer eigenen Methode implementiert ist, erhält den Button ebenfalls in Form eines Selectors. Wird der Button nun betätigt, ruft er schlicht und einfach aus seinem Target (dem zugewiesenen Controller) die Action (die Methode mit der gewünschten auszuführenden Implementierung) auf. Damit ändert ein Button ganz schnell sein Verhalten, abhängig davon, welches Target und welche Action ihm zugewiesen werden.

Target-Action in der Praxis

Um die praktische Verwendung von Target-Action zu verstehen, sehen wir uns einmal ein konkretes Beispiel dazu näher an. Zu diesem Zweck soll eine eigene vereinfachte *Button*-Klasse erstellt werden, die in ihrem Initializer ein passendes Target mitsamt passender Action erhält und diese Informationen in Properties speichert. Wird dann eine bestimmte Methode der *Button*-Klasse aufgerufen, greift die Klasse auf ihr Target zu und ruft darauf die gespeicherte Action auf. Das Grundgerüst dieser Klasse zeigt Listing 5.

Bei *target* handelt es sich um eine Property vom Typ *AnyObject*, schließlich kann es sich dabei ja um jedes beliebige Objekt handeln; der genaue Typ braucht uns dabei nicht zu interessieren. *action* hingegen ist vom Typ *Selector*. Dieser Typ dient in Swift zur Abbildung von Methodennamen für das Target-Action-Pattern. Der Initializer der Klasse nimmt für beide Properties einen passenden Wert entgegen.

Interessant wird es in der Methode *performAction*. Diese Methode dient beispielsweise für das Betätigen unseres Buttons. Wann immer der Button betätigt wird, soll nun die übergebene Action-Methode auf dem übergebenen Target ausgeführt werden.

Zu diesem Zweck steht uns unter anderem die Methode *performSelectorOnMainThread:withObject:waitUntilDone:* der Klasse *NSObject* zur Verfügung. Die genaue Deklaration dieser Methode habe ich im Folgenden aufgeführt:

```
func performSelectorOnMainThread
(_ aSelector: Selector, withObject arg: AnyObject?,
waitUntilDone wait: Bool)
```

Zunächst einmal erwartet die Methode einen Selector, also den Namen einer Methode, den sie auf dem Objekt, das *performSelectorOnMainThread:withObject:waitUntilDone:* aufruft, ausführt. Darüber hinaus lässt sich dieser Methode noch ein optionales Objekt übergeben, das sie an die auszuführende Methode des Targets als Parameter mitsendet. Zu guter Letzt legt der finale Parameter *wait* noch fest, ob die Ausführung des nachfolgenden Codes verhindert werden soll, bis die entsprechende Aktion erfolgreich beendet wurde (*true*) oder nicht (*false*).

Listing 6: Verwenden von Target-Action

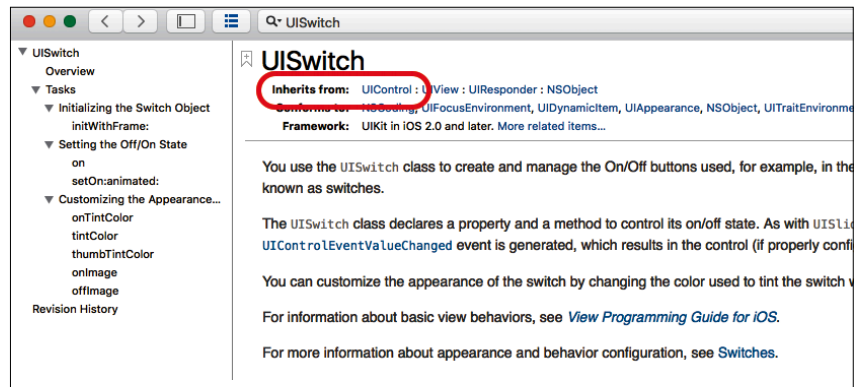
```
class MyFirstViewController: NSObject {
    func myFirstViewControllerAction(sender: Button) {
        print("MyFirstViewController action.")
    }
}

class MySecondViewController: NSObject {
    func mySecondViewControllerAction
(sender: Button) {
        print("MySecondViewController action.")
    }
}

let myFirstViewController = MyFirstViewController()
let mySecondViewController =
MySecondViewController()

let firstButton = Button(target:
myFirstViewController, action:
"myFirstViewControllerAction:")
firstButton.performAction()
// MyFirstViewController action.

let secondButton = Button(target:
mySecondViewController, action:
"mySecondViewControllerAction:")
secondButton.performAction()
// MySecondViewController action.
```



Erbt eine View-Klasse aus UIKit von UIControl, so unterstützt sie Target-Action (Bild 4)

Wann immer nun also auf einem *Button*-Objekt die Methode *performAction* aufgerufen wird, wird die entsprechende Action-Methode des übergebenen Target ausgeführt. Jedes neue Target und jede neue Action sorgt also dafür, dass ein *Button*-Objekt trotz unverändertem Code immer gänzlich verschiedene Aktionen ausführt.

Verwenden von Target-Action

Im Prinzip ist mit der eben gezeigten Implementierung der *Button*-Klasse alles getan, was zur Vorbereitung von Target-Action nötig ist. Wie nun eine mögliche Verwendung aussehen kann, zeigt Listing 6.

Dort werden zwei neue Klassen mit je einer Methode definiert. Der Einfachheit halber geben diese lediglich ein Print-Statement aus. Anschließend wird von diesen beiden Klassen je eine Instanz erstellt und diese in einer Konstanten gespeichert.

Dann erfolgt die Erstellung zweier *Button*-Instanzen (*firstButton* und *secondButton*). *firstButton* wird dabei als Target *myFirstViewController* sowie als Action die einzige Methode der Klasse *myFirstViewControllerAction:* übergeben. Bei *secondButton* wird äquivalent dazu *mySecondViewController* mit *mySecondViewControllerAction:* verwendet. Wie Sie sehen, werden dabei Selectors in Swift als einfache Strings übergeben. Zu beachten ist auch hier der Doppelpunkt am Ende des Methodennamens, da beide Methoden einen Parameter entgegennehmen (nämlich die Instanz der *Button*-Klasse, die die Methode aufruft).

Je nachdem, auf welcher der beiden *Button*-Instanzen nun die Methode *performAction* aufgerufen wird, erhält man ein unterschiedliches Ergebnis. Bei *firstButton* wird die Methode *myFirstViewControllerAction:* der Klasse *MyFirstViewController* aufgerufen, bei *secondButton* hingegen ist es die Methode *mySecondViewControllerAction:* der Klasse *MySecondViewController*. Obwohl sich der Code der Klasse *Button* nicht verändert, führt sie aufgrund des geänderten Targets und der geänderten Action jeweils eine andere Aktion durch – genau das ist das Prinzip und der große Vorteil von Target-Action.

Einen konkreten Fall für die Verwendung von Target-Action in der iOS-Entwicklung haben wir bereits im vorangegangenen Beispiel kennengelernt: Schaltflächen sind ide- ►

ale Views für den Einsatz von Target-Action, da sie generell nur eine Aktion verstehen (bei Schaltflächen eben das Betätigen derselbigen). So setzt auch die Klasse *UIButton* aus dem UIKit-Framework auf Target-Action. Andere Beispiele für Target-Action aus dem UIKit-Framework sind die Klassen *UISlider* oder *UISwitch*.

Wenn Sie wissen möchten, ob eine bestimmte View-Klasse aus den Frameworks von Apple auf Target-Action setzt, können Sie dazu die Dokumentation von Xcode zu Rate ziehen. Dort wird die Vererbungshierarchie jeder einzelnen Klasse im Feld *Inherits from* aufgelistet. Findet sich in dieser Auflistung die Klasse *UIControl*, dann wissen Sie, dass die betreffende Klasse mittels Target-Action konfiguriert werden kann (Bild 4).

UIControl stellt dabei einige grundlegende Funktionen zur Arbeit und Verwendung von Target-Action zur Verfügung. Sie können auch selbst eigene View-Klassen auf Basis von *UIControl* erstellen, um darüber eine Unterstützung für Target-Action zu implementieren.

Sie haben aber in diesem Abschnitt gesehen, was genau dahintersteckt, und sind in der Lage, selbst eigene Implementierungen für Target-Action zu schreiben und zu erstellen.

Delegation

Bleibt nun noch eine letzte Technik zur Kommunikation zwischen View und Controller, um das Thema »MVC in der iOS-Entwicklung« abzuschließen. Dabei handelt es sich um die sogenannte Delegation.

Dieser Name ist Programm. Vereinfacht ausgedrückt werden bei der Delegation tatsächlich einzelne Aktionen von einem Objekt an ein anderes delegiert. Statt also selbst eine Implementierung für eine bestimmte Aktion anzubieten, reicht ein Objekt die Durchführung der entsprechenden Aktion einfach an ein anderes Objekt weiter. Je nachdem, wie dieses andere Objekt die gewünschte Aktion implementiert hat, kommt ein anderes Ergebnis zustande, ähnlich wie bei Target-Action.

Listing 7: Definition einer Beispiel-View-Klasse

```
protocol ListViewDelegate {
    func didCreateListView()
    func didSelectRowAtIndex(index: Int)
}

class ListView {
    var delegate: ListViewDelegate
    init(delegate: ListViewDelegate) {
        self.delegate = delegate
        self.delegate.didCreateListView()
    }
    func selectRowAtIndex(index: Int) {
        delegate.didSelectRowAtIndex(index)
    }
}
```

Damit Delegation funktioniert, braucht es in jedem Fall ein Protokoll, das die zu delegierenden Methoden definiert. Warum das so ist, erfahren wir gleich. Für den Moment ist aber Fakt, dass bei Delegation ohne Protokolle nichts geht; sie sind gewissermaßen die Basis der Delegation.

Möchte nun eine Klasse – beispielsweise eine View – bestimmte Aufgaben an ein anderes Objekt delegieren, so erhält diese Klasse eine Referenz auf das entsprechende Objekt. So weit, so gut, schließlich muss die Klasse ja irgendwie die gewünschten Methoden des Objekts aufrufen können. Doch anstatt für dieses Objekt einen festen, statischen Typ anzugeben, wird stattdessen der Typ eines Protokolls verwendet, der festlegt, dass das Objekt konform zum angesprochenen Protokoll sein muss. Das hat zur Folge, dass wir zwar nicht wissen, welchen exakten Typ das der Klasse zugewiesene Objekt besitzt, aber wir wissen dafür sehr wohl, dass es die im Protokoll definierten Methoden implementiert und dass wir diese somit auf dem Objekt aufrufen können.

Genau das ist der Sinn von Delegation: Wir wissen nichts über den Typ des Objekts, an das wir Aufgaben delegieren, außer, dass es konform zu einem Protokoll ist, das all die Methoden deklariert, die wir für die Delegation benötigen.

Delegation in der Praxis

Sehen wir uns dazu einmal ein ganz konkretes Beispiel an. Als Vorbild soll die Klasse *UITableView* aus dem UIKit-Framework dienen, welche ebenfalls mit Delegation arbeitet. Zu diesem Zweck gibt es unter anderem das *UITableViewDelegate*-Protokoll, das beispielsweise definiert, was bei Auswahl einer Zelle in der Tabelle geschehen soll.

Listing 7 zeigt die Implementierung eines ähnlichen Beispiels, wenn auch deutlich vereinfacht. Dort werden ein Protokoll *ListViewDelegate* sowie eine View-Klasse *ListView* im-

Listing 8: Protokollkonforme Klassen

```
class MyFirstListViewController: ListViewDelegate {
    func didCreateListView() {
        print("Create list view with 20 rows.")
    }
    func didSelectRowAtIndex(index: Int) {
        print("MyFirstViewController selected row at index \(index).")
    }
}

class MySecondListViewController: ListViewDelegate {
    func didCreateListView() {
        print("Create list view with 100 rows.")
    }
    func didSelectRowAtIndex(index: Int) {
        print("MySecondViewController selected row at index \(index).")
    }
}
```

Listing 9: Zuweisung verschiedener Delegate-Objekte

```
let myFirstListViewController =
    MyFirstListViewController()
let mySecondListViewController =
    MySecondListViewController()
var listView = ListView(delegate:
    myFirstListViewController)
// Create list view with 20 rows.
listView.selectRowAtIndex(19)
// MyFirstViewController selected row at index 19.
listView = ListView(delegate:
    mySecondListViewController)
// Create list view with 100 rows.
listView.selectRowAtIndex(99)
// MySecondViewController selected row at index 99
```

plementiert. Das Protokoll definiert zwei Methoden, die somit jede Klasse implementieren muss, die konform zu *ListViewDelegate* ist. Die *ListView*-Klasse verfügt über eine Property namens *delegate*, der als Typ das Protokoll *ListViewDelegate* zugewiesen ist. Diese Deklaration hat zur Folge, dass *delegate* jedes beliebige Objekt zugewiesen werden kann, solange es nur konform zum Protokoll *ListViewDelegate* ist. Dadurch wissen wir, dass wir auf die Property *delegate* alle Methoden des Protokolls *ListViewDelegate* aufrufen können. Und genau das geschieht in der Implementierung der Klasse auch an zwei Stellen: einmal direkt während der Initialisierung, das andere Mal innerhalb einer eigens definierten Methode.

Damit ist Delegation bereits in ihren Grundzügen umgesetzt. Schließlich haben wir nun eine Klasse erstellt, die einzelne Aufrufe an ihr *delegate*-Objekt weiterreicht. Damit erhält man – ohne den Code von *ListView* ändern zu müssen – immer ein anderes Ergebnis, abhängig davon, welcher Delegate der Klasse zugewiesen wird. Auch das soll nun anhand eines Beispiels verdeutlicht werden. Listing 8 zeigt dazu die Deklaration zweier beispielhafter Controller-Klassen, die äquivalent zu einem *UITableViewController* agieren sollen. Beide Klassen sind konform zum *ListViewDelegate*-Protokoll und müssen daher die beiden im Protokoll deklarierten Methoden implementieren. Die Implementierung unterscheidet sich dabei jeweils bei beiden Klassen.

Werden nun Objekte dieser Klassen erzeugt, unterscheidet sich das Ergebnis, das wir aus einem *ListView*-Objekt heraus erhalten, abhängig davon, welcher Delegate ihm zugewiesen wird. Listing 9 verdeutlicht diese unterschiedlichen Ergebnisse, die abhängig vom gesetzten Delegate zustande kommen.

Vor- und Nachteile von Target-Action

Nach der Vorstellung der beiden typischen Techniken zur Kommunikation zwischen View und Controller bleibt auch hier die Frage, wann welche dieser Techniken idealerweise zum Einsatz kommt. Erfreulicherweise lässt sich das schnell und eindeutig beantworten. Target-Action ist ideal, wenn eine View genau eine bestimmte Aktion ausführen soll, die ein

zugeordneter Controller implementieren muss. Dann wird der View einfach das gewünschte Objekt mitsamt auszuführendem Selector zugewiesen und der Aufruf erfolgt dann vonseiten der View, sobald er benötigt wird. Allerdings ist diese Technik eben auch auf genau eine Aktion beschränkt. Soll eine View mehr als eine Aktion an einen Controller weiterreichen, kommt zwangsläufig Delegation ins Spiel.

Delegation ist etwas komplexer als Target-Action, nicht zuletzt aufgrund der Notwendigkeit eines zusätzlichen Protokolls, das die zu delegierenden Aufgaben in Form von Methoden definiert. Dafür hat eine View mit einem solchen Delegate-Objekt die Möglichkeit, deutlich mehr Aktionen an einen Controller weiterzuleiten und von ihm ausführen zu lassen. Gerade komplexe View-Klassen sind daher schnell auf Delegation angewiesen, da sie viele Informationen nicht selbst speichern, sondern dynamisch von einem zugewiesenen Controller abfragen.

Fazit

Das MVC-Pattern ist essenziell wichtig, wenn man Apps für iOS entwickelt. Bereits die System-Frameworks von Apple setzen darauf, und es finden sich dort massenhaft Techniken wie Target-Action, Delegation und Notifications. Man kommt mit MVC somit selbst bei einfachen Apps sehr schnell in Berührung.

Umso wichtiger ist es, beim Schreiben des eigenen Codes ebenfalls das MVC-Pattern zu beachten und die eigenen Klassen entsprechend zu strukturieren. Stellen Sie sich durchaus während der Erstellung einer Klasse die Frage, in welche der drei Bereiche – Model, View und Controller – die Klasse passt und wo sie hingehört. Wenn Sie darauf keine Antwort wissen, ist Ihre Code-Struktur möglicherweise nicht optimal und sollte noch einmal überdacht werden.

Hat man die verschiedenen Techniken des MVC-Patterns aber einmal verinnerlicht, sorgt die korrekte Anwendung der verschiedenen Funktionen schnell dafür, deutlich besser lesbaren und optimierten Code zu schreiben. Gerade bei der Arbeit im Team werden Ihnen Ihre Kollegen dieses Vorgehen danken. Und auch wenn Sie selbst einmal auf Fehlersuche sind, ist die Strukturierung des eigenen Codes nach MVC eine gute Grundlage, um sich schnell im Projekt zurechtzufinden und die korrekte Problemstelle auszumachen. Da es schließlich keine Abhängigkeiten zwischen Model- und View-Klassen gibt, können Sie schneller beurteilen, an welcher Stelle Sie aufgrund des jeweiligen Fehlers ansetzen müssen. ■



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.

SCHNELLE APP-ENTWICKLUNG

Mobile Strategie

Entwickler haben es oft schwer, den Wünschen von Geschäftskunden gerecht zu werden.

Schlaue Software-Entwickler haben sicherlich die Einführung des iPhones und des dazugehörigen iOS im Januar 2007 genau beobachtet. Richtig los mit der Revolution um die mobilen Apps ging es aber im Juli 2007, als der App Store im Zuge von iOS 2 vorgestellt wurde.

Apps von Drittanbietern waren nichts Neues im Smartphone-Umfeld, nicht einmal bei Apple, aber erst der App Store machte die Verbreitung, die Suche und den Kauf von Apps zu einem Kinderspiel.

Dann veröffentlichte Apple im April 2010 das iPad. Erneut war es so, dass Tablets nichts Neues waren, aber Apple ist es mit dem iPad gelungen, die Wirtschaftswelt auf sich aufmerksam zu machen. Als Mitarbeiter und Führungskräfte damit begannen, Smartphones und Tablets zur Arbeit mitzunehmen, stieg die Nachfrage nach mobiler Software sprunghaft an.

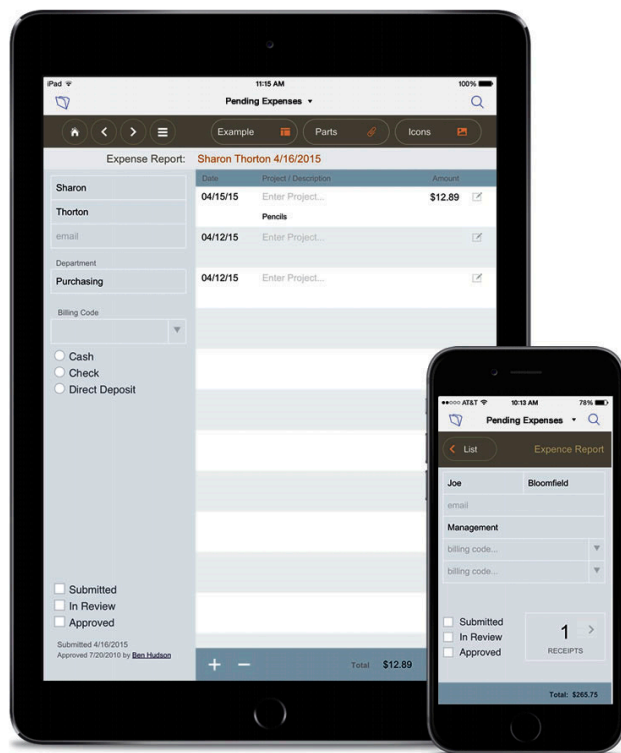
Zunächst wandten sich Entwickler kundenspezifischen Anwendungen zu, indem sie Tools wie Xcode, Swift (Apples Programmiersprache für iOS und OS X) verwendeten. Das alles geschah im Einklang mit der zunehmenden Popularität von Android und Java. Im Dienstleistungssektor machten Entwicklerfirmen Werbung für die mobile Software-Entwicklung und betrachteten mobile Apps als neuen Branchenstandard.

Von der Euphorie zum Pragmatismus

Die Entwicklung von nativen mobilen Anwendungen war für Großunternehmen machbar, solange hierfür Budget, Personal und Zeit aufgewendet werden konnten. Da sich der Markt jedoch weiterentwickelte und die anfängliche Begeisterung für mobile Software allmählich praktischem unternehmerischem Denken wich, begannen Unternehmen zunehmend damit, ihre etablierten Desktop-Plattformen auf mobile Geräte auszuweiten, anstatt auf die Entwicklung nativer mobiler Apps zu setzen.

Im Zuge der Weiterentwicklung des Mobilfunkmarkts haben fast alle großen Softwareplattformen auf dem Markt eine brauchbare mobile Strategie entwickelt. Von vertikalen Anwendungen bis hin zu Open Source, Web und proprietären Plattformen – heutzutage lohnt sich keine Software-Entwicklung mehr, bei der man ausschließlich auf den Desktop setzt.

Im Zusammenspiel all dieser Kräfte hat sich die Software-Entwicklung von Unternehmen in Richtung Geräteunabhängigkeit entwickelt: Bestehende Plattformen für den Desktop haben sich voll und ganz mobilen Geräten angepasst, und es ist heutzutage genau so altmodisch, eine rein mobile Anwendung zu entwickeln, wie eine reine Desktop-Version. Prag-



Eine App für verschiedene Plattformen (Bild 1)

matismus war eben schon immer eine der wirtschaftlichen Triebfedern. Das führte dazu, dass IT- und Softwareteams dazu angehalten waren, praktische Lösungen zu finden, die einer Codebasis auf dem am geeignetsten erscheinenden Gerät zum Durchbruch verhelfen.

Mehr als nur ein Lückenfüller

Und genau an diesem Punkt kommt die FileMaker-Plattform ins Spiel. Die FileMaker-Plattform, die von der gleichnamigen Apple-Tochter FileMaker vertrieben wird, ist einer der am meisten unterschätzten Technologieerfolge im Silicon Valley. In der heutigen Welt der mobilen Apps stellt die FileMaker-Plattform für Entwickler jedoch ein wichtiges und zugleich außergewöhnliches Instrument dar, um mobile Anwendungen für Unternehmen zu entwickeln.

Seit nunmehr 30 Jahren auf dem Markt, hat sich FileMaker von einer einfachen Datenbank-Anwendung zu einer strapazierfähigen Plattform mit eigenem Server, Skripts und Berechnungs-Tools sowie der Möglichkeit zur Integration von APIs entwickelt. Bestens geeignet zur Anwendung durch ganze Arbeitsgruppen, füllt es die Lücke zwischen klassischer Desktop-Software wie Excel und Anwendungen für

Großunternehmen wie Oracle oder SAP. Die Stärke von FileMaker liegt in der Benutzerfreundlichkeit: Ein Entwickler kann innerhalb von Stunden das erstellen, wozu man mit herkömmlichen Programmiersprachen Tage oder sogar Wochen benötigte.

Maßgeschneiderte FileMaker-Apps kommen als Desktop-Clients, mobile Anwendungen sowie als Webclients zum Einsatz – und das alles lässt sich mit nur einer einzigen Codebasis realisieren. Entwickler schaffen oft gerätespezifische Interfaces, aber die zugrunde liegende Anwendung, Datenarchitektur und Geschäftslogik unterliegt einer gemeinsamen Lösung.

Einmal erstellen und auf vielen Plattformen nutzen

All das ist wahrlich kein neues Konzept. Code-once- und Deploy-to-multiple-platforms-Lösungen gab es schon immer. Das Besondere an FileMaker ist jedoch, dass die Bereitstellung ein einmaliger Prozess ist, bei dem die App an den Anwendungsserver angebunden wird.

Das hat den Vorteil, dass Lösungen für ein bestimmtes Betriebssystem oder eine Plattform nicht extra erstellt oder angepasst werden müssen. FileMaker-Anwender greifen so auf eine App via Desktop, Mobilgerät oder Webclient zu, wie sie es beim Besuch einer Webseite über einen Browser auf einem Desktop- oder Mobilgerät tun würden.

Dieser Ansatz, eine App nur einmal zu erstellen und dann auf vielen Plattformen zu nutzen, vereinfacht die Entwicklung von maßgeschneiderten Apps erheblich und erleichtert die Arbeit von Entwicklern (Bild 1).

Zudem ist FileMaker keine Programmiersprache im Sinne von Java oder Objective-C: Es ist Plattform und integrierte Entwicklungsumgebung in einem. Client und Server kümmern sich um die gesamte Netzwerk-Kommunikation sowie um den Austausch, die Sicherheit und Interaktion mit demjenigen Betriebssystem, auf dem sie gerade laufen.

Entwickler erstellen Geschäfts-Apps nach Maß unter der Verwendung von Schnittstellenobjekten und Autorentools, die eher konfiguriert als programmiert werden müssen. Darüber hinaus steht eine komplette Skriptsprache zur Verfügung, um logische Routinen und Anwendungsfälle zu verwalten. Das alles ergibt eine fundierte Umgebung, innerhalb derer Entwickler zeitsparend und ohne komplexe Verfahren Lösungen entwickeln können.

Entwickler nutzen in FileMaker den Layoutmodus, mit dessen Hilfe sie Objekte auf dem Bildschirm platzieren und Skripts zu Knöpfen und anderen Triggern zuordnen. Die Plattform wird von Berechnungswerkzeugen unterstützt und bietet Berichte und Diagrammobjekte, Datenbankdefinitionen, Debugging-Tools, Stile und Designs und vieles mehr (Bild 2).

Wiedergabegetreuer Webclient

Obwohl FileMaker ein nativer Android-Client fehlt, bietet es einen sehr wiedergabegetreuen Webclient, der sich auch gut für Android-Nutzer eignet. Seine Oberfläche ist auf jene Elemente beschränkt, die FileMaker bereits von Haus aus un- ►

Trainings für Webentwickler

Codequalität mit JavaScript

Trainer: Golo Roden
3 Tage, Termin & Ort n. V.



Domain Driven Design mit PHP

Trainer: Stefan Pribsch
2 Tage, München, Termin n. V.



Testgetriebene PHP-Entwicklung

Trainer: Sebastian Bergmann
2 Tage, Köln, Termin n. V.



JavaScript für C#-Entwickler

Trainer: Golo Roden
3 Tage, Termin & Ort n. V.



Effiziente Softwarearchitekturen mit PHP

Trainer: Stefan Pribsch
2 Tage, München, Termin n. V.



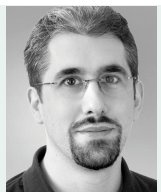
Qualitätssicherung in PHO-Projekten

Trainer: Sebastian Bergmann
2 Tage, Köln, Termin n. V.



PHP-Security

Trainer: Arne Blankerts
2 Tage, Hamburg, Termin n. V.



terstützt. Dazu zählen Registerfelder, Schieberegler, Popup-Panels, Ausschnittreihen (sogenannte Portale), Listenansichten und andere Funktionen.

Animationen dagegen werden zum Beispiel nicht unterstützt. Aber man will ja auch kein Computerspiel mit FileMaker programmieren.

Für Geschäftsanwendungen jedoch können mit FileMaker verschiedenste Interface-Komponenten kombiniert werden, was Entwicklern eine nahezu unendliche Palette an Optionen bietet.

FileMaker eignet sich auch nicht besonders, um große Hochleistungssysteme zu unterstützen. Benötigt man eine Anwendung, um Tausende von Transaktionen pro Minute zu verwalten oder um Tausenden von Benutzern gleichzeitig die Arbeit zu ermöglichen, ist man mit einer auf Großunternehmen zugeschnittenen Lösung auf jeden Fall besser bedient.

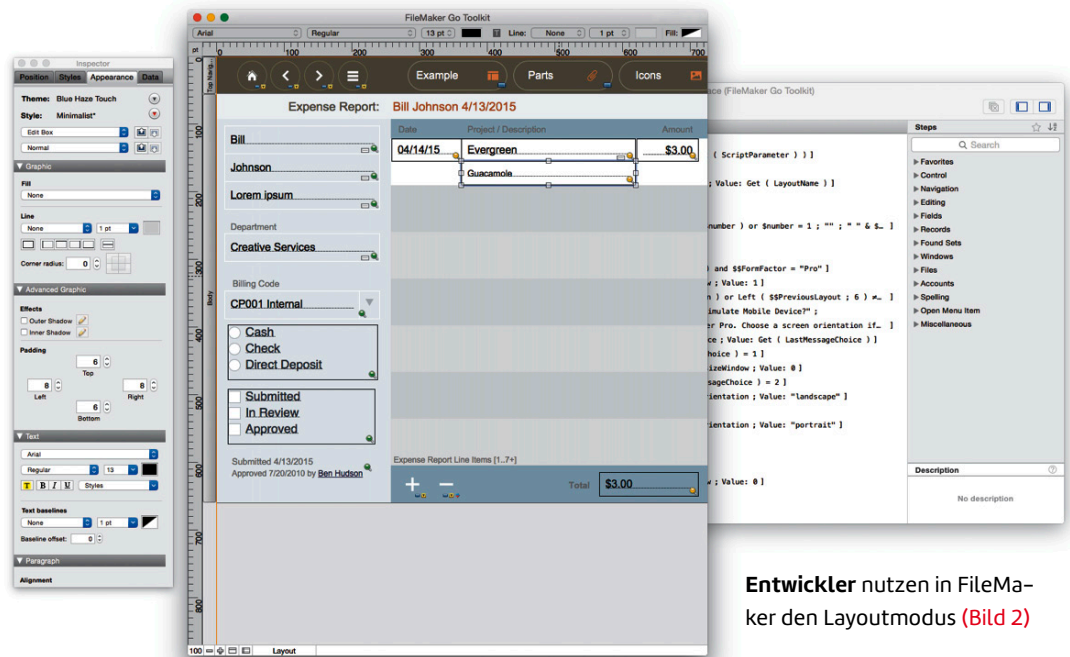
Zwar bringt die Entwicklung nativer Apps unglaubliche Freiheiten mit sich. Andererseits bedeutet dies auch, dass sich der Zeitaufwand für Entwickler um den Faktor 10 erhöht. Für Unternehmen, die hingegen auf FileMaker setzen, wird der Return on Investment (ROI) zum Kinderspiel.

Fest etabliert

Ein zweiter Faktor, der für FileMaker spricht, ist die Tatsache, dass es sich hier um eine seit 30 Jahren auf dem Markt bestehende Plattform handelt, die durch ein ausgereiftes Ökosystem an Drittanbietern unterstützt wird und hinter der ein fest etabliertes Mutterunternehmen steht: Eine native App hingegen ist immer nur so zu verlässlich wie das Team, das dahinter steckt.

Mit FileMaker weiß ein IT-Team genau, was es bekommt, und es kann sich auf die Plattform stützen. Sicherheit und Authentifizierung sowie Netzwerkanbindung sind bereits im Produkt enthalten, und FileMaker Server enthält Admin-Tools, Backup-Funktionen und vieles mehr. FileMaker bietet die althergebrachte Zuverlässigkeit einer Plattform und ein Ökosystem, das native Apps nicht unbedingt gewährleisten können.

Drittens: Wenn eine FileMaker-App einmal in einem Unternehmen installiert wurde, können so viele Lösungen hinzugefügt werden wie nötig. Sobald eine Investition in die Plattform-Infrastruktur erstmal getätigt wurde, gibt es nur wenig bis gar keine Folgekosten, die beim Hinzufügen zusätzlicher Apps anfallen, abgesehen von Lizenzgebühren für zusätzliche Benutzer.



Entwickler nutzen in FileMaker den Layoutmodus (Bild 2)

FileMaker entwickelt sich zu einer sehr gut unterstützten Suite, die sich kontinuierlich entwickelt, wächst und mit dem Unternehmen verändert, ohne auf zusätzliche Infrastruktur zurückgreifen zu müssen.

Wenn man sich ansieht, wie einfach es ist, mit FileMaker zu entwickeln, dann kann ein Unternehmen seine größte Stärke ausspielen: die Inhouse-Entwickler. Genau so, wie manche Mitarbeiter in Unternehmen zu Experten in Excel oder anderen Business-Tools werden, verhält es sich bei FileMaker. Die überwältigende Mehrheit der FileMaker-Entwickler kommt von der Geschäftsseite oder der jeweiligen Branche. So kann ein bereits erfahrener Anwender zu einem FileMaker-Neuling werden und sich dann zu einem zertifizierten und erfahrenen Entwickler weiterbilden. FileMaker kann einem Unternehmen sozusagen das Laufen beibringen.

Für Unternehmen, die eine breite Palette an Geschäftslösungen einsetzen möchten, ganz gleich ob inhouse oder von einem Drittanbieter entwickelt, hält FileMaker Funktionen bereit, die so nur wenige Plattformen bieten.

Native Apps bieten zwar Flexibilität und sind mächtig, sie können in ihrer Entwicklung jedoch zeitaufwendig sein und stark in der Qualität schwanken. Der permanente praktische Einsatz von FileMaker bietet hier eine echte Alternative und einen Wettbewerbsvorteil. ■



Scott Love

ist der COO und Mitbegründer von Soliant Consulting, einem Drittanbieter für Software-Entwicklung, der sich auf FileMaker, Salesforce und Webanwendungen spezialisiert hat.

www.soliantconsulting.com

Jetzt kostenlos testen!



Das Fachmagazin für IT-Entscheider

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

www.com-magazin.de/gratis



Foto: Fotolia / sdecoret

ENTERPRISE MOBILITY MANAGEMENT (EMM)

Mobile Devices verwalten

EMM-Lösungen sind beim Management von Smartphones und Tablets unverzichtbar.

Die Nutzung mobiler Endgeräte in Unternehmen steigt rapide. Allein 2014 hat die Zahl der in Firmen verwalteten Smartphones und Tablets im Jahresvergleich um 72 Prozent zugenommen, stellt der Mobile Analytics Report 2015 von Citrix fest. Bereits 2017 wird ein Fünftel aller Tablets nicht privat, sondern für den geschäftlichen Einsatz gekauft werden, so die Marktforscher von Forrester. Auch auf die weltweiten Datenströme hat das veränderte Nutzungsverhalten massive Auswirkungen. Nach einer Prognose von Cisco wird sich der geschäftliche Datenverkehr über mobile Netze bis zum Jahr 2019 gegenüber 2014 fast verzehnfacht haben, in der Region Westeuropa soll er auf das Siebenfache anwachsen.

Beim Management der mobilen Geräte sieht es allerdings düster aus, wie der erwähnte Citrix-Report aufzeigt. Das Unternehmen hat auf Basis von Netzbetreiberdaten und mit Hilfe von Big-Data-Analysen weltweit das Verhalten der Nutzer im mobilen Netz analysiert und die Ergebnisse erstmals nach Privatkunden und Unternehmen aufgeschlüsselt. Letztere nutzen die Möglichkeiten aktueller Enterprise-Mobility-Management-Lösungen nur unzureichend (Bild 1). Immerhin 90

Prozent machen einen Passwortschutz zur Pflicht, aber nur circa ein Viertel kontrolliert die Nutzung von Apps, GPS oder WLAN auf den mobilen Endgeräten, und nur 11 Prozent verlangen eine sichere Kommunikation über Virtual Private Networks (VPN).

Zu etwas positiveren Ergebnissen kommt das Marktforschungsunternehmen IDC, das im Mai 2015 Fach- und Führungskräfte in 243 deutschen Unternehmen befragt hat.

Enterprise-Mobility-Strategie

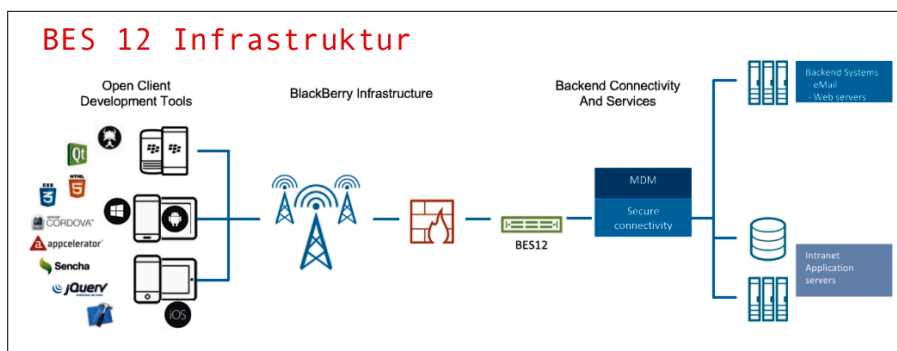
Demnach haben 72 Prozent der befragten IT-Verantwortlichen Anti-Malware-Software auf den mobilen Endgeräten im Einsatz, 43 Prozent schalten einzelne Gerätefunktionen gezielt ab und 50 Prozent machen Vorgaben für eine sichere Verbindung über Proxies oder Gateways. Immerhin 35 Prozent planen die Einführung einer Geräteverschlüsselung, 54 Prozent haben sie bereits im Einsatz, und nur 8 Prozent beschäftigen sich gar nicht mit dem Thema.

Bei der Konzeption einer Enterprise-Mobility-Strategie muss man allerdings mehr als nur Sicherheitsaspekte berücksichtigen, sagt Stratos Komotoglou, Senior Manager EMEA Product & Field Marketing beim EMM-Spezialisten MobileIron: »Natürlich ist Sicherheit das Fundament, aber die Einführung eines EMM-Systems in Kombination mit der Einführung von Business-Apps kann den Mitarbeitern helfen, produktiver zu arbeiten.« Das kann zu erheblichen Einsparungen führen, wie Komotoglou am Beispiel eines amerikanischen Kunden zeigt, der MobileIron zusammen mit Business-Apps in Krankenhäusern einsetzt: »Man spricht dort von Einsparungen in Höhe von 50.000 Dollar pro Woche.«

»Ein strategisches Vorgehen ist gefragt, denn nur so kann das Potenzial von EMM komplett ausgeschöpft werden«, findet auch Sascha Milani, Principal Mobility Consultant & Senior Solution Architect bei dem IT-Fullservice-Anbieter Freudenberg IT. Es müssten nicht nur die technischen Voraussetzungen geschaffen werden, sondern auch die Arbeitsweise und die Erwartungshaltung der Mitarbeiter müsse berücksichtigt werden, so Milani weiter. »Ohne diese Schritte entstehen meist hohe Folgeaufwände.« Nach Ansicht von Wolfgang Schwab, Lead Advisor Platforms & Infrastructure bei der Experton Group, ist diese Erkenntnis aber noch nicht weit genug verbreitet: »Nicht nur im Mittelstand, auch bei den großen Konzernen gibt es immer noch viele, die ihre mobile Infrastruktur gar nicht oder nur halbherzig managen.«

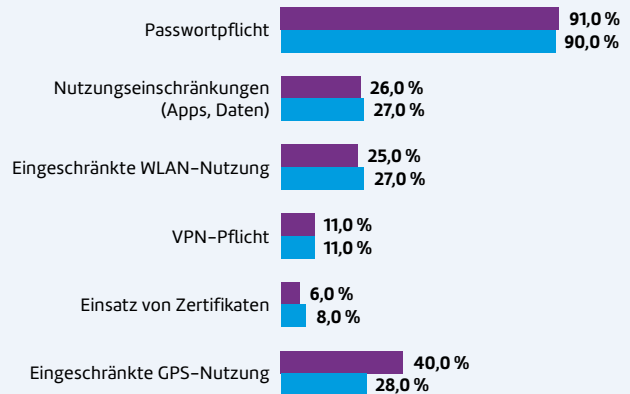
Die Bestandteile

EMM geht also über die Kontrolle von mobilen Endgeräten weit hinaus. Dennoch ist die reine Verwaltung, meist als Mobile Device Management (MDM) bezeichnet, auch heute noch für viele Anwender der Einstieg ins Enterprise Mobility Management – und das Herzstück jeder EMM-Strategie (Bild 2). »Die Erfahrung hat gezeigt, dass die größte Hebelwirkung die Kernkompetenz Mobile Device Management bietet«, sagt Freudenberg-Consultant Milani. Zu den wichtigsten MDM-Funktionen gehören die Inventarisierung und Konfiguration aller mobilen Endgeräte, eine Versionskontrolle des Betriebssystems und das automatisierte Einspielen von Updates, das Ein- und Ausschalten von Gerätefunktionen wie Kamera, WLAN, Bluetooth oder GPS sowie das entfernte Sperren (Remote Lock) beziehungsweise das entfernte Löschen (Remote Wipe) bei Verlust des Geräts.



Mobile Device Management (MDM) ist das Herzstück jeder EMM-Strategie (Bild 2)

Sicherheitsrichtlinien für mobile Endgeräte



Ein durchgängiges Management von mobilen Endgeräten ist bei den Unternehmen noch wenig verbreitet (Bild 1)

web & mobile developer 5/2016

Quelle: Citrix Mobile Analytics Report 2015

Zusätzlich zu den Geräten sind die darauf befindlichen Applikationen zu verwalten, was als Mobile Application Management (MAM) bezeichnet wird. Das ist besonders dann wichtig, wenn der Mitarbeiter das Gerät nicht nur dienstlich, sondern auch für private Zwecke einsetzt.

Um private und geschäftliche Nutzung zu trennen, gibt es im Prinzip drei Möglichkeiten:

- Das Gerät kann logisch in zwei voneinander getrennte Funktionseinheiten aufgeteilt werden (Dual Persona). Die IT hat nur Zugriff auf den unternehmenseigenen Teil, sie kann diesen verwalten und gegebenenfalls löschen.
- Business-Apps lassen sich in einem eigenen Container auf dem Endgerät installieren. In diesen Containern sind bestimmte Funktionen deaktiviert, zum Beispiel Copy and Paste oder das Teilen von Inhalten über soziale Netzwerke.
- Beim sogenannten App-Wrapping kombiniert der Administrator eine App mit zusätzlichen Sicherheits- und Verwaltungsfunktionen und stellt diese anschließend über den firmeneigenen App Store zum Download zur Verfügung.

Ein weiterer Bereich von EMM ist das Mobile Content Management (MCM), die Verwaltung von Inhalten. MCM stellt sicher, dass sensible Firmeninformationen nicht in fremde

Hände geraten (Bild 3). Auch der Datenschutz spielt eine Rolle, wenn personenbezogene Daten auf den mobilen Endgeräten gespeichert sind, etwa Kundendaten in einem mobilen CRM. MCM umfasst Funktionen wie die Verschlüsselung von Ordnern und Dokumenten, E-Mails oder E-Mail-Anhängen. Auch Data Loss Prevention (DLP) kann ein Bestandteil sein. DLP-Produkte verhindern den Abfluss von Informationen, indem sie etwa als vertraulich klassifizierte Dokumente überwachen. ►

Schließlich ist noch die Authentifizierung und Autorisierung als Funktionseinheit zu betrachten, das sogenannte Mobile Identity Management (MIM) (Bild 4). Hierzu gehören beispielsweise das Passwortmanagement, rollenbasierte Zugangsbeschränkungen für Apps und Daten oder eine erweiterte Mehrfaktoren-Authentifizierung, etwa mittels Tokens oder Wearables.

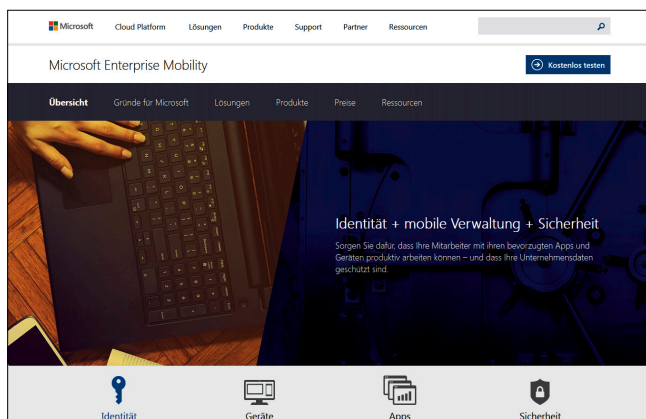
Wie sich EMM weiterentwickelt

Der Fokus in Unternehmen verschiebt sich vom reinen Mobile Device Management hin zum Content- und App-Management, sagt Thomas Völker, Senior Consultant Cybersecurity & Enterprise Mobility bei Fritz & Macziol Software und Computervertrieb: »Enterprise Mobility Management wird immer umfassender und schließt auch Bereiche wie Identity Management, Mobile Analytics und natürlich Cloud Computing ein.« Ziel sei dabei die Beschleunigung und Mobilisierung von Geschäftsprozessen durch mobile Applikationen, beispielsweise in Vertrieb, Marketing, Service und Support oder dem Betriebsmanagement. »Aufgrund von immer mehr mobilen Mitarbeitern findet eine Appification bestehender Unternehmensanwendungen statt.«

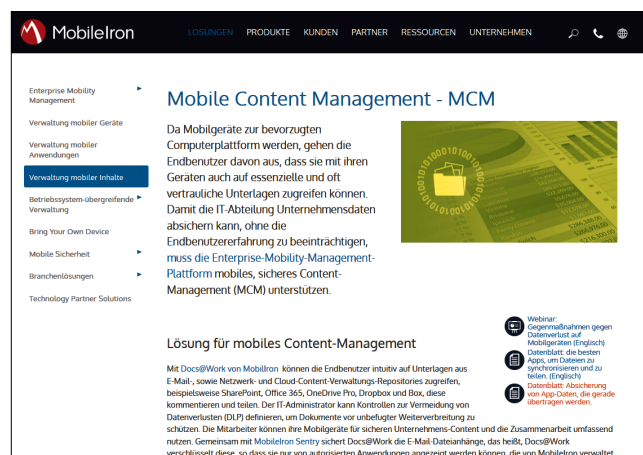
Doch nicht nur die Zahl der Apps nimmt zu – zumal viele Unternehmen selbst in die App-Entwicklung eingestiegen sind –, sondern auch die Vielfalt an zu verwaltenden mobilen Endgeräten. »Im Zuge der rasch voranschreitenden Digitalisierung sind Geräte aus dem Bereich Smart Home, Industrie 4.0 oder zum Beispiel Wearables von Interesse«, sagt Bernd Bäumler, Head of MDM bei der Unternehmensberatung Seven Principles.

Der Trend weg von klassischen Anwendungen mit ihrer offenen Betriebssystemarchitektur hin zu gekapselten Apps, wie man sie aus Android oder iOS kennt, hat laut Stratos Komotoglou von MobileIron längst auch die klassischen Desktop-Umgebungen erreicht: »Windows 7 ist das letzte traditionelle Betriebssystem von Microsoft.«

In der Konsequenz verschwimmen die Grenzen zwischen der Verwaltung mobiler Endgeräte und klassischer PCs oder Notebooks. »Enterprise Mobility Management entwickelt sich weiter zu einem Unified Device Management, mit dem



Mobile Identity Management (MIM) ist ein weiterer Bereich von EMM (Bild 4)



Beim Mobile Content Management (MCM) geht es um die Verwaltung von Inhalten (Bild 3)

Unternehmen ihre mobilen Smart Devices ebenso verwalten können wie ihre stationären Geräte«, sagt Elke Papaioannou, Solution Manager bei dem IT-Dienstleister Computacenter. Die EMM-Lösungen von Microsoft, MobileIron, Citrix und VMware seien bereits in der Lage, Windows-10-Clients über die MDM-Schnittstelle zu managen (Bild 5). »Gerade Laptops haben immer mehr Verwaltungsfunktionen im Betriebssystem integriert, sodass sie sich wie Mobilgeräte steuern lassen«, ergänzt Sascha Milani von Freudenberg IT. Experten trägt dieser Entwicklung Rechnung und hat den Mobile Enterprise Vendor Benchmark aus dem vergangenen Jahr zum Digital Workspace Vendor Benchmark 2016 weiterentwickelt.

Kriterien für ein gutes EMM

Funktionsvielfalt ist also ein wichtiges Kriterium für EMM-Lösungen geworden. Ein Produkt, das nur reines Gerätemanagement beherrscht, kann höchstens noch in Spezialfällen überzeugen. Die Vielzahl an Features geht allerdings auf Kosten der Benutzerfreundlichkeit und Übersichtlichkeit – und außerdem häufig an den Anforderungen der Anwender vorbei. »Es gibt auf dem Markt sehr wenige MDM-/EMM-Anbieter, die maßgeschneiderte Lösungen anbieten«, sagt Bernd Bäumler von Seven Principles. Flexibilität, Erweiterbarkeit und Zukunftssicherheit sind weitere Kriterien für die Kaufentscheidung. Eine besondere Rolle kommt dabei dem Zero-Day- beziehungsweise Same-Day-Support zu. »Darunter versteht man, dass neue Betriebssysteme oder -funktionen direkt am Veröffentlichungstag unterstützt werden«, erklärt Sascha Milani von Freudenberg IT.

Zukunftsfähigkeit bedeutet aber auch, dass der Anbieter und das Produkt die Konsolidierung überleben, die den EMM-Markt erfasst hat. Bereits 2012 hatte Citrix den MDM-Anbieter Zenprise gekauft, 2013 folgte die Übernahme von Fiberlink durch IBM. 2014 ging AirWatch für rund 1,5 Milliarden Dollar an VMware, und im vergangenen Jahr verließ sich BlackBerry Good Technology ein. »Die großen Player auf dem Markt haben alle erkannt, dass EMM nicht nur ein Trend, sondern ein Fundament für IT-Security sein wird und

auch schon ist«, sagt MobileIron-Manager Komotoglou. MobileIron werde sich weiter als unabhängiger Anbieter auf EMM konzentrieren, verspricht er.

Tatsächlich zählen die meisten befragten Systemhaus-Experten ebenso wie Experton oder Gartner MobileIron zu den führenden EMM-Anbietern. Am häufigsten genannt werden darüber hinaus VMware AirWatch, Citrix XenMobile, Good Technology (BlackBerry) und Microsoft Intune. »Es werden die EMM-Anbieter übrig bleiben, die dem Kunden über MDM hinaus funktionale Mehrwerte bieten und das Thema mobiler Arbeitsplatz ganzheitlich abdecken können«, sagt Thomas Völker von Fritz & Macziol. »Weiterhin wird es immer Nischen geben, in denen sich kleinere MDM-Anbieter mit für den Kundenbedarf maßgerechten Lösungen positionieren können«, ergänzt Bäumler von Seven Principles. Als Beispiel für einen solchen Spezialanbieter nennt Matthias Beck, der als Technical Sales, Competence Center Mobile Solutions beim Bechtle IT-Systemhaus tätig ist, die Casper Suite von JAMF Software, mit der sich Macs, iPhones und iPads managen lassen.

EMM richtig einführen

Anforderungen definieren, seitenlange Funktionslisten durchforsten und dabei auch noch die Zukunftssicherheit des Anbieters im Blick haben – die Auswahl des richtigen EMM-Tools ist keine leichte Aufgabe. Generell sollten CIOs und IT-Administratoren immer mit einer Erfassung des aktuellen Status beginnen, empfiehlt Freudenberg-Consultant Milani:

Ausgewählte Enterprise-Mobility-Plattformen

Die nachfolgende Aufstellung gibt einen Überblick über das Angebot an aktuellen Enterprise-Mobility-Management-Lösungen.

- Citrix / XenMobile
<https://www.citrix.de/products/xenmobile/overview.html>
- MobileIron
<https://www.mobileiron.com/de>
- BlackBerry / Enterprise Server 12
<http://de.blackberry.com/enterprise.html>
- Microsoft / Enterprise Mobility Suite
<https://www.microsoft.com/de-de/server-cloud/enterprise-mobility>
- IBM / MaaS360
www.maas360.com
- ISEC7 / EMM Suite
www.isec7.com/de/produkte/isec7-emm-suite
- VMware / AirWatch
www.air-watch.com/de
- Sophos / Mobile Control
<https://www.sophos.com/de-de/products/mobile-control.aspx>
- Symantec / Symantec Mobility Suite
<https://www.symantec.com/de/de/mobility>



Moderne EMM-Lösungen sind in der Lage, Windows-10-Clients über die MDM-Schnittstelle zu managen (Bild 5)

»Wer setzt welche Geräte für welche mobilen Arbeitsprozesse ein?« Anschließend sei zu bewerten, welche Einsatzszenarien gemäß der Geschäftsziele sinnvoll und wünschenswert sind – und welche nicht. »Gerade hier stellen die Vielfalt der mobilen Geräte sowie die zahlreichen Betriebssysteme und unterschiedlichen Apps Verantwortliche beim Treffen der richtigen Entscheidungen vor große Herausforderungen.«

Noch komplizierter wird das Ganze dadurch, dass nicht nur der Status quo im Blick zu behalten ist, sondern auch die zukünftige Entwicklung abgeschätzt werden muss, sagt Fritz-&Macziol-Consultant Völker: »Es sollte eine ganzheitliche Mobility-Strategie erarbeitet werden – beginnend mit einer methodisch sinnvollen Anforderungsanalyse, die nicht nur die aktuellen, sondern auch die möglichen zukünftigen Anforderungen des Unternehmens berücksichtigt.«

Vor allem sollten Unternehmen möglichst bald eine Entscheidung über die Eigentumsmodelle treffen, empfiehlt Elke Papaioannou von Computacenter. Bring Your Own Device (BYOD), also der Einsatz privater Endgeräte im Unternehmen, ist in Deutschland nicht sehr verbreitet, was unter anderem an steuerrechtlichen Problemen liegt. Viele Unternehmen setzen auf das CYOD-Konzept (Choose Your Own Device), bei dem der Anwender aus einer Geräteliste ein Smartphone oder Tablet auswählen kann, das dann vom Unternehmen angeschafft wird. Je nachdem ob das Gerät auch privat genutzt werden darf oder nicht, unterscheidet man zwischen COPE (Company Owned Personally Enabled) und COBO (Company Owned Business Only).

Aus der Entscheidung für oder gegen ein Beschaffungsmodell ergeben sich rechtliche Anforderungen, damit Datenschutzgesetze oder Compliance-Vorgaben nicht verletzt werden. Rechtsabteilung, Betriebsrat, aber auch Fachabteilungen sollten deshalb möglichst frühzeitig in die Planung miteinbezogen werden.

Überhaupt ist Kommunikation ein wesentlicher Schlüssel für den Erfolg beim EMM-Rollout. Schließlich sollten Unternehmen die Anwender persönlich in die Pflicht nehmen, ►

Interview

»Es mangelt an einer echten mobilen Strategie.«

Wolfgang Schwab, Lead Advisor Platforms & Infrastructure bei Experton, befasst sich mit der Zukunft des Arbeitsplatzes. Im Interview erklärt er, welche Rolle mobile Endgeräte dabei spielen und welche Fehler Unternehmen beim Management der Geräte machen.

web & mobile developer: Herr Schwab, Sie arbeiten gerade am Digital Workspace Vendor Benchmark 2016, dem thematisch erweiterten Nachfolger des Mobile Enterprise Vendor Benchmark 2015. Wie hat sich der Markt seit der Untersuchung im vergangenen Jahr verändert?

Wolfgang Schwab: Das können wir noch nicht abschließend sagen. Es zeichnet sich aber bereits ab, dass sich die Ausrichtung der Unternehmen verändert. Sie betrachten Mobility nicht mehr isoliert, sondern als Teil einer ganzheitlichen Arbeitsplatzstrategie. Das ist ein relativ neuer Ansatz, der wahrscheinlich auch zielführender ist, als Desktop-Clients und mobile Endgeräte getrennt zu behandeln.

web & mobile developer: Heißt das, Unternehmen setzen auf eine virtuelle Desktop-Infrastruktur, auf die mit mobilen und stationären Endgeräten gleichermaßen zugegriffen werden kann?

Schwab: Nein, das funktioniert noch nicht. Wonach Unternehmen zunächst suchen, ist eine einheitliche Management-Plattform, mit der man alle Endgeräte aus einer Oberfläche heraus verwalten, Apps zur Verfügung stellen und Rechte rollenbasiert vergeben kann. Wichtig ist auch ein ordentlicher Workflow für Bestellung, Freigabe und Provisionierung von Tools und Funktionen.

web & mobile developer: Gibt es diese Lösungen schon?

Schwab: Das Angebot ist derzeit noch sehr überschaubar. Matrix42 wäre hier zu nennen, AirWatch ist auf dem besten Weg dorthin, und auch MobileIron bietet solche Funktionen in gewissem Umfang.

web & mobile developer: Ist diese ganzheitliche Strategie nicht etwas überambitioniert? Viele Unternehmen im Mittelstand haben ja noch nicht einmal ein vernünftiges Enterprise Mobility Management implementiert.

Schwab: Nicht nur im Mittelstand, auch bei den großen Konzernen gibt es immer noch viele, die ihre mobile Infrastruktur gar nicht oder nur halbherzig managen. Das ist brandgefährlich. Ich habe letztlich keine Kontrolle über die Endgeräte und weiß nicht, was darauf passiert.

web & mobile developer: Warum ignorieren Unternehmen diese offensichtlichen Gefahren?

Schwab: Weil es nach wie vor an einer echten mobilen Strategie



Wolfgang Schwab
Lead Advisor Platforms
www.experton-group.de

mangelt. Es ist außerdem bisher relativ wenig passiert, sodass sich die Unternehmen in falscher Sicherheit wiegen.

web & mobile developer: Spielt es auch eine Rolle, dass der Markt recht unübersichtlich ist?

Schwab: Ich weiß nicht, ob der Markt so unübersichtlich ist. Es gibt vielleicht 30 ernst zu nehmende Firmen, der Rest

sind Nischenanbieter mit einem sehr eingeschränkten Portfolio, die im ein oder anderen Fall vielleicht die richtige Lösung haben.

web & mobile developer: Nach welchen Kriterien sollte man eine EMM-Plattform auswählen?

Schwab: Man kann natürlich einen Feature- und Kostenvergleich (Software- und Integrationskosten) durchführen – falls man eine mobile Strategie hat, sollte man dies auch tun. Für alle anderen, und das sind die meisten, gilt, dass man mit den großen Anbietern nicht viel falsch macht. Unterm Strich sind die Kleineren mit wenigen Ausnahmen auch nicht wesentlich günstiger. Bei den Marktführern kann ich mir außerdem relativ sicher sein, dass sie auch in zwei oder drei Jahren noch existieren. Ein EMM-Tool einzuführen ist zwar nicht der Riesenaufwand, aber wenn man den Anbieter wechseln muss, weil dieser den Markt verlässt, sind die Kosten nicht unerheblich.

web & mobile developer: Im EMM-Markt gab es in den vergangenen Jahren eine ganze Reihe von Firmenübernahmen. So hat etwa VMware AirWatch gekauft und BlackBerry Good Technologies. Gehen Sie davon aus, dass die Konsolidierung weitergehen wird?

Schwab: Auf jeden Fall. Es gibt definitiv zu viele Player, als dass alle überleben könnten. Dazu kommt, dass vor allem im Bereich Mobile Device Management die wenigsten Anbieter Gewinne machen. Sie müssen alle mehr oder weniger schnell ihr Portfolio auf Enterprise Mobility Management erweitern, aber ob das ausreichend ist, das wage ich zu bezweifeln.

web & mobile developer: EMM-Lösungen werden gern unterteilt in MDM, MAM, MCM oder MIM. Ist diese Kategorisierung noch zeitgemäß?

Schwab: Letztlich nicht, das werden wir in diesem Jahr auch nicht mehr so machen. EMM insgesamt beziehungsweise die Teilsegmente MDM und MIM kann man betrachten. Alles andere macht keinen Sinn, zumal es ohnehin immer die gleichen Lösungen sind, nur aus unterschiedlichen Blickwinkeln gesehen.

web & mobile developer: Im Vendor Benchmark 2015 haben Sie bemängelt, dass viele Unternehmen keine klare Entscheidung treffen zwischen Bring Your Own Device (BYOD) und der Beschränkung auf unternehmenseigene Geräte. Hat sich das verändert?

Schwab: Von BYOD haben sich die meisten verabschiedet, weil das steuertechnisch schwierig ist und Haftungsfragen nach wie vor ungeklärt sind. Der Trend geht eher in Richtung Choose Your Own Device, zumindest was rein mobile Endgeräte wie Smartphones und Tablets betrifft. Es wird in den meisten Unternehmen allerdings stillschweigend geduldet, dass Mitarbeiter ihre eigenen Geräte mitbringen. Mit der richtigen Management-Software, die den privaten vom geschäftlichen Bereich trennt, ist das auch kein Problem.

web & mobile developer: Sie hatten auch vorhergesagt, dass 2015 das Jahr werde, in dem Unternehmen ein ganzheitliches Rollenkonzept für mobiles Arbeiten einführen werden. Inwieweit ist diese Prognose von Ihnen eingetreten? Welche Aktivitäten gab es da in den Unternehmen?

Schwab: Von der Unternehmens-IT haben wir wenig Aktivität in diesem Bereich gesehen. Aus den Fachabteilungen kamen dagegen viele Impulse, allerdings weniger mit einer strategischen Ausrichtung, sondern eher mit dem pragmatischen Ziel, vernünftig arbeiten zu können.

web & mobile developer: Sie prognostizieren für 2016, dass die Digitalisierungswelle auch einen massiven Einfluss auf die mobile Strategie eines Unternehmens haben wird. Wie meinen Sie das?

Schwab: Am herkömmlichen Arbeitsplatz wird sich wenig ändern, aber in Bereichen, in denen Mobilität ein entscheidendes Kriterium ist, wird sich einiges tun, etwa im Vertrieb, im Service oder in der Lagerlogistik. Wer hier mithalten will, wird um eine mobile Strategie zumindest in Teilbereichen nicht herumkommen.

web & mobile developer: Wer treibt diesen Strategiewechsel voran?

Schwab: Nicht unbedingt die IT-Mitarbeiter. Wir sehen es dagegen relativ häufig, dass es neben der klassischen IT einen eigenständigen Bereich mit einem sogenannten Chief Digital Officer gibt, der für die Digitalisierung zuständig ist. Diese CDOs haben oft weitreichende Befugnisse, in die IT-Strategie einzugreifen.

web & mobile developer: Heißt das, die klassische IT wird entmachtet?

Schwab: Es ist sogar noch schlimmer. Die klassische IT darf die langweiligen Altsysteme betreiben, und alles, was innovativ und spannend ist, machen die neuen ITler. Wenn man sich ansieht, wie passiv sich die klassischen IT-Abteilungen in den vergangenen fünf bis sechs Jahren im Mobile-, aber auch im Cloud-Umfeld verhalten haben, ist das auch kein Wunder.

beim Umgang mit mobilen Endgeräten und Daten die nötige Sorgfalt walten zu lassen. »Wir empfehlen den Bechtle-Kunden, dass Mitarbeiter bei der Ausgabe mobiler Arbeitsmittel immer eine Nutzungsvereinbarung unterzeichnen, die Rechte und Pflichten regelt – vor allem dann, wenn die Geräte auch privat genutzt werden dürfen«, sagt Matthias Beck vom Bechtle-Systemhaus Neckarsulm.

Die Frage, ob ein Unternehmen überhaupt ein Enterprise Mobility Management benötigt, stellt sich nach Ansicht der Experten dagegen nicht mehr. »Sobald ihre mobilen Geräte Unternehmensdaten enthalten, kommen die Kunden an einer EMM-Lösung nicht mehr vorbei«, sagt Computacenter-Managerin Papaioannou. Selbst wenn ein Unternehmen das gar nicht will, kann es nie sicher sein, dass die Mitarbeiter nicht doch Daten auf ihren mobilen Geräten speichern, warnt Thomas Völker von Fritz & Macziol: »Sie nutzen dafür die aus dem privaten Umfeld bekannten kostenlosen Dienste, um sich Daten auf ihre Smartphones zu holen.« Ohne entsprechende Infrastruktur kann das Unternehmen nicht kontrollieren oder steuern, welche Daten wo und wie gespeichert beziehungsweise verarbeitet werden. »Aus diesem Grund sollte jedes Unternehmen eine EMM-Lösung nutzen, um der IT-Abteilung das Heft in die Hand zu geben«, so Völker weiter.

Fazit

Enterprise Mobility Management ist eine wichtige strategische Aufgabe in Unternehmen. Dennoch setzen die Firmen EMM häufig nur halbherzig oder nur in Teilbereichen um. Wer so vorgeht, handelt sich früher oder später massive Probleme ein. Zwar sind die mobilen Betriebssysteme mit ihrem Sandboxing-Ansatz von Haus aus wesentlich sicherer als die traditionellen Pendants auf dem Desktop. Dennoch haben Kriminelle längst erkannt, wie wertvoll die Daten auf mobilen Geräten sind, und greifen rigoros an. »Die vier wichtigsten Bedrohungen, die zurzeit für iOS- und Android-Geräte existieren, nutzen äußerst clevere Taktiken. Die Benutzer merken möglicherweise überhaupt nicht, dass ihre Geräte infiziert wurden, bis ihre Daten gehackt sind«, warnt Mobile-Iron-Manager Komotoglou.

Wenn ein Unternehmen seine mobile Infrastruktur nicht genauso konsequent absichert wie PCs, Server und Netzwerk, ist es nur eine Frage der Zeit, bis es zu gravierenden Sicherheitsproblemen kommt. Nur mit einer einheitlichen Sicht auf alle mobilen Endgeräte, Applikationen und Datenströme lassen sich Fehlentwicklungen frühzeitig erkennen und korrigieren. ■



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>

IOS-APPS MIT HANDOFF-FUNKTION

Hands off

Mit dem Handoff API lassen sich geräteübergreifende Bearbeitungsfunktionen für Apps erstellen.

Wäre diese Vorstellung nicht schön? Sie sind mit Ihrem Smartphone unterwegs, zu Fuß natürlich, und schreiben dabei eine Mail. Irgendwann kommen Sie im Büro an, greifen zum Tablet oder Desktop-PC und können die auf dem Smartphone angefangene Mail auf dem folgenden Gerät nahtlos fortschreiben. Ein Wunschtraum?

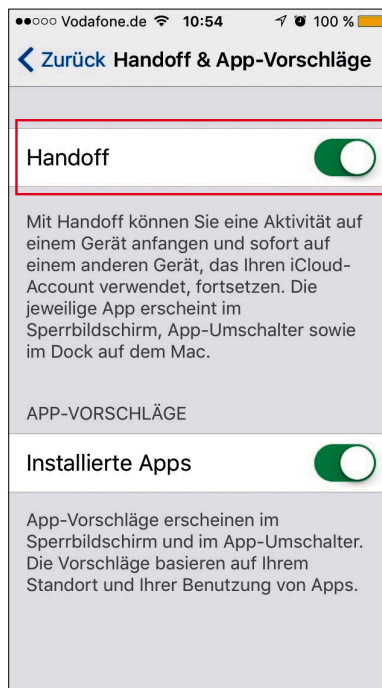
Nein – mit relativ aktuellen Apple-Geräten der i-Generation – ab iPhone 5, iPad 4. Generation, iPod touch 5. Generation – beziehungsweise einem Mac ab Mitte 2012 ist diese Funktion Wirklichkeit und kann genutzt werden.

Neben den Voraussetzungen im Bereich der Hardware gibt es natürlich auch welche für die erforderliche Software. Ein iDevice benötigt mindestens iOS 8.x. Sollen auch Daten an einen Mac übergaben werden, so muss dort mindestens Yosemite installiert sein.

Einrichtung von Handoff

Die Handoff-Funktion wird aktiviert, indem die *Einstellungen*-App aufgerufen wird. Dort muss dann der Punkt *Allgemein* geöffnet werden. Schaut man dann etwas weiter unten in der Liste nach, so findet man dort den Menüpunkt *Handoff & App-Vorschläge*. Dieser Punkt muss geöffnet werden. Zwei Optionen sind hier zu finden. *Handoff* und *Installierte Apps* (Bild 1). Um Handoff zu aktivieren, muss nur der entsprechende Switch betätigt werden.

Anschließend ist Handoff auf dem Gerät aktiv, sofern es unterstützt wird. Außerdem ist es erforderlich, dass sich die Ge-



Die Handoff-Funktion unter iOS aktivieren (Bild 1)

räte zum Datenaustausch im selben drahtlosen Netzwerk befinden. Die Unterstützung hängt nicht nur vom Betriebssystem und der Hardware ab, sondern natürlich auch von der App selbst. So gibt es eine entsprechende Unterstützung beispielsweise bei der Mail-App und der SMS-App von iOS.

Neue Klasse bereitgestellt

Damit auch iOS-Entwickler problemlos die Handoff-Funktion in die eigenen Apps integrieren können, wurde mit iOS 8 eine neue Klasse von Apple bereitgestellt.

Bei der Implementierung hatte Apple im Hinterkopf, dass jeder Aktion innerhalb einer App aus mindestens einer Aktivität besteht. Die Aktivität sowie ein zugehöriges Benutzerkonto (iCloud-Account) bilden dann eine Einheit. Mittels des Benutzerkontos kann dann eine oder auch mehrere Aktivitäten geräteübergreifend zugeordnet werden.

Voraussetzung hierfür ist natürlich, dass der Benutzer sich an beiden Geräten mit demselben iCloud-Account angemeldet hat. Zuletzt muss dann nur noch die Aktivität selbst von Gerät zu Gerät übertragen werden. Basis für die Übertragung dieser Aktivität ist die Klasse *NSUserActivity*.

Wird eine Aktivität über eine App hinaus – also von mehreren Apps – verwendet, dann ist es zusätzlich erforderlich, dass die App in Xcode mit einem Team Identifier (Team ID) gekennzeichnet wurde.

Key	Type	Value
Information Property List	Dictionary	(17 items)
NSUserActivityTypes	Array	(1 item)
Item 0	String	de.christian.bleske.note
Application Category	String	
Localization native development re...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)

Ein neuer Activity-Type wird angelegt (Bild 2)

Die Arbeit mit Handoff kann in drei Funktionen münden. Die erste wäre, dass eine neue `UserActivity` zum Datenaustausch erzeugt und von einem Gerät zum anderen übertragen wird. Die zweite Möglichkeit beinhaltet die Aktualisierung einer bereits vorhandenen beziehungsweise erzeugten `UserActivity`. Die letzte Aktion beinhaltet die Fortführung einer `UserActivity` auf einem anderen Gerät.

Jede `UserActivity` wird durch einen `ActivityType` identifiziert. Hierbei handelt es sich um einen simplen String, der allerdings eindeutig sein muss. Dieser String wird in der Datei *Info.plist* eines Projekts hinterlegt.

Achten Sie also darauf, dass sich bei der Definition von mehreren `UserActivities` diese durch den Identifier leicht unterscheiden lassen. Apple empfiehlt, zur Namensgebung auf die Reverse Domain Name Notation, kurz Reverse DNS, zurückzugreifen. Im Beispiel habe ich die `UserActivity` unter Beachtung dieser Empfehlung daher folgendermaßen benannt: *de.christian.bleske.note*.

Allgemeine Empfehlungen

Hier noch einige allgemeine Empfehlungen: Generell gilt, dass die Menge der Daten, die mittels der Handoff-Funktion übertragen werden sollen, so klein wie möglich sein sollte. Je

größer die Datenmenge ist, desto weniger flüssig ist der Prozess der Fortführung der Arbeit, weil gegebenenfalls erst gewartet werden muss, bis die Daten vollständig von einem zum anderen Gerät übertragen worden sind.

Müssen doch einmal größere Datenmengen übertragen werden, so ist es möglich, hierfür einen speziellen Streaming-Mechanismus zu verwenden, der auch von Handoff unterstützt wird.

Projekt einrichten

Das Beispielprojekt soll es ermöglichen, eine kleine Notiz, die auf einem Gerät – zum Beispiel dem iPhone – begonnen wurde, auf einem anderen Gerät – zum Beispiel dem iPad – fortzuschreiben. Als Projektvorlage wurde eine Single View Application gewählt.

Nach dem Anlegen des Projekts sollte zuerst in der Datei *Info.plist* der `ActivityType` eingerichtet werden. Durch einen Doppelklick in der Projektübersicht lässt sich die *Info.plist*-Datei im entsprechenden Editor öffnen. Innerhalb der *Information Property List* sollte nun ein neuer Eintrag angelegt werden. Dieser trägt die Bezeichnung *NSUserActivityTypes*. Unterhalb des neuen Punkts kann dann der oder die neuen Activity-Typen angelegt werden. ►

Listing 1: Quellcode des Beispiels

```
import UIKit
class ViewController: UIViewController,
    UITextViewDelegate {

    @IBOutlet var uiTextView: UITextView!
    override func viewDidLoad() {
        super.viewDidLoad()
        uiTextView.delegate = self;
        createUserActivity();
    }
    func createUserActivity() {
        let nsUserActivity = NSUserActivity
            (activityType: "de.christian.bleske.note")
        nsUserActivity.title = "Notiz"
        nsUserActivity.userInfo = ["content": ""]
        userActivity = nsUserActivity
        userActivity?.becomeCurrent()
    }

    override func
        updateUserActivityState(
            activity: NSUserActivity)
        {
            activity.AddUserInfoEntriesFromDictionary
                (["content": self.uiTextView.text])
            print("Update UserActivityState...")
            print(activity.userInfo?["content"] as! String)
            super.updateUserActivityState(activity)
        }
    }

    func textField(textField: UITextField,
        shouldChangeCharactersInRange range:
            NSRange, replacementString string:
                String) -> Bool {
        self.updateUserActivityState(userActivity!)
        return true
    }

    func textView(textView: UITextView,
        shouldChangeTextInRange range:NSRange,
        replacementText text:String) -> Bool {
        self.updateUserActivityState(userActivity!)
        return true
    }

    override func
        restoreUserActivityState(activity: NSUserActivity) {
            print("Restore UserActivity...")
            print(activity.userInfo?["content"] as! String)

            self.uiTextView.text =
                activity.userInfo?["content"] as! String
        }

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


Im Beispiel wird nur einer angelegt: *de.christian.bleske.note*. Achten Sie beim Anlegen darauf, dass die Bezeichnung des neuen Typs in der Eigenschaft *Value* angelegt wird (Bild 2).

Anschließend wird das GUI der App gestaltet. Hier genügt es, ein *UITextField*-Control in den *ViewController* sowie ein *Label*-Control einzufügen. Im folgenden Schritt kann man mit der Codierung beginnen.

Activity anlegen

Eine Activity beziehungsweise der Inhalt einer Notiz soll im Beispiel übertragen werden, sobald der Anwender ein neues Gerät in Gebrauch nimmt und die entsprechende App auf dem neuen Gerät öffnet. Es müssen also nicht nur Daten übertragen werden, sondern dies muss auch zu einem bestimmten Zeitpunkt passieren. Listing 1 enthält die hierfür verwendete Programmlogik. Die Klasse *ViewController* wird von *UIViewController* abgeleitet und implementiert den *UITextFieldDelegate*.

Los geht es mit dem Anlegen des Outlets für das *TextField*-Control. Innerhalb der Methode *viewDidLoad* wird dann der *Delegate* für das *UITextField*-Control zugewiesen und außerdem die Methode *createActivity* aufgerufen. Innerhalb der Methode *createActivity* erfolgt schließlich das Anlegen der neuen Activity. Hierzu wird der Konstruktor der Klasse *NSUserActivity* unter Angabe des *ActivityTypes* (in Form eines Strings) aufgerufen. Hierbei handelt es sich natürlich um den zuvor in der *Info.plist*-Datei definierten Typ. Anschließend erhält die neue *NSUserActivity*-Instanz über die Eigenschaft *title* eine Bezeichnung zugewiesen.

Schlüssel und Datum werden übergeben

Es folgt eine Zuweisung an die Eigenschaft *UserInfo*. Dieser Eigenschaft kommt eine zentrale Bedeutung zu. Es handelt sich hierbei um ein Dictionary, dem ein Schlüssel und das zugehörige Datum übergeben werden. Bei dem Datum handelt es sich um die Information, die via *UserActivity* weitergegeben werden soll. Zuletzt wird die initialisierte Activity der global de-

klarierten Instanz *userActivity* zugewiesen und mittels eines Aufrufs der Methode *becomeCurrent* zur aktiven Activity gemacht.

Update und Ereignisse

Die *UserActivity*-Instanz ist nun vorhanden und initialisiert. Nun kann der Code für die Steuerung der Aktualisierung sowie der Übertragung der Information implementiert werden. Im Quellcode folgt die Methode *updateUserActivityState*.

Dieser Methode wird eine *UserActivity*-Instanz als Parameter übergeben. Nachdem Aufruf erfolgt die Übergabe des Inhalts an die *UserActivity*-Instanz. Hierzu bedient man sich der Methode *addUserInfoEntriesFromDictionary*.

Das bekannte Schema wird hier erneut genutzt, und unter Verwendung des Keys *content* wird der neue Inhalt übergeben. Die zusätzlichen Print-Anweisungen dienen der besseren Nachvollziehbarkeit des Beispiels (Bild 3).

Im Quellcode geht es weiter mit der Methode *shouldChangeCharactersInRange*. Sobald sich am Inhalt des *TextFields* etwas ändert, wird diese Methode aufgerufen und somit erneut die Methode *updateUserActivityState*. Auch die Methode *shouldChangeTextInRange* wird aus demselben Grund aufgerufen. Die letzte Methode innerhalb der Klasse ist *restoreUserActivityState*.

Diese Methode wird aufgerufen, wenn der Inhalt eines *UserActivity* wieder ausgegeben werden soll. Im Beispiel wird der Inhalt durch Abruf des Dictionarys und Übergabe der Daten an das *UITextField*-Control vorgenommen.

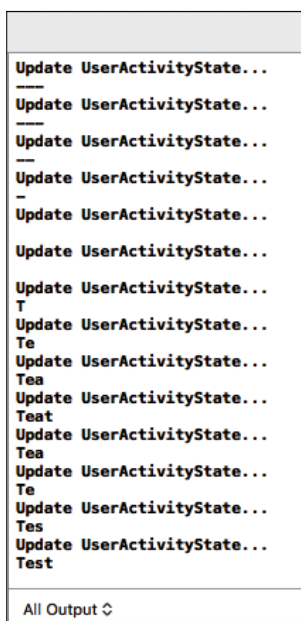
Auch hier wird abermals die Print-Anweisung genutzt, um den Programmverlauf zu verdeutlichen. Nach Eingabe des Inhalts wird der Text auf das zweite mit iOS oder OS X betriebene Gerät automatisch übertragen, sobald dort die entsprechende App geöffnet wird (Bild 4).

Fazit

Mittels der Klasse *NSUserActivity* können Informationen von einem iDevice auf ein anderes übertragen und dort dann weiterbearbeitet werden. Eine überaus praktische Erweiterung für die eigene App. ■



Eingabe von Text in der Test-App (Bild 4)



Print-Ausgaben: Das Beispiel erzeugt Print-Ausgaben zum Nachvollziehen (Bild 3)



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

Update für Ihr Know-How

„Die Faszination der Software-entwicklung liegt darin begründet, dass nur die eigene Fantasie und Kreativität begrenzen, was möglich ist.“

Golo Roden
Technologischer Visionär, Sprecher, Autor



Hybrid-Apps mit Ionic, Cordova und AngularJS

Trainer: Hendrik Lösch

2 Tage, 09.-10.05.2016, Köln

Frühbucher: 1.799,- EUR zzgl. MwSt.
Normaltarif: 1.999,- EUR zzgl. MwSt.



Cross-Plattform-Apps mit C# und Xamarin

Trainer: Sebastian Seidel

3 Tage, 10.-12.05.2016, Köln

Frühbucher: 2.199,- EUR zzgl. MwSt.
Normaltarif: 2.399,- EUR zzgl. MwSt.



Webanwendungen mit HTML5, CSS3 und JavaScript

Trainer: David Tielke

2 Tage, 11.-12.07.2016, Köln

Frühbucher: 1.799,- EUR zzgl. MwSt.
Normaltarif: 1.999,- EUR zzgl. MwSt.



Webentwicklung mit ASP.NET, MVC und Web API

Trainer: David Tielke

3 Tage, 13.-15.07.2016, Köln

Frühbucher: 2.199,- EUR zzgl. MwSt.
Normaltarif: 2.399,- EUR zzgl. MwSt.



Ihr Ansprechpartner:

Fernando Schneider – Key Account Manager – developer media

Telefon: +49 (0)89 74117-831 – E-Mail: fernando.schneider@developer-media.de

OPENGL FÜR QUEREINSTEIGER

Mathematisches Wissen

Für Android 3D-Spiele unter OpenGL zu programmieren kann lehrreich sein.

Wer heute 3D-Spiele für Android programmiert, greift zu fertigen Engines wie Unity und Co. Das mag aus kommerzieller Sicht durchaus sinnvoll sein – beim Programmieren unter OpenGL lernt man jedoch wesentlich mehr.

Die Programmierschnittstelle genießt eine nicht sonderlich gute Reputation: Im Vergleich zu Managed Frameworks wie XNA und Co. ist wesentlich mehr mathematisches Wissen notwendig. Zudem ist OpenGL wesentlich näher an der Hardware als die diversen Frameworks. Dies sorgt für höheren Lernaufwand, ermöglicht aber auch besseren Zugriff und bessere Auslastung der Hardwarekomponenten.

Unterschiedliche Versionen

Android unterstützt anstelle des normalen OpenGL eine abgespeckte Version namens OpenGL ES. Zum Zeitpunkt der Drucklegung gibt es die folgenden fünf Versionen:

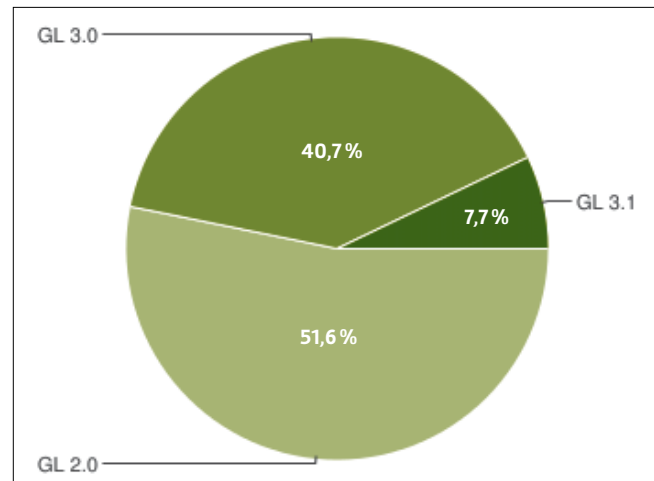
- OpenGL ES 1.0,
- OpenGL ES 1.1 (mit Shadererweiterungen),
- OpenGL ES 2.0,
- OpenGL ES 3.0,
- OpenGL ES 3.1.

Der Unterschied zwischen OpenGL ES 1 und ES 2 ist signifikant: Nur die zweite Version unterstützt Shader, die die Realisierung fortgeschrittener Grafikeffekte ermöglichen. Erfreulicherweise spielen Telefone auf Basis von OpenGL ES 1.0 heute keine große Rolle mehr.

Googles offizielle, unter <http://developer.android.com/about/dashboards> einsehbare Liste präsentierte zum Zeitpunkt der Drucklegung dieser Heftausgabe die in Bild 1 gezeigte Marktverteilung.

Listing 1: content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.LinearLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:app="http://schemas.android.com/
apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tamoggemon.nmgopengles.
    MainActivity">
</android.support.design.widget.LinearLayout>
```



OpenGL ES 1.0 wird nicht mehr aufgelistet (Bild 1)

Wir wollen unsere Arbeit mit Android Studio beginnen. Erzeugen Sie ein neues Projekt und löschen Sie die Datei *content_main.xml*. Der Inhalt von *activity_main* wird durch das in Listing 1 gezeigte Minimallayout ersetzt.

Frustbewertungen wie die in Bild 2 gezeigte lassen sich dadurch vermeiden, dass man den Play Store mit Informationen über die erforderliche Hardware versorgt. Google zeigt die App in diesem Fall nur auf jenen Smartphones an, auf denen die notwendige Hardware vorhanden ist. Im Fall von OpenGL ES 2.0 sieht die Änderung in *AndroidManifest.xml* wie in Listing 2 aus.

Im nächsten Schritt müssen wir die Activity anpassen, um eine Zeichenfläche zu schaffen. Öffnen Sie dazu die Klasse *MainActivity.java* und passen Sie den Konstruktor nach folgendem Schema an:

```
public class MainActivity extends AppCompatActivity {
    private GLSurfaceView myGLView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        myGLView = new GLSurfaceView(this);
        myGLView.setEGLContextClientVersion(2);
        myGLView.setPreserveEGLContextOnPause(true);
        myGLView.setRenderer(new NMGRenderer());
        setContentView(myGLView);
    }
}
```

GLSurfaceView ist eine vom Android-Betriebssystem zur Verfügung gestellte Convenience-Klasse, die bei der Realisie-

nung von OpenGL-basierten Applikationen hilft. Ihr markantes Feature ist das Bereitstellen eines eigenen Threads, der für das Abarbeiten des Rendering-Codes zuständig ist. Dadurch wird der unter Android heilige Hauptthread nicht belastet, ANR-Dialoge treten nicht auf.

Da die Klasse *GLSurfaceView* für alle Versionen von OpenGL ES gleichermaßen geeignet ist, setzen wir die zu verwendende Version des Frameworks durch Aufruf von *setEGLContextClientVersion*. Die Zuweisung der Klasse *NMGRenderer* wird im nächsten Abschnitt besprochen.

Lebenszyklus von Android-Activities

GLSurfaceView bindet sich in den Lebenszyklus von Android-Activities ein. Leider kann das Steuerelement die Ereignisse nicht selbstständig erkennen, weshalb zwei Handler für *onResume* und *onPause* notwendig sind:

```
@Override
protected void onResume() {
    super.onResume();
    if (myGLView != null) {
        myGLView.onResume();
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (myGLView != null) {
```



Kristin Mercer August 12, 2013

★★★★★

Sucks It wouldn't let me even go on the app it said 'sorry but gotta text has stopped working please try again later' and I tried later many many times and still the same thing happens I say don't waist ur time downloading this stupid app uninstalling:-(really wanted to try this app :-(

UnME2, Inc. August 16, 2013

With all due respect, you attempted to install gottaTxt on a NOOK. gottaTxt is a text messaging system which is location aware, and the NOOK does not support SMS or GPS. No piece of software can make a NOOK send text messages or know where it is. It seems mean-spirited to write that the app 'Sucks' when your device lacks necessary hardware.

Eigentlich offensichtlich: Ein SMS-Programm funktioniert auf einem reinen Tablet ohne Funkmodul nicht (Bild 2)

Listing 2: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tamoggemon.nmgopengles">
    <uses-feature android:glEsVersion="0x00020000"
        android:required="true" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name">
```

```
        myGLView.onPause();
    }
}
```

Sollten Sie in Ihrem Code mehrere Varianten des Renderpfads implementieren wollen, so müssen Sie die Information über die unterstützte OpenGL-ES-Variante zur Laufzeit ermitteln. Dies lässt sich nach dem in der Funktion *hasGLES20* gezeigten Schema bewerkstelligen:

```
private boolean hasGLES20() {
    ActivityManager anAM = (ActivityManager)
        getSystemService(getApplicationContext().
            ACTIVITY_SERVICE);
    ConfigurationInfo info =
        anAM.getDeviceConfigurationInfo();
    return info.reqGlEsVersion >= 0x20000;
}
```

Unsere *GLSurfaceView* stellt eine Zeichenfläche zur Verfügung – die eigentliche Darstellungsintelligenz müssen wir selbst beitragen. Dazu ist in Android eine Klasse verantwortlich, die von *GLSurfaceView.Renderer* abzuleiten ist. In Android Studio ist man mit folgender Struktur konfrontiert:

```
public class NMGRenderer implements
    GLSurfaceView.Renderer {
    @Override
    public void onSurfaceCreated(GL10 gl,
        EGLConfig config) {
    }
    @Override
    public void onSurfaceChanged(GL10 gl, int width,
        int height) {
    }
    @Override
    public void onDrawFrame(GL10 gl) {
    }
}
```

Die Funktion *onSurfaceCreated* wird immer dann aufgerufen, wenn der OpenGL-Kontext erstmals entsteht. Sie ►

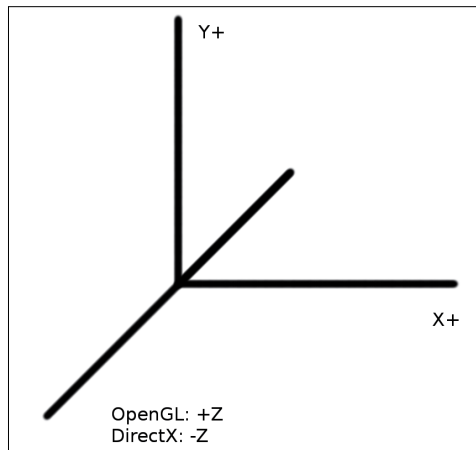
dient zur Einrichtung von diversen OpenGL-Parametern und dem Setzen von Zuständen in der Grafik-Pipeline. *onSurfaceChanged* wird immer dann aufgerufen, wenn sich die Größe des Viewports ändert. Das klassische Beispiel dafür wäre das Ändern der Ausrichtung des Smartphones. Bei einem Wechsel von Portrait- und Landscape-Modus verändert sich die Größe der View natürlicherweise, was im Rendering Berücksichtigung finden sollte.

Die meisten Spiele nutzen den Activity-Manager zum Blockieren der Ausrichtung. Es ist sehr schwer möglich, ein Spiel zu realisieren, das im Portrait- und Landscape-Modus gleich gut aussieht.

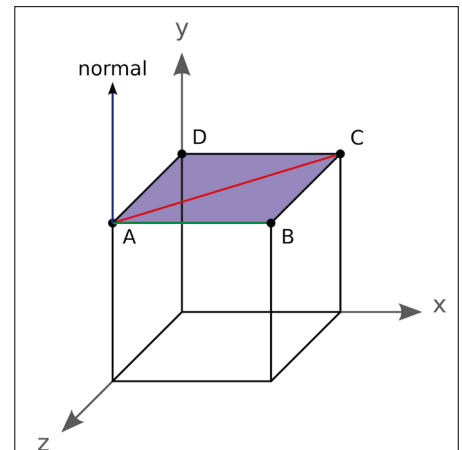
Die eigentliche Zeichenintelligenz findet sich in der Methode *onDrawFrame*. Sie wird vom Betriebssystem, je nach Einstellung, entweder periodisch oder nach Aufforderung aufgerufen und animiert die Plattform zur Erzeugung neuer Grafiken. Aufmerksame Leser fragen sich an dieser Stelle, welche Rolle die Variable *GL10* spielt. Das Interface wird in Android für OpenGL 1, 2 und 3 verwendet. Während OpenGL 2 und 3 mit einer statischen Klasse arbeiten, arbeitet OpenGL ES 1 mit Methoden und dem hier als 1.0 bezeichneten GL-Parameter.

Von Inhalten und Vertizes

Die meisten Grafikbeispiele beginnen mit der Darstellung von zweidimensionalen Objekten. Wir starten hier sofort mit einem dreidimensionalen Element. Auf diese Art und Weise können wir die Mathematik sofort en bloc abwickeln und uns in der nächsten Ausgabe einer schwierigeren Aufgabe zuwenden. Dreidimensionale Objekte leben in einem als Modellkoordinatensystem bezeichneten Raum, dessen drei Achsen normalerweise von eins bis minus eins reichen (Bild 3).



Das OpenGL- und das XNA-Koordinatensystem sind unterschiedlich (Bild 3)



Kombination: Zwei Dreiecke ergeben bei OpenGL ein Viereck (Bild 4)

Wikimedia Commons / Gsliepen

Ein Modell besteht aus einer Sammlung von Punkten, die in der Sprache der 3D-Grafik als Vertices bezeichnet werden. Der Grafikchip fasst je drei Vertices zu einem Dreieck zusammen, das daraufhin durch einen im folgenden Abschnitt im Detail besprochenen Prozess auf den Bildschirm gebracht wird.

OpenGL arbeitet so gut wie immer mit Floats: Modelle liegen in Form einer Punktwolke aus Floats vor, die von der 3D-Engine in Dreierschritten abgearbeitet wird:

```
float[] cubePositionData = {
    //Seite 1
    -1.0f, 1.0f, 1.0f,
    -1.0f, -1.0f, 1.0f,
    1.0f, 1.0f, 1.0f,
    -1.0f, -1.0f, 1.0f,
    1.0f, -1.0f, 1.0f,
    1.0f, 1.0f, 1.0f,
}
```

Aufmerksame Beobachter fragen sich an dieser Stelle, wieso wir für die aus vier Punkten bestehende erste Seite sechs Punkte benötigen. Der Grund dafür ist, dass OpenGL prinzi-

Listing 3: Deklaration

//Seite 2	-1.0f, -1.0f, -1.0f,	1.0f, 1.0f, -1.0f,
1.0f, 1.0f, 1.0f,	-1.0f, 1.0f, -1.0f,	-1.0f, 1.0f, 1.0f,
1.0f, -1.0f, 1.0f,	//Seite 4	1.0f, 1.0f, 1.0f,
1.0f, 1.0f, -1.0f,	-1.0f, 1.0f, -1.0f,	1.0f, 1.0f, -1.0f,
1.0f, -1.0f, 1.0f,	-1.0f, -1.0f, -1.0f,	//Seite 6
1.0f, -1.0f, -1.0f,	-1.0f, 1.0f, 1.0f,	1.0f, -1.0f, -1.0f,
1.0f, 1.0f, -1.0f,	-1.0f, -1.0f, -1.0f,	1.0f, -1.0f, 1.0f,
//Seite 3	-1.0f, -1.0f, 1.0f,	-1.0f, -1.0f, -1.0f,
1.0f, 1.0f, -1.0f,	-1.0f, 1.0f, 1.0f,	1.0f, -1.0f, 1.0f,
1.0f, -1.0f, -1.0f,	//Seite 5	-1.0f, -1.0f, 1.0f,
-1.0f, 1.0f, -1.0f,	-1.0f, 1.0f, -1.0f,	-1.0f, -1.0f, -1.0f,
1.0f, -1.0f, -1.0f,	-1.0f, 1.0f, 1.0f,	};

piell mit Dreiecken arbeitet. Ein Viereck entsteht aus zwei Dreiecken, die an ihrer Hypotenuse aneinandergefügt sind (Bild 4). Mit diesem Wissen können wir uns den restlichen Seiten zuwenden, deren Deklaration Listing 3 zeigt. Wer mit der räumlichen Vorstellung Probleme hat, sollte sich mit einer Handzeichnung behelfen. Erfahrungsgemäß brauchen auch Personen mit sehr gutem räumlichen Vorstellungsvermögen beim ersten Durchlauf etwas Hilfe.

Komplexe Geometrien

Das manuelle Erstellen von Modellen spielt in der Praxis eine sehr untergeordnete Rolle: Komplexe Geometrien werden so gut wie immer von 3D-Grafikern in dedizierten Programmen erzeugt und zur Laufzeit mit speziellen Bibliotheken geladen. Wir besprechen diese Vorgehensweise in einer der nächsten Ausgaben. Im Moment besteht unsere Aufgabe darin, die nun im Speicher der Java-VM vorliegenden Informationen in ein für das auf C basierende OpenGL-API verständliches Format umzuwandeln.

Dies erfolgt durch Nutzung einer Instanz der Klasse *Float-Buffer*, die durch den in im Folgenden gezeigten Code bevölkert wird:

```
public class NMGRenderer implements
GLSurfaceView.Renderer {
    FloatBuffer myCubePositions;
    public NMGRenderer() {
        float[] cubePositionData = {
            ...
            myCubePositions = ByteBuffer.allocateDirect
            (cubePositionData.length * 4).order
            (ByteOrder.nativeOrder()).asFloatBuffer();
            myCubePositions.put(cubePositionData).position(0);
        }
    }
}
```

Da Java keinen *SizeOf*-Operator mitbringt, nehmen wir für Float eine Länge von vier Bytes an. Im praktischen Code wird diese Zahl oft in eine Konstantenklasse ausgelagert, um die Übersichtlichkeit zu erhöhen – Strings sind klarer als Zahlen.

Schon aus Gründen der Didaktik wollen wir unseren Würfel mit Farbinformationen versehen. In der Welt der Grafikprogrammierung sind Farbdaten so gut wie immer vierwertig: Neben den drei additiven Komponenten: R für rot, G für Grün und B für Blau gibt es mit A auch einen Wert, der den Grad der Transparenz beziehungsweise Sättigung beschreibt. Das Farb-Array für unseren Würfel sieht folgendermaßen aus:

```
float[] cubeColorData = {
    1.0f, 0.0f, 0.0f, 1.0f,
    ...
}
```

```
0.0f, 1.0f, 0.0f, 1.0f,
...
0.0f, 0.0f, 1.0f, 1.0f,
...
1.0f, 1.0f, 0.0f, 1.0f,
...
0.0f, 1.0f, 1.0f, 1.0f,
...
1.0f, 0.0f, 1.0f, 1.0f,
...
};
```

Materialfarben erlauben das Zuweisen von Farben an einzelne Vertices. Unser Listing ist insofern stark gekürzt, als es nur die Farben des ersten Vertex abdruckt. Die durch ... ersetzten Zeilen sind jeweils fünf weitere Kopien der zuvorgehenden Zeile. Auch das für das Vorhalten der Farbinformationen zuständige Array hat am Ende 36 Elemente. Seine Konversion in einen Float-Buffer erfolgt nach dem schon von der anderen Methode bekannten Schema und wird hier aus Platzgründen nicht nochmals abgedruckt.

Schritt für Schritt zum Ziel

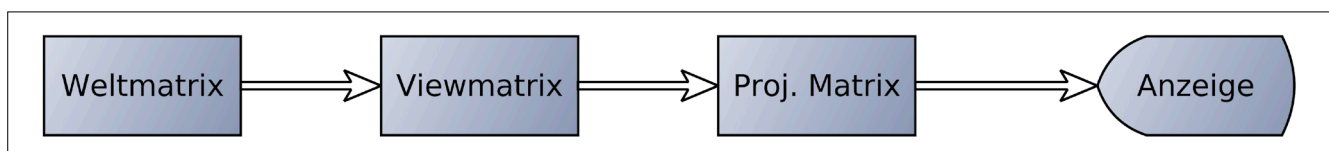
Vor dem Weiterprogrammieren müssen wir kurz innehalten und überlegen, was wir bisher realisiert haben. Unser darzustellender Würfel liegt nun in Form einer Punktwolke vor, die für den Grafikchip verständlich und ansprechbar ist.

Zumindestens ist sie dies in der Theorie. Bei sorgfältiger Betrachtung fällt auf, dass der Würfel derzeit erstens dreidimensional und zweitens im Zahlenraum von minus eins bis plus eins beheimatet ist.

Bildschirme sind zweidimensional. Im Fall des Smartphones des Autors weisen sie eine Auflösung von 1920 x 1080 auf. Trotz der sehr leistungsstarken GPU kann das LCD-Panel mit dreidimensionalen Koordinaten nichts anfangen. Daraus folgt, dass die Vertices vor der Darstellung in den Bildschirmkoordinatenraum transformiert werden. Es handelt sich dabei um eine vergleichsweise komplizierte Prozedur, die als Multiplikation mit drei Matrizen dargestellt werden kann (Bild 5).

Die genaue mathematische Implementierung tritt gegenüber der Rolle der einzelnen Matrizen in den Hintergrund. Zum einfacheren Verständnis wollen wir den Lebenszyklus unseres Würfels analysieren, der im Moment in der Mitte des Modellkoordinatensystems sitzt.

Die Weltmatrix hat die Aufgabe, den im Modellkoordinatensystem befindlichen Würfel an seinen Platz im Raum zu verschieben und ihn dabei zu skalieren beziehungsweise zu rotieren. Wollen wir unseren Würfel beispielsweise an den ►



Drei Matrizen erledigen die Transformation für den Entwickler (Bild 5)

Punkt -20/20/20 bringen und in der Größe 5x5 darstellen, so finden sich diese Parameter in der Weltmatrix wieder. Änderungen am Aufenthaltsort und der Größe der Objekte setzen somit immer Änderungen der Weltmatrix voraus.

Im nächsten Schritt kommt die als Viewmatrix bezeichnete Zahlengruppe zum Einsatz. Die eigentliche Darstellung der Objekte erfolgt nicht im Modell- beziehungsweise Szenenkoordinatensystem, sondern in einem speziellen Koordinatensystem, an dessen Ursprung ein als Kamera bezeichnetes virtuelles Objekt steht.

Stellen Sie sich die Kamera durchaus einmal als ein menschliches Auge vor, das an der vorgegebenen Position steht, freudig in die Szene guckt und seine Perspektive an die GPU weiterleitet.

Dreidimensionaler Modellraum

Die View-Matrix hat die Aufgabe, die diversen sich im dreidimensionalen Modellraum befindlichen Koordinaten zwecks einfacherer Darstellbarkeit in ein Koordinatensystem zu übertragen, in dessen Ursprung das jeweilige Auge steht. Daraus folgt, dass Änderungen der Blickrichtung oder der Kameraposition stets über die Viewmatrix erfolgen.

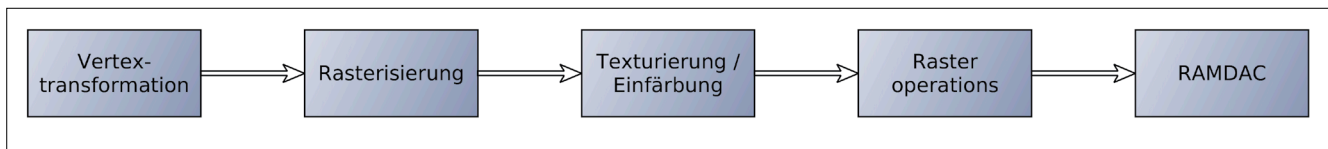
Die Projection-Matrix ist dann dafür zuständig, die dreidimensionalen Informationen im Rahmen eines Projektionsver-

dass sich statt der Schritte Vertextransformation und Einfärbung ganze Funktionsblöcke verstecken.

Programmierbare Intelligenz

GPUs sind von der inneren Architektur her wesentlich simpler als klassische Prozessoren: Die auf die Verarbeitung großer Datenmengen optimierten Rechenelemente wären bei Programmierung in klassischem C nicht sonderlich effizient. OpenGL begegnet diesem Problem durch die Einführung der GLSL. Es handelt sich dabei um eine von der Syntax her an C angelehnte Programmiersprache, die die Erzeugung von Shadercode ermöglicht.

Die innere Architektur von GPUs unterscheidet sich von Hersteller zu Hersteller. Jeder Hersteller braut auf der Suche nach optimaler Performance sein eigenes Süppchen. Aus dieser Situation ergibt sich ein Problem für Entwickler von Grafik-APIs: Wer seinen Shadercode auf der Workstation kompiliert, nimmt dem Hardwarehersteller die Möglichkeit für Optimierungen. Aus diesem Grund erfolgt die Kompilation des Codes erst direkt vor seiner Nutzung. Die Anbieter der GPU stellen ihre Treiber im Rahmen der Implementierung von OpenGL mit einem vollwertigen Compiler aus, der die als Zeichenketten angelieferten Shader dynamisch in ein für die jeweilige GPU optimales Format bringt.



Die Graphics-Pipeline ist von eminentester Bedeutung (Bild 6)

fahrens auf zweidimensionale Koordinaten zu reduzieren. Diese mathematischen Prozesse begründen den ersten Schritt des als Graphics-Pipeline bezeichneten Prozesses (Bild 6).

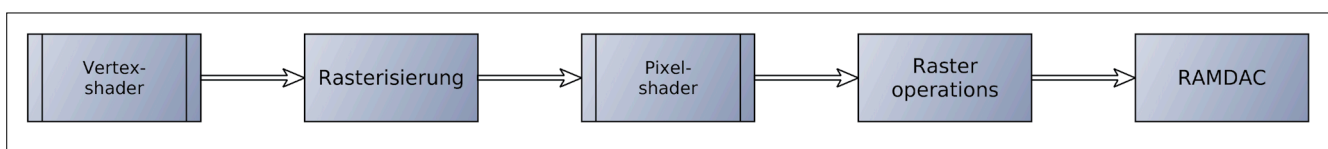
Nach der Transformation aller Vertices werden die bereits transformierten Vertices analysiert. Der Sinn dieser Analyse ist die Eliminierung von überflüssigen beziehungsweise nicht sichtbaren Elementen. Im nächsten Schritt erfolgt die Einfärbung der Pixel, die daraufhin im Rahmen von Raster-Operations ins Render-Target wandern.

In der Anfangszeit der Computergrafik – im Handcomputer war dies bis inclusive OpenGL ES 1.1 der Fall – hatte man es mit sogenannten Fixed-Function-Pipelines zu tun, bei denen diese Funktionen zur Gänze in Hardware abgebildet waren. Bild 7 zeigt zum Unterschied dazu die programmierbare Pipeline: Kenner von Flussdiagrammen stellen sofort fest,

Daraus ergibt sich für Entwickler ein kleines Ärgernis: Das erfolgreiche Durchlaufen des Kompilationsprozesses bedeutet noch nicht, dass die Applikation auch wirklich funktionieren wird. Eine sinnvolle Gegenmaßnahme ist die Nutzung spezieller Debugbuilds, die im Rahmen des Startvorgangs alle Shader kompilieren und beim Auftreten eines Fehlers sofort eine Warnung anzeigen.

Aus der Konserve

Auch wenn es in Tutorials gern gezeigt wird: Das direkte Einbetten des Shadercodes in String-Variablen ist eine Pest. Shader sind Ressourcen, die vom Android-Ressourcensystem bereitgestellt werden. Wer es anders macht, riskiert Encodings- und sonstige Probleme. Versetzen Sie die Projektübersicht von Android Studio in den Android-Modus. Klicken Sie



Die Programmable Graphics Pipeline ist wesentlich flexibler (Bild 7)

im nächsten Schritt das *Res*-Verzeichnis an und wählen Sie im daraufhin erscheinenden Kontextmenü die Option *New* und *Android Resource Folder* aus. Erzeugen Sie sodann einen Ordner namens *raw*, der Ressourcen vom Typ *Raw* enthält (Bild 8).

Klicken Sie den leeren Ordner im nächsten Schritt rechts an, um eine neue Textdatei namens *vertexshader.txt* anzulegen. Nach einem Durchlauf von *Build* und *Rebuild Project* sammelt das Ressourcensystem die Inhalte automatisch ein. Es ist Zeit, die Datei mit folgendem Shader zu bevölkern:

```
uniform mat4 u_MVPMatrix;
attribute vec4 a_Position;
attribute vec4 a_Color;
varying vec4 v_Color;
```

Vertexshader haben die Aufgabe, die weiter oben beschriebenen Matrixmultiplikationen in die Praxis umzusetzen. In OpenGL bestehen sie aus der Variablendefinition und einer oder mehreren Rechenfunktionen sowie dem als *main()* bezeichneten Einsprungpunkt. Variablen werden nach ihrer Verwendung katalogisiert: Mit dem Attribut *uniform* eingeleitete Felder bleiben während der Rendering eines gesamten Elements konstant. Da unser Shader Modelle en bloc abarbeitet, können wir die Welt-, View- und Projektionsmatrizen mit dieser ressourcensparenden Eigenschaft ausstatten.

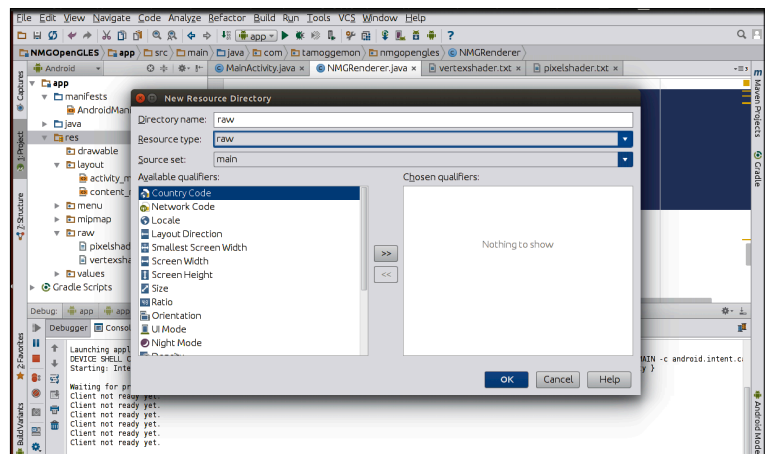
Attribute-Variablen enthalten Informationen, die sich bei jedem einzelnen Vertex ändern. Das Attribut ist insofern relevant, als eine als *attribute* definierte Variable in den meisten OpenGL-Implementierungen nur vom Vertexshader angesprochen werden kann. Das Attribut *varying* dient derweil zur Beschreibung von Feldern, die für beide Shader gleichermaßen relevant sind.

Damit können wir uns dem zweiten Teil des Vertexshaders zuwenden, der die eingehenden Vertizes mit der MVP-Matrix multipliziert und das Ergebnis in *gl_Position* schreibt. Es handelt sich dabei um eine magische Variable, deren Inhalt als Ergebnis der Vertextransformation angesehen wird. Zudem bevölkern wir *v_Color*, um die in *a_Color* liegenden Informationen an den Pixelshader weiterzuleiten:

```
void main() {
    v_Color = a_Color;
    gl_Position = u_MVPMatrix * a_Position;
}
```

Schon aus Vollständigkeitsgründen sei angemerkt, dass sich im XNA- und DirectX-Bereich eine andere Vorgehensweise etabliert hat. Als Beispiel dafür dient der folgende Standardshader für eine Xbox 360, der das komponentenweise Zusammensetzen der Matrizen auf der GPU illustriert:

```
VertexShaderOutput VertexShaderFunction
(VertexShaderInput input) {
    VertexShaderOutput output;
```



Unsere Ressourcen sollen direkt in die .apk-Datei wandern (Bild 8)

```
float4 worldPosition = mul(input.Position, World);
float4 viewPosition = mul(worldPosition, View);
output.Position = mul(viewPosition, Projection);
return output;
}
```

Die programmierbare Grafikpipeline bietet zwei leere Slots an: Neben dem Vertex- ist auch ein Pixelshader erforderlich. Unser Beispiel legt ihn in eine Datei namens *pixelshader.txt*, die mit folgendem Code auszustatten ist:

```
precision mediump float;
varying vec4 v_Color;
void main() {
    gl_FragColor = v_Color;
}
```

varying leitet auch hier die Deklaration einer Variablen ein. *gl_FragColor* ist eine spezielle Variable, die den aus dem Shader zurückzugebenden Farbwert enthält. Die in ihr enthaltenen Informationen wandern im nächsten Schritt in den Framebuffer. Bei der Arbeit mit Fragment Shadern ist das Festlegen der Rechengenauigkeit nötig. Wir entscheiden uns hier für einen Kompromiss zwischen Rechenleistung und Präzision.

Im DirectX- und XNA-Bereich ist anstelle der unter OpenGL geläufigen Bezeichnung Fragment Shader der – nach Meinung des Autors besser zutreffende – Begriff Pixel Shader geläufig.

Zur Laufzeit müssen wir das Ressourcensystem anzapfen, um einen String mit dem Shadercode zu erhalten. Dieser wandert im nächsten Schritt in eine dedizierte OpenGL-Funktion, die den in die Runtime integrierten Compiler zur Assemblierung anweist. Da unser Programm nur zum Abernten von Shadern auf das Ressourcensystem zugreift, kombinieren wir die beiden Aufgaben in die Funktion *getShader()*:

```
int getShaderHandle(int whichOne, int whichType) {
    InputStream shaderSource= myApplication.
    getResources().openRawResource(whichOne);
    java.util.Scanner s = new java.util.
```



```
Scanner(shaderSource).useDelimiter("\\A");
String myString = s.hasNext() ? s.next() : "";
int shaderHandle = GLES20.glCreateShader(whichType);
```

RAW-Ressourcen werden von Android über die Methode *openRawResource* zur Verfügung gestellt. Für den Zugriff auf die zur Applikation gehörenden Ressourcen ist eine Referenz auf die *Application*-Klasse notwendig, die im Idealfall im Konstruktor bevölkert wird.

Das Einlesen des Strings erfolgt über einen kleinen, auf StackOverflow beschriebenen Trick: Das Zeichen `\\A` steht für den Anfang der Datei.

An dieser Stelle lernen wir das *GLES20*-Objekt erstmals kennen. Es handelt sich hier um eine statische Klasse, die für die Kommunikation zwischen Grafikpipeline und Applikation zuständig ist. Der Aufruf von *glCreateShader* erzeugt einen neuen Slot für einen Shader des angegebenen Typs, der durch die zurückgegebene Nummer beschrieben ist.

Das API ist nicht objektorientiert

OpenGL unterscheidet sich an dieser Stelle stark von Managed Frameworks: Das API ist nicht objektorientiert. Ressourcen werden über IDs angesprochen, die die jeweilige Generatorfunktion zurückgibt. Wundern Sie sich also nicht, wenn wir im Lauf der folgenden Schritte diverse andere Elemente als *ints* anlegen.

Der eigentliche Ladeprozess wird nur dann angeworfen, wenn die Erstellung des Shader Handles erfolgreich verlief (Listing 4).

Viele OpenGL-ES-Methoden kommunizieren mit dem Aufrufer durch ein vorher anzulegendes Array. Ist sein Rückgabewert *null*, so ist ein Fehler aufgetreten – weitere Informationen zu den Gründen lassen sich dann durch Aufrufen der jeweiligen Logfunktion ermitteln. Nach der erfolgreichen Abarbeitung der Lademethode liegt der Shader in Maschinencodeform vor. Zu seinem praktischen Einsatz fehlt noch

Listing 4: Shader Handle

```
if (shaderHandle != 0) {
    GLES20.glShaderSource(shaderHandle, myString);
    GLES20.glCompileShader(shaderHandle);
    final int[] compileStatus = new int[1];
    GLES20.glGetShaderiv(shaderHandle,
        GLES20.GL_COMPILE_STATUS, compileStatus, 0);

    if (compileStatus[0] == 0) {
        Log.e("NMG", "Kompilation gescheitert: " +
            GLES20.glGetShaderInfoLog(shaderHandle));
        GLES20.glDeleteShader(shaderHandle);
        shaderHandle = 0;
    }
}
return shaderHandle;
}
```

Listing 5: View-Matrix

```
final float eyeX = 0.0f;
final float eyeY = 0.0f;
final float eyeZ = -0.5f;
final float lookX = 0.0f;
final float lookY = 0.0f;
final float lookZ = -5.0f;
final float upX = 0.0f;
final float upY = 1.0f;
final float upZ = 0.0f;
Matrix.setLookAtM(myViewMatrix, 0, eyeX, eyeY, eyeZ,
    lookX, lookY, lookZ, upX, upY, upZ);
final int vSH = getShaderHandle(R.raw.vertexshader,
    GLES20.GL_VERTEX_SHADER);
final int pSH = getShaderHandle(R.raw.pixelshader,
    GLES20.GL_FRAGMENT_SHADER);
...
```

die Zuweisung von Variablen und die Assemblierung in ein Programm.

Alles zusammenfügen

Nach diesen Vorbereitungshandlungen können wir mit der Erzeugung der eigentlichen Renderingpipeline fortfahren. Die im Abschnitt zur Vertextransformation beschriebenen Matrizen entstehen in Form von sechzehnwertigen Float-Arrays, die als Member des Renderers anzulegen sind:

```
public class NMGRenderer implements
    GLSurfaceView.Renderer {
    ...
    private float[] myModelMatrix = new float[16];
    private float[] myViewMatrix = new float[16];
    private float[] myProjectionMatrix = new float[16];
    private float[] myMVPMatrix = new float[16];
```

onSurfaceCreated ist für die Belegung der diversen Matrizen zuständig. Als Erstes wollen wir eine statische Szene rendern, in der das Modell unverändert in der Mitte steht. Dies ist insofern vorteilhaft, als wir uns das permanente Anpassen der Weltmatrix des Würfelmodells ersparen:

```
@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config)
{
    GLES20.glClearColor(0.0f, 0.0f, 0.3f, 1.0f);
    GLES20.glEnable(GLES20.GL_CULL_FACE);
    GLES20.glEnable(GLES20.GL_DEPTH_TEST);
```

Als ersten Akt setzen wir einige Einstellungen für die Grafikpipeline. Die beiden *glEnable*-Aufrufe aktivieren die Funktionen *CULL_FACE* und *DEPTH_TEST*: Ersteres ist für die automatische Eliminierung von nicht sichtbaren Teilen des Modells zuständig, während *DEPTH_TEST* den für per-

spektivische Darstellung wichtigen Depth Sheck aktiviert. Er weist die GPU dazu an, dass der Tiefenpuffer Berücksichtigung finden muss. Im nächsten Schritt folgt die Deklaration der View-Matrix und die Bereitstellung des Shader-Maschinencodes (Listing 5).

Für die Erzeugung der Viewmatrix sind drei Parametersätze notwendig. Die drei mit *eye* bezeichneten Variablen legen dabei den Aufenthaltsort des virtuellen Auges fest. *Look* beschreibt die Blickrichtung, während der *Up* einen Einheitsvektor anlieert, der die Richtung von unten nach oben festlegt. Einheitsvektoren sind Vektoren mit einer Länge von 1. Sie beschreiben eine dimensionslose Richtung.

Das manuelle Bevölkern von Matrixfeldern ist unter OpenGL nur in den seltensten Fällen notwendig. Alle wichtigen Matrizen werden von einer Vielzahl von Generatorfunktionen unterstützt, die die eingehenden Parameter automatisch in Matrixwerte umwandeln.

Letzter Akt der Initialisierung

Als Maschinencode vorliegende Shader sind nur dann nützlich, wenn sie mit darzustellenden Daten verdrahtet sind. Zudem ist ein alleinstehender Vertex- oder Pixelshader im Moment nicht hilfreich. Reine Pixelshader kommen nur beim Post-Processing zum Einsatz, das wir erst in einem späteren Artikel beschreiben. Der letzte Akt der Initialisierung ist das Zusammenstellen der Maschinencodestücke zu einem betriebsbereiten Shader (Listing 6).

Unser Vertexshader exponiert Attributvariablen. Diese müssen bei der Erzeugung des fertigen Programms bei der Pipeline durch Aufruf von *glBindAttribLocation* angemeldet werden, sodass wir sie später mit den gewünschten Parametern bevölkern können. Wie bei der Kompilation der individuellen Codestücke prüfen wir auch hier, ob der Assemblierungsprozess erfolgreich durchgelaufen ist. Die Prüfung des Arrayelements gegen 0 ist ein fundamentales Pattern der Programmierung mit OpenGL ES:

```
final int[] linkStatus = new int[1];
GLES20.glGetProgramiv(myFinalProgramHandle,
```

```
GLES20.GL_LINK_STATUS, linkStatus, 0);
if (linkStatus[0] == 0) {
    Log.e("NMG", "Programmassemblierung fehlgeschlagen: "
        + GLES20.glGetProgramInfoLog(myFinalProgramHandle));
    GLES20.glDeleteProgram(myFinalProgramHandle);
}
}
```

Man muss nicht immer direkt in den Bildschirmspeicher rendern. Viele Grafikeffekte entstehen dadurch, dass man die 3D-Szene im ersten Schritt in einen Puffer rendert, der im nächsten Schritt durch einen weiteren Shader auf den Bildschirm wandert.

Die Projektionsmatrix ändert ihren Wert nur dann, wenn die Dimensionen des Bildschirms eine Veränderung erfahren. *OnSurfaceChanged* ist somit der ideale Aufenthaltsort für die Berechnung dieses Feldes:

```
@Override
public void onSurfaceChanged(GL10 gl, int width, int
height) {
    GLES20.glViewport(0, 0, width, height);
    final float ratio = (float) width / height;
    final float left = -ratio;
    final float right = ratio;
    final float bottom = -1.0f;
    final float top = 1.0f;
    final float near = 1.0f;
    final float far = 10.0f;
    Matrix.frustumM(myProjectionMatrix, 0, left,
        right, bottom, top, near, far);
}
```

glViewport informiert den Renderer über die Größe der zu bespielenden Fläche. *frustumM* erledigt die Zuweisung der Projektionsmatrix. Sie legt (unter anderem) den Beginn und das Ende des sichtbaren Bereichs fest.

Zu guter Letzt müssen wir bei jedem Durchlauf der Rendermethode die gesamte Zeichenfläche ausleeren. Dazu ist eine kleine Änderung in *onDrawFrame* erforderlich:

```
@Override
public void onDrawFrame(GL10 gl) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT |
        GLES20.GL_DEPTH_BUFFER_BIT);
}
```

glClear nimmt eine Gruppe von Bits entgegen, die die zu leeren Puffer beschreiben. Zum Zeitpunkt der Drucklegung kennt die Methode die drei Konstanten *GL_COLOR_BUFFER_BIT*, *GL_DEPTH_BUFFER_BIT* und *GL_STENCIL_BUFFER_BIT*, die bei Bedarf über den mathematischen Oder-Operator kombinierbar sind.

Rein theoretisch spricht nur wenig dagegen, immer alle drei Puffer zu leeren. Ist einer der Speicherbereiche nicht angelegt, so ignoriert die Methode das Bit bei spezifikationsgetreuer Implementierung. ►

Listing 6: Zusammenstellen der Maschinencodestücke

```
String[] attributes = {"a_Position", "a_Color"};
myFinalProgramHandle = GLES20.glCreateProgram();
if (myFinalProgramHandle != 0) {
    GLES20.glAttachShader
        (myFinalProgramHandle, vSH);
    GLES20.glAttachShader(myFinalProgramHandle,
        pSH);
    final int size = attributes.length;
    for (int i = 0; i < size; i++) {
        GLES20.glBindAttribLocation
            (myFinalProgramHandle, i, attributes[i]);
    }
    GLES20.glLinkProgram(myFinalProgramHandle);
```

Wer das Programm im vorliegenden Zustand ausführt, erfreut sich am in **Bild 9** gezeigten, leicht blauen Bildschirm. Die seltsame Farbe ist Schwarz beim Testing insofern überlegen, als sich Shaderfehler oft durch das Zurückgeben von komplett weißen oder komplett schwarzen Elementen artikulieren – bei einem einfarbigen Hintergrund dauert es länger, bis man die Anomalie findet.

Leider ist dies für uns nicht ganz ausreichend. Das Codebeispiel soll nämlich den Würfel auf den Bildschirm bringen. Dazu sind Änderungen an *onDrawFrame* notwendig (**Listing 7**).

Als erste Aufgabe müssen wir die Grafikpipeline darüber informieren, welches Programm für die Darstellung zuständig ist. Die Zuweisung von Werten an Variablen erfolgt über Attribut-Handles – die Zahlen lassen sich durch Aufrufe der *glGet*Location*-Methode übergeben, die den Namen der jeweiligen Variable in Stringform entgegennehmen.

Bereitstellung der Weltmatrix

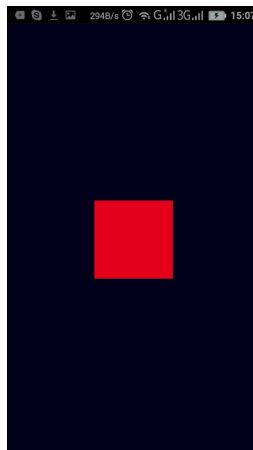
Als Nächstes folgt die Bereitstellung der Weltmatrix für das jeweilige Modell. Wir beschränken uns im Moment darauf, das Element in Richtung der z-Achse zu verschieben. *setIdentityM* bevölkert die angegebene Matrix mit einer speziellen Zahlenfolge, die das übergebene Objekt bei Multiplikation überhaupt nicht verändert:

```
Matrix.setIdentityM(myModelMatrix, 0);
Matrix.translateM(myModelMatrix, 0, 0.0f, 0.0f, -7.0f);
Matrix.rotateM(myModelMatrix, 0, 0, 1.0f, 0.0f, 0.0f);
```

Die Bereitstellung der Attributinformationen erfolgt durch Setzen von Verweisen auf die beiden Arrays. OpenGL ES wird die zum jeweiligen Punkt gehörenden Daten zur Lauf-



Die Renderingpipeline funktioniert perfekt – es sind keine seltsamen Objekte sichtbar (**Bild 9**)



Die Renderingpipeline bringt den Würfel vorschriftsgemäß auf den Bildschirm (**Bild 10**)

zeit automatisch bereitstellen:

```
myCubePositions.position(0);
GL20.glVertexAttribPointer(
    aPos, 3, GL20.GL_FLOAT,
    false, 0, myCubePositions);
GL20.
glEnableVertexAttribArray(
    aPos);
myCubeColors.position(0);
GL20.glVertexAttribPointer(
    aCol, 4, GL20.GL_FLOAT,
    false, 0, myCubeColors);
GL20.
glEnableVertexAttribArray(
    aCol);
```

Zu guter Letzt müssen wir die MVP-Matrix für den Shader bereitstellen. Die in Android implementierte Methode für Matrixmultiplikationen erleichtert uns dabei die Arbeit. Der Aufruf von *glDrawArrays* sorgt dann für den eigentlichen Zeichenbefehl – der dritte Parameter legt fest, wie viele Punkte abzuarbeiten sind:

```
Matrix.multiplyMM(myMVPMatrix, 0, myViewMatrix, 0,
    myModelMatrix, 0);
Matrix.multiplyMM(myMVPMatrix, 0, myProjectionMatrix,
    0, myMVPMatrix, 0);
GL20.glUniformMatrix4fv(aMVP, 1, false,
    myMVPMatrix, 0);
GL20.glDrawArrays(GL20.GL_TRIANGLES, 0, 36);
}
```

Damit sind wir an dieser Stelle vorerst fertig. Führen Sie das Programm auf einem Telefon Ihrer Wahl aus, um sich an der in **Bild 10** gezeigten Ausgabe zu erfreuen.

Schritt um Schritt

Das Anzeigen eines statischen Würfels mag aus mathematischer Sicht interessant sein – wirklich Spaß machen Spiele indes nur dann, wenn sich die am Bildschirm befindlichen Elemente herumbewegen. Als Entwickler steht man dabei vor einem Problem, weil sich Objekte der realen Welt so gut wie immer kontinuierlich bewegen.

Game Loops lösen dieses Problem durch Diskretisierung. Die Spielhandlung wird dabei in als Frames bezeichnete Elemente unterteilt, in denen alle Bewegungen als atomisch angenommen werden. Bei ausreichend klein gewählten Frameweiten und einer ausreichend hohen Framerate führt das Verfahren zum Eindruck eines kontinuierlich aktualisierten Bildes.

Nach dieser Beschreibung ist der Weg zu Bewegung am Bildschirm klar: Dank der geringen Anforderungen an die Grafikhardware brauchen wir uns im Moment keine Gedanken um Timing machen, weshalb wir die für das Rendering der Frames zuständige Methode auch zur Aktualisierung der

Listing 7: onDrawFrame

```
@Override
public void onDrawFrame(GL10 gl) {
    GL20.glClear(GL20.GL_COLOR_BUFFER_BIT |
        GL20.GL_DEPTH_BUFFER_BIT);
    GL20.glUseProgram(myFinalProgramHandle);
    int aMVP = GL20.glGetUniformLocation(
        myFinalProgramHandle, "u_MVPMatrix");
    int aPos = GL20.glGetAttribLocation(
        myFinalProgramHandle, "a_Position");
    int aCol = GL20.glGetAttribLocation(
        myFinalProgramHandle, "a_Color");
```

Listing 8: Rendering der Frames

```
@Override
public void onDrawFrame(GL10 gl) {
    rotationStep+=0.01;
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT |
        GLES20.GL_DEPTH_BUFFER_BIT);
    GLES20.glUseProgram(myFinalProgramHandle);
    int aMVP = GLES20.glGetUniformLocation
        (myFinalProgramHandle, "u_MVPMatrix");
    int aPos = GLES20.glGetAttribLocation
        (myFinalProgramHandle, "a_Position");
    int aCol = GLES20.glGetAttribLocation
        (myFinalProgramHandle, "a_Color");
    Matrix.setIdentityM(myModelMatrix, 0);
    Matrix.translateM(myModelMatrix, 0, 0.0f,
        0.0f, -7.0f);
    Matrix.rotateM(myModelMatrix, 0,
        rotationStep, 1.0f, 0.0f, 0.0f);
```

Position des Würfels benutzen (Listing 8).

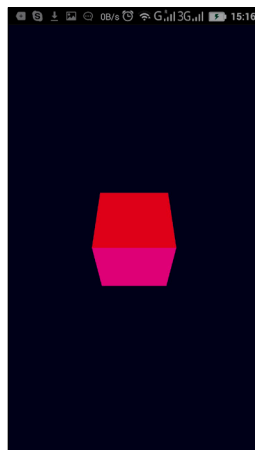
rotateM unterscheidet sich in mehrerer Hinsicht von den in anderen Programmiersprachen beziehungsweise Frameworks implementierten Drehmethoden. Erstens nimmt die Funktion Werte in Grad entgegen – das im Bereich der 3D-Programmierung weit verbreitete Radiantmaß wird nicht verstanden. Die zweite Änderung betrifft die Anforderung an das Vorhandensein einer Rotationsachse – die Methode ist unter Android folgendermaßen deklariert:

```
static void rotateM(float[] m,
    int mOffset, float a, float x, float y, float z)
```

x, y und z nehmen dabei die drei Komponenten eines Einheitsvektors entgegen, der die Drehachse beschreibt. Das Speichern des Wertes von *rotationStep* erfolgt hier in einer Membervariable von *NMGRenderer*. In komplexeren Applikationen beziehungsweise Spielen ist es sinnvoller, jedes einzelne Modell samt den dazugehörigen Parametern in einer Modellklasse zu speichern:

```
public class NMGRenderer implements
    GLSurfaceView.Renderer {
    ...
    float rotationStep=0;
```

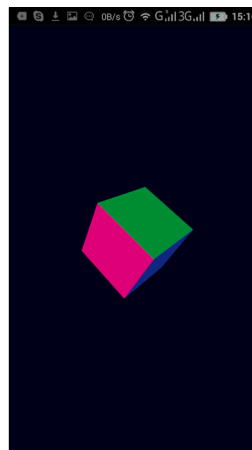
Wer das Programm im vorliegenden Zustand ausführt, sieht einen in eine Richtung rotierenden Quader – Bild 11 demon-



Der Quader dreht sich (Bild 11)

Listing 9: Rotationen

```
@Override
public void onDrawFrame(GL10 gl) {
    rotationStep+=1;
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT |
        GLES20.GL_DEPTH_BUFFER_BIT);
    GLES20.glUseProgram(myFinalProgramHandle);
    int aMVP = GLES20.glGetUniformLocation
        (myFinalProgramHandle, "u_MVPMatrix");
    int aPos = GLES20.glGetAttribLocation
        (myFinalProgramHandle, "a_Position");
    int aCol = GLES20.glGetAttribLocation
        (myFinalProgramHandle, "a_Color");
    Matrix.setIdentityM(myModelMatrix, 0);
    Matrix.translateM(myModelMatrix, 0, 0.0f,
        0.0f, -7.0f);
    Matrix.rotateM(myModelMatrix, 0,
        rotationStep, 1.0f, 1.0f, 0.0f);
```



Der Würfel taumelt nun durch den Raum (Bild 12)

striert das Verhalten. Rotationen lassen sich auch um mehrere Achsen gleichzeitig bewerkstelligen. Der einfachste Weg dazu ist die Nutzung folgendes Snippets, das den Würfel wie in Bild 12 gezeigt um zwei Achsen gleichzeitig bewegt (Listing 9).

Matrixmultiplikationen sind nicht kommutativ. Das bedeutet im Klartext, dass die Multiplikation von $a \times b$ und $b \times a$ zu unterschiedlichen Resultaten führt.

Im Allgemeinen ist es so, dass die Skalierung als Erstes erfolgt – rotiert wird erst danach, die Translation folgt als Letztes. Das ist wichtig, weil das Rotieren eines schon translatierten Objekts seine Position verändert. Der Mittelpunkt der Rotation ist immer 0|0|0.

Leider arbeiten Matrizen die in sie geschriebenen Orders in inverser Reihenfolge ab. Aus diesem Grund ist die Translation normalerweise im Code ganz oben, während Rotation und Skalierung darunter zu liegen kommen. ■



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

JOINS IN SQL-DATENBANKEN MEISTERN

Tabellen-Dschungel

Mit dem JOIN-Befehl lassen sich in SQL-Datenbanken Tabelleninhalte zusammenführen.

Eine der Schlüsselideen von relationalen Datenbanken ist die logische Trennung von Daten in einzelnen Tabellen. Um aus diesen normalisierten Informationen bestimmte Auswertungen zu gewinnen, ist es dann erforderlich, die getrennten Informationen wieder in geeigneter Weise zu verknüpfen. Und das geschieht je nach Aufgabenstellung auf verschiedene Weise (Bild 1).

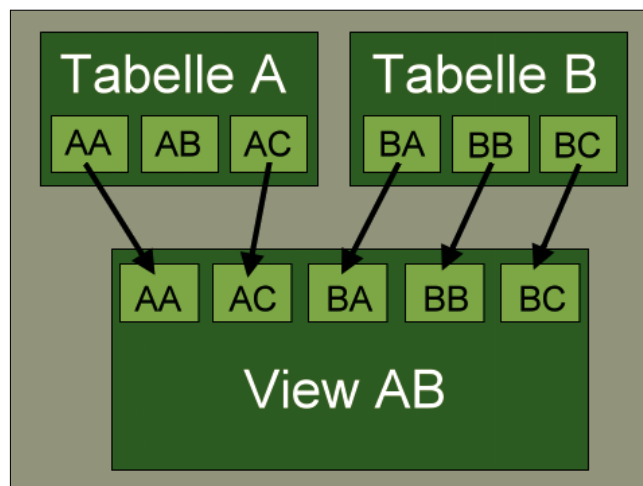
Um also beispielsweise innerhalb eines Shopsystems alle Bestellungen eines Tages zusammen mit den jeweils enthaltenen Artikeln geliefert zu bekommen, muss man die Tabellen mit den Bestellungen und die Tabelle, die die einzelnen Posten von Bestellungen enthält, durch ein Schlüsselfeld logisch verknüpfen (Bild 2).

Tabellen koppeln

Diese Koppelung können Sie an verschiedenen Stellen innerhalb Ihrer Architektur vornehmen. Ein Weg zum Ziel wäre, zuerst in Ihrer Anwendungssprache alle passenden Bestellungen von der Datenbank abzurufen, die Bestellungen dann einzeln durchzugehen und jeweils die zugehörigen Artikel zu ermitteln.

Dieses Verfahren wäre allerdings ziemlich ineffizient: Für jede Bestellung ist damit eine separate SQL-Abfrage notwendig, worunter die Ausführungszeit leidet. Besser ist es, die Verknüpfungsarbeit dort erledigen zu lassen, wo die Daten liegen: also auf dem SQL-Server. Denn der ist darin besonders gut.

Dazu gibt es bei SQL das Konzept der JOINS. Im Beispiel würden Sie damit sinngemäß sagen: »Gib mir alle Bestellungen und die zugehörigen Posten, die du anhand der Auftragsnummer zuordnest.« Damit haben Sie nur noch eine Abfra-



Bei relationalen Datenbanken werden die Daten in einzelnen Tabellen getrennt (Bild 1)

ge, die Sie in Ihrer Anwendung durchgehen müssen, und daher ein entsprechend schnelles Ergebnis.

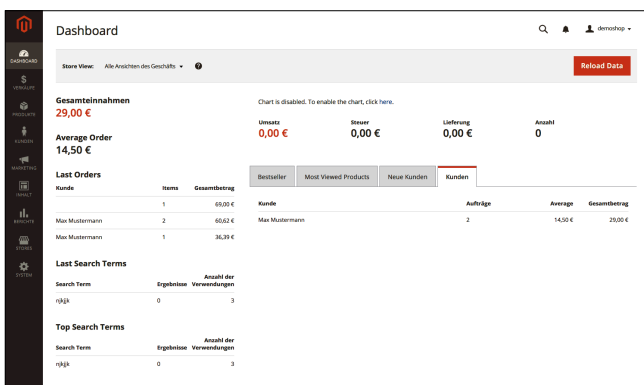
So arbeiten Sie mit dem JOIN-Kommando

Es gibt in SQL verschiedene Verknüpfungsmethoden, die man je nach der Problemstellung einsetzt. Um diese Varianten besser einordnen zu können, ist es hilfreich, wenn man sich gedanklich dem Konzept von JOINS über das sogenannte kartesische Produkt nähert, denn es ist quasi der größte denkbare JOIN.

Dabei werden alle Elemente der einen Tabelle mit den Elementen der anderen Tabelle verknüpft, man erhält also $n \times m$ Kombinationen. Alle JOINS kann man als Varianten dieses kartesischen Produkts verstehen, die nur jeweils anders gefiltert sind. Ein SQL-Server geht natürlich nicht so vor, sondern verwendet effizientere Methoden zur Ermittlung von JOINS. Aber als Denkmodell funktioniert das ganz gut.

Das kartesische Produkt ist übrigens kein rein theoretisches Konstrukt, sondern kann durchaus eine sinnvolle Antwort liefern: Denken Sie an eine Online-Partnerbörse. Dort ist es sicher interessant, jede gespeicherte Person mit jeder anderen Person testweise zu kombinieren, um über eine geeignete Berechnungsfunktion den Grad der Übereinstimmung zu ermitteln. Entsprechend sortiert erhalten Sie dann so eine Liste der am besten zueinander passenden Mitglieder.

Ein kartesisches Produkt haben Sie vielleicht auch schon aus Versehen selbst einmal produziert: Wenn Sie im `SELECT`-



Abfrage-Optionen in einem Shop-System, das auf einer relationalen Datenbank aufbaut (Bild 2)

Statement zwei Tabellen nennen und weder mit *JOIN* noch mit gleichwertigen *WHERE*-Einschränkungen arbeiten, liefert Ihnen die Datenbank genau dieses Produkt aus beiden Tabellen und braucht dafür – je nach Anzahl der enthaltenen Datensätze – oft sehr lange:

```
SELECT * FROM a,b
```

Um an Spieldaten für JOINS zu kommen, stellen wir uns eine Autovermietung vor, die in einer Tabelle *autos* ihre Fahrzeuge verwaltet und in der Tabelle *kunden* ihre Kunden. Leihst ein Kunde ein Auto aus, dann wird im Feld *kunden_id* der Tabelle *autos* dessen Kundennummer vermerkt.

Ein unvermietetes Auto hat hier eine 0 eingetragen. Natürlich ist dieses simple Modell kein taugliches Vorbild für eine echte Anwendung, aber es soll ja vor allem leicht nachvollziehbar sein, und man kann immerhin damit auch Kunden darstellen, die mehrere Autos gemietet haben.

Der Screenshot von phpMyAdmin zeigt Ihnen den Inhalt beider Tabellen (Bild 3). Einige Fahrzeuge sind noch unvermietet, einige sind vergeben und der Kunde Huber hat sogar zwei Autos gemietet.

Der Standard-Fall für JOINS

Die klassische Frage an die Datenbank in diesem Szenario ist eine Liste von Kunden zusammen mit ihren derzeit entliehenen Fahrzeugen. Dazu dient ein *INNER JOIN*. Weil dies die häufigste Variante eines *JOIN* ist, können Sie in allen bekannten Datenbank-Systemen das Schlüsselwort *INNER* auch weglassen und schreiben:

```
SELECT kunden.name,autos.typ
FROM kunden JOIN autos
ON kunden.id = autos.kunden_id
+-----+-----+
| name      | typ      |
+-----+-----+
| Huber, Franz | Audi A4   |
| Huber, Franz | Ford Transit |
| Wuttke, Wim  | VW Touran |
| Schmidt, Ben | Opel Adam |
+-----+-----+
```

Das Ergebnis der Abfrage sind alle Datensätze die sich aus dem kartesischen Produkt ergeben und bei denen der Eintrag im Feld *kunden.id* gleich dem Wert von *autos.kunden_id* ist. Herr Huber ist dort zweimal vertreten, weil er zwei Fahrzeuge ausgeliehen hat.

Ganz ohne *JOIN*-Syntax können Sie übrigens in diesem Fall das Ergebnis auch so erreichen:

```
SELECT kunden.name,autos.typ
FROM kunden, autos
WHERE kunden.id = autos.kunden_id
```

Allerdings ist die Verwendung des *JOIN*-Konstrukts immer empfehlenswert: Kommen noch weitere Tabellen und filtern-

de *WHERE*-Bedingungen hinzu, wird es schwierig zu erfassen, welcher Teil für die Verknüpfung und welcher für die zusätzlichen Filterungen zuständig ist.

Belastung für den Server

Eine Gefahr droht auch bei der Weiterverwendung solcher Abfragen an anderer Stelle per Copy and Paste. Denn wenn Sie diese Abfrage umformulieren, können Sie ganz schnell aus Versehen den Teil des *WHERE* löschen, der die Tabellen verknüpft. Das Ergebnis ist dann ein kartesisches Produkt, das den Server arg belastet.

In den Kreisdiagrammen entspricht diese Verknüpfung dem Bild, in dem nur die Schnittmenge ausgefüllt ist, denn es kommen ja nur die Datensätze zur Ansicht, bei denen ein Auto seinen Entleiher und ein Entleiher sein Fahrzeug hat.

LEFT JOIN-Variante

Nun sollen im Diagramm noch die Elemente aus dem linken Kreis dazukommen, die nicht in der Schnittmenge enthalten sind. Das sind also zusätzlich zum vorigen Ergebnis alle Kun-

SELECT * FROM autos				SELECT * FROM kunden	
id	typ	farbe	kunden_id	id	name
1	Audi A4	rot	2	1	Müller, Micha
2	VW Touran	grün	4	2	Huber, Franz
3	Dacia Logan	silber	0	3	Sinn, Lena
4	VW Golf	schwarz	0	4	Wuttke, Wim
7	Ford Transit	weiß	2	5	Schmidt, Ben
8	Opel Adam	rot	5		

Das sind die Testdaten unserer kleinen Autovermietung, die Kunden und Fahrzeuge zueinander bringt (Bild 3)

den, die derzeit kein Auto geliehen haben. Dies sieht in SQL folgendermaßen aus:

```
SELECT kunden.name,autos.typ
FROM kunden LEFT JOIN autos ON kunden.id =
autos.kunden_id
+-----+-----+
| name      | typ      |
+-----+-----+
| Müller, Micha | NULL     |
| Huber, Franz  | Audi A4   |
| Huber, Franz  | Ford Transit |
| Sinn, Lena    | NULL     |
| Wuttke, Wim   | VW Touran |
| Schmidt, Ben  | Opel Adam |
+-----+-----+
```

Alleine der Zusatz *LEFT* bewirkt also, dass der *JOIN* zusätzlich auch diejenigen Datensätze bringt, bei denen es keine verbundenen Datensätze in der verknüpften Tabelle gibt. ►

Hier hat die Spalte für den Autotyp den Wert *NULL*, weil eben keine Zuordnung möglich war.

Falls Sie Probleme haben, sich die Wirkung von links und rechts (was bald hinzukommt) zu merken: Mit der Seite ist immer die Stellung der maßgebenden Tabelle im SQL-Statement gemeint: Bei einem *LEFT JOIN* erscheinen gegenüber dem einfachen *JOIN* zusätzlich noch die Einträge der linken Tabelle, die keine korrespondierenden Verwandten in der rechten Tabelle haben.

Nur unverknüpfte Datensätze zeigen lassen

Im Beispiel der angeführten Autovermietung könnte auch die Frage auftreten, welche Kunden zurzeit gar kein Auto entliehen haben. Den Output dieser Abfrage könnte man etwa für eine Mailing-Aktion verwenden und den Kunden dabei Sonderrabatte anbieten.

Für diese Abfragevariante gibt es allerdings keine eigene *JOIN*-Syntax. Sie erhalten das gewünschte Ergebnis stattdessen durch einer weitere Filterung der letzten Variante, dem *LEFT JOIN*. Dazu geben Sie einfach vor, dass eines der Felder der verknüpften Auto-Tabelle unbelegt, also *NULL* sein soll:

```
SELECT kunden.name
FROM kunden
LEFT JOIN autos ON kunden.id = autos.kunden_id
WHERE autos.kunden_id IS NULL
```

name

Müller, Micha	
Sinn, Lena	

Das Feld *autos.kunden_id* ist dabei nur eine Option. Sie könnten theoretisch genauso den Typ des Autos verwenden, weil er genauso *NULL* ist. Die Verwendung des Schlüsselfeldes ist aber sicher die beste Wahl, um ihre Absichten bei der Abfrage klar zu machen, wenn jemand später Ihren Sourcecode verstehen soll.

RIGHT JOIN-Variante

Hatte der *LEFT JOIN* des Beispiels alle Kunden zusammen mit den eventuell gebuchten Autos gebracht, würde ein *RIGHT JOIN* alle Autos zusammen mit den Kunden bringen, oder eben *NULL*-Werte in den Kunden-Feldern, falls der Wagen frei ist:

```
SELECT kunden.name,autos.typ
FROM kunden RIGHT JOIN autos ON kunden.id =
autos.kunden_id
```

name	typ
Huber, Franz	Audi A4
Wuttke, Wim	VW Touran
NULL	Dacia Logan
NULL	VW Golf
Huber, Franz	Ford Transit

Syntax-Varianten

Der Teil der SQL-Abfrage, der die Verkettung der beteiligten Tabellen definiert, kann in vielen Fällen verkürzt dargestellt werden.

Sehen Sie sich etwa diesen *JOIN* an:

```
SELECT * FROM kunden k JOIN bestellungen b ON
k.kundennr=b.kundennr
```

Diese Abfrage können Sie auch in dieser Kurzform ausdrücken:

```
SELECT * FROM kunden k JOIN bestellungen
USING(kundennr)
```

Allerdings erfordert diese Variante, dass das für die Verknüpfung verwendete Feld in beiden Tabellen gleich benannt ist, hier also beispielsweise *kundennr*. Bei MySQL, Postgres und vielen anderen Systemen können Sie *USING* verwenden, Microsofts SQL-Server kennt das Schlüsselwort dagegen nicht.

Beim Beispiel des einfachen *JOIN* haben Sie gesehen, dass man es auch durch eine passende *WHERE*-Bedingung ersetzen kann. Zu den Anfangszeiten von SQL, als das Konzept des Schlüssel-

wortes *JOIN* noch nicht etabliert war, mussten sich die Datenbank-Hersteller etwas einfallen lassen, um auch die anderen Verknüpfungen ausdrückbar zu machen.

Dazu kam beispielsweise Oracle auf die Idee, die *WHERE*-Bedingung um ein in Klammern gesetztes Pluszeichen zu erweitern. Einen *LEFT JOIN* erreichte man dort dann durch eine Verknüpfung im *WHERE* wie:

```
SELECT * FROM kunden,autos WHERE kunde.id=
auto.kunden_id (+)
```

Das spezielle Symbol erweitert hier also die Antworten um die Fälle, in denen der Kunde kein zugeordnetes Fahrzeug hat. Bei Microsofts SQL-Server lautete das korrespondierende Konstrukt:

```
SELECT * FROM kunden,autos WHERE kunde.id *=
auto.kunden_id
```

Sollten Sie solche Abfragen in alten Code-Teilen sehen, ist es empfehlenswert, diese auf die modernere *JOIN*-Syntax umzustellen. Diese Varianten sind schwer zu verstehen und sind potenzielle Streichkandidaten für kommende Versionen der SQL-Server.

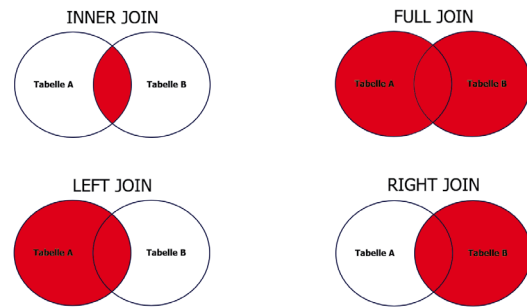
Visualisierung der JOIN-Typen

Mit diesen Diagrammen können Sie sich die Wirkungsweisen der verschiedenen JOINS besser einprägen.

Sie veranschaulichen, welche Datensätze jeweils in der Ergebnismenge enthalten sind. Das sind die farbig hervorgehobenen Flächen. Dabei ist der mittlere Teil, also die Schnittmenge, der Anteil der Daten, wo es korrespondierende Informationen in Tabelle A und B gibt.

Allerdings sollten Sie sich von einem Aspekt nicht aufs Glatteis führen lassen: Die Diagramme könnten suggerieren, dass es um Mengenoperationen geht, wozu JOINS aber nicht dienen. Um das nachzubilden, gibt es in SQL Operatoren wie UNION, INTERSECT oder MINUS.

JOIN-Varianten im Überblick



Die Wirkungsweisen der verschiedenen JOINS

```
| Schmidt, Ben | Opel Adam |
+-----+-----+
```

Allerdings ist diese Abfrage lediglich eine spiegelverkehrte Version der Query, bei der Sie als Haupt-Tabelle *autos* und als Nebentabelle *kunden* angeben:

```
SELECT kunden.name,autos.typ
FROM autos LEFT JOIN kunden ON kunden.id =
autos.kunden_id
```

Sie können also einen *RIGHT JOIN* immer durch einen *LEFT JOIN* ausdrücken, wenn Sie nur die Reihenfolge der Tabellen in der Query umdrehen. Angeblich sollen die meisten SQL-Server auch besser darin sein, *LEFT JOINS* zu optimieren, als *RIGHT JOINS* – wahrscheinlich, weil die linke Variante einfach viel häufiger benutzt wird.

Ohne passende Verknüpfungen

Natürlich hat auch der *RIGHT JOIN* eine Variante, die nur die Datensätze ohne passende Verknüpfungen in der linken Tabelle ausgibt:

```
SELECT kunden.name,autos.typ
FROM kunden RIGHT JOIN autos ON kunden.id =
autos.kunden_id
WHERE kunden.id IS NULL
+-----+-----+
| name | typ      |
+-----+-----+
| NULL | Dacia Logan |
| NULL | VW Golf    |
+-----+-----+
```

Auch dieser Anwendungsfall wäre durch die Erweiterung des *LEFT JOIN* um die *WHERE*-Bedingung ersetzbar, die nur Datensätze mit leerem Kunden-Teil durchlässt. Allerdings ist dieser Query, die die Frage nach den derzeit unbelegten Fahrzeugen beantwortet, viel einfacher ganz ohne *JOIN* lös-

bar. Denn die Tabelle *autos* hat mit dem Verknüpfungsfeld *kunden_id* ja bereits alles, was Sie dafür brauchen:

```
SELECT kunden.name,autos.typ
FROM autos where kunden_id=0
```

Sie sollten also trotz der Begeisterung über die mächtigen Möglichkeiten mit *JOIN* immer bedenken, ob für die konkrete Fragestellung der Einsatz überhaupt notwendig ist. Denn natürlich kostet die Verwendung einer Verknüpfung mehr CPU-Zyklen als die Abfrage einer einzigen Tabelle.

Alles in einen Topf werfen: der FULL OUTER JOIN

Schließlich gibt es noch den *FULL OUTER JOIN*. In unserem Beispiel würde der in einer Abfrage alle belegten Autos, alle autolosen Kunden und alle kundenlosen Autos bringen. Der Anwendungsfall ist so selten, dass er in MySQL nicht direkt enthalten ist, sondern durch die Verkettung zweier Abfragen per *UNION* nachgebildet werden muss:

```
SELECT typ,name FROM autos LEFT JOIN kunden
ON autos.kunden_id=kunden.id UNION
SELECT typ,name FROM kunden LEFT JOIN autos
ON autos.kunden_id=kunden.id
+-----+-----+
| typ      | name          |
+-----+-----+
| Audi A4   | Huber, Franz  |
| VW Touran | Wuttke, Wim   |
| Dacia Logan | NULL         |
| VW Golf   | NULL          |
| Ford Transit | Huber, Franz |
| Opel Adam | Schmidt, Ben  |
| NULL      | Müller, Micha |
| NULL      | Sinn, Lena    |
+-----+-----+
```

Das *UNION* kümmert sich dann darum, dass die doppelt vorkommenden Datensätze für die Fälle *Kunde hat ein Auto* ►

aus den beiden Teilabfragen im Ergebnis letztlich nur einmal erscheinen.

Die Verknüpfungen weiter ausbauen

Die JOINS, die Sie bislang gesehen haben, zeigten immer nur die minimal notwendige Syntax. Aber Sie können dabei natürlich zusätzlich alle Mittel zum Filtern oder Gruppieren verwenden, wie mit anderen Abfragen auch. Dabei wirken diese Zusätze immer erst nach dem Erfüllen der vorgegebenen Verknüpfungen: Der Server führt immer zuerst den *JOIN* aus und erst auf dieses Zwischenergebnis wirken *WHERE*-Filter oder *GROUP BY*-Operatoren.

Ein Beispiel für einen solchen Zusatz: Vielleicht möchte die Autovermietung herausfinden, wer alles ein rotes Fahrzeug gemietet hat. Die anderen Kunden interessieren also nicht. Das würde die folgende Abfrage herausfinden:

```
SELECT * FROM kunden JOIN autos ON kunden.id = autos.kunden_id WHERE autos.farbe='rot'
```

Genauso einfach können Sie eine Gruppierung einfügen. Das könnte beispielsweise dann interessant sein, wenn Sie nicht die Details zu den Fahrzeugen wissen möchten, die Ihre Kunden gemietet haben, sondern nur die reine Anzahl der Fahrzeuge:

```
SELECT kunden.name, COUNT(*) as anzahl
FROM kunden JOIN autos ON kunden.id = autos.kunden_id
group by kunden.id
```

name	anzahl
Huber, Franz	2
Wuttke, Wim	1
Schmidt, Ben	1

JOINS kennen viele Namen

Gerade bei JOINS haben sich im Lauf der Zeit viele Bezeichnungen eingebürgert, die zum Teil Synonyme bilden und daher das Verständnis unnötig erschweren.

Dass der *INNER JOIN* wegen seiner Verbreitung in Diskussionen, aber eben auch in der SQL-Syntax selbst einfach als *JOIN* bezeichnet wird, haben Sie ja bereits im Beitrag gesehen. Er wird aber auch noch als *EQUIVALENT JOIN* oder kurz *EQUI JOIN* bezeichnet.

Der *LEFT JOIN*, wo also auch Datensätze vorkommen, in denen die verknüpfte Tabelle keine Entsprechung hat, wird oft auch als *LEFT OUTER JOIN* bezeichnet. Das ist anschaulich, da im Schnittmengendiagramm ja gegenüber dem einfachen *JOIN* auf der linken Seite diejenigen Elemente hinzukommen, die sich nicht »innen«, also in der Schnittmenge befinden.

Ein *CROSS JOIN* entspricht dem Ergebnis des kartesischen Produkts: Das Ergebnis ist eine Kombination aller Elemente der einen und aller Elemente der anderen Tabelle. Das Ergebnis hat also so viele Elemente wie das Produkt der Anzahl von Elementen in allen beteiligten Tabellen.

Ein *SELF JOIN*, auch *AUTO JOIN* genannt, ist kein separater *JOIN*-Typ, sondern lediglich ein *JOIN* aus einer Tabelle mit sich selbst. Das klingt nicht so nützlich, es gibt aber durchaus praktische Anwendungsfälle. Stellen Sie sich etwa eine Personaltafel vor, die neben der Personalnummer der jeweiligen Person auch die seines Chefs enthält. Dann können Sie eine Personalliste erzeugen, in denen der Name des Vorgesetzten erscheint, also etwa:

```
SELECT p1.nr, p1.name, p2.name AS vorgesetzter
FROM perso p1 LEFT JOIN perso p2
WHERE p1.chef_nr=p2.nr
```

Wie Sie sehen, müssen Sie hier zwingend mit Aliassen für die Tabellen arbeiten (hier *p1* und *p2*), damit der Server weiß, welche der beiden gleichnamigen Tabellen Sie in den Bezügen meinen.

Allerdings zeigt diese Abfrage nur Kunden, die mindestens ein Auto gemietet haben, was nicht am *JOIN* liegt, sondern an der Wirkungsweise von Gruppierungen. Wollten Sie hier auch Kunden ohne Auto aufgeführt haben, müssten Sie mit einem Subselect statt einem *JOIN* arbeiten:

```
SELECT kunden.name, ( SELECT COUNT(*) FROM autos WHERE kunden_id=kunden.id) FROM kunden
```

Aus Gründen der Übersichtlichkeit haben wir bislang nur immer zwei Tabellen verknüpft. JOINS sind aber nicht darauf beschränkt. Sie können also einfach eine oder mehrere weitere Verkettungen in Ihre Abfragen einbauen.

Denken Sie an einen Internet-Shop. Dort soll eine Liste von Kunden ausgegeben werden, die auch deren bestellte Produkte enthält. Dazu dient dann eine Abfrage, die drei Tabellen verknüpft:

```
SELECT k.name, p.artikel FROM kunden k
JOIN bestellungen b ON k.kdnr=b.kdnr
JOIN bestellposten p ON b.auftragnr=p.auftragnr
```

Theoretisch könnten Sie auf diese Weise beliebig viele *JOIN*-Verknüpfungen nutzen. Mit jeder weiteren Ebene wird aber der Datenbank-Server ein Stück mehr belastet, sodass Sie damit trotzdem sparsam umgehen sollten. ■



Markus Schraudolph

ist Journalist und Programmierer. Er schreibt seit vielen Jahren Bücher und Artikel für Fachzeitschriften. Seine Schwerpunktgebiete als Programmierer sind die Webprogrammierung und Datenbanken.

Downloaden, aufschlauen!



SERVER VIRTUALISIEREN

Virtualisierung

Nicht nur auf Desktop-Rechnern ist Virtualisierung ein großes Thema.

Dank Virtualisierung lassen sich in die Jahre gekommene PCs leicht ausrangieren. Die installierten Windows- und Linux-Betriebssysteme können mit einem Desktop-Tool wie dem kostenlosen VMware vCenter Converter (www.vmware.com/de/products/converter) unter Windows mit einem vergleichsweise geringen Aufwand in eine virtuelle Umgebung überführt werden. Dazu wird der bisherige Server als virtuelles Image gesichert und auf dem Zielrechner nach Anpassung der Konfiguration innerhalb des Haupt-Betriebssystems in einer Virtualisierungs-Software wie VMware Workstation Pro (www.vmware.com/de/products/workstation) gestartet.

Vorteile der Server-Virtualisierung

Virtualisierung vereinfacht aber nicht nur das Migrieren älterer Server auf neue Hardware-Plattformen, sondern hilft auch, vorhandene Server transparent zusammenzufassen und dadurch Kosten zu sparen. Neue Server-Hardware ist oft so leistungsfähig, dass die Hardware-Ressourcen ausreichen, um auf einem Gerät mehrere ältere Server zu vereinen und parallel auszuführen. So kann ein gut ausgestatteter Server-Rechner oft drei und mehr ältere Server ersetzen – entsprechend groß ist das Einsparpotenzial bei Energiebedarf, Wartung und Service im Vergleich zu echten Servern.

Durch Server-Virtualisierung bündelt der Administrator außerdem heterogene Netzwerk-Ressourcen, indem er beispielsweise unterschiedliche Netzstränge über virtuelle Netzwerkkarten auf einem Server miteinander verknüpft und so einen Datenaustausch ermöglicht. Anstelle echter

Router und Switches genügt es, die virtuellen Netzadapter entsprechend zu konfigurieren.

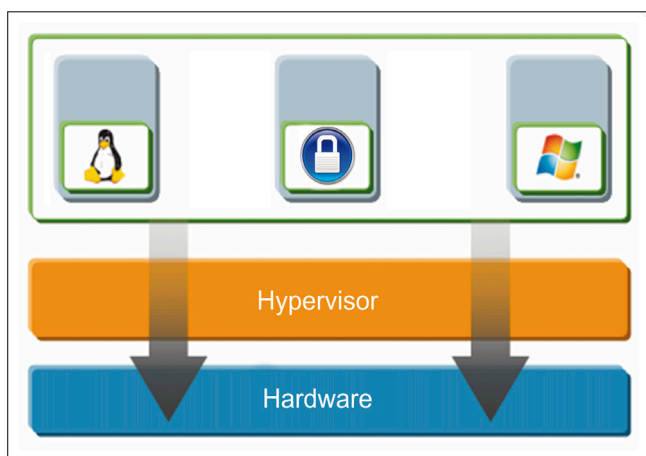
Die einfache Skalierbarkeit von virtualisierten Servern an die jeweiligen Bedürfnisse ist ein weiterer Vorteil der Server-Virtualisierung. So kann man einen als virtuelle Maschine vorliegenden Server bei Bedarf auf schnellere und besser ausgestattete Hardware verlagern. Auch zusätzlicher Speicherplatz lässt sich auf einfache Weise bereitstellen. Im Schadensfall reduziert ein virtueller Server den Aufwand für die Wiederherstellung im Vergleich zu einem echten Server deutlich. Es genügt, den im Rahmen eines Backups gesicherten virtuellen Server auf ein anderes System zurückzuschreiben und zu starten.

Virtualisierungs-Plattformen für Server

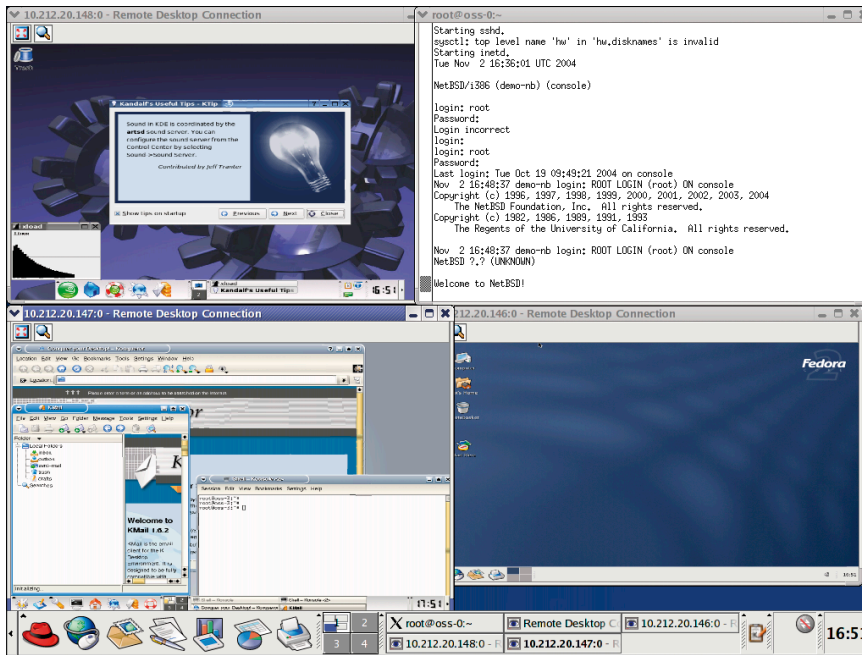
Microsoft Hyper-V, Citrix Xen Server und VMware ESXi sind allesamt kostenlose Virtualisierungsplattformen zur Server-Virtualisierung und sollen den Einstieg erleichtern. Die Gratis-Varianten unterscheiden sich im Vergleich zu den Vollversionen beispielsweise durch limitierte Verwaltungsfunktionen, Begrenzung der unterstützten Prozessoren oder durch reduzierten Support für Speichersysteme und Hochverfügbarkeit. Das Geld wird über die Managementwerkzeuge verdient, die das Vollprodukt ausmachen und für größere Umgebungen zwingend benötigt werden. Kern aller Systeme ist der Hypervisor. Er fungiert als Motor und übernimmt die Steuerung der Ressourcenzuteilung an die einzelnen virtuellen Maschinen. Er setzt entweder direkt auf die Hardware oder auf ein laufendes Betriebssystem auf.

Microsoft Hyper-V Server: Die proprietäre Lösung stellt ihren Hypervisor kostenlos zur Verfügung. Die Management-Suite Microsoft System Center für die Administration größerer Umgebungen von Hochverfügbarkeits- über Storage- und Netzwerk-Virtualisierung bis hin zu Disaster Recovery bleibt den zahlenden Kunden vorbehalten. Somit eignet sich der Gratis-Virtualisierer als Lösung für Standalone-Server mit den Anwendungsschwerpunkten Testen, Betrieb von Entwicklungs- und Testumgebungen sowie für den Betrieb weniger kritischer Systeme.

Die Hypervisor-Technik von Hyper-V Server ist nicht nur identisch mit der in Windows Server 2012 R2, sondern enthält dabei auch nahezu sämtliche Funktionen des Vollprodukts (Bild 1). Damit stehen so mächtige Funktionen wie Live-Migration, High Availability, VM-Replikation, der Hyper-V-Switch für Netzwerkvirtualisierung und Mandantenfähigkeit der virtualisierten Umgebung zur Verfügung – um nur die wichtigsten Funktionen zu nennen. Auch das VHDX-Dateiformat ist an Bord, das große virtuelle Festplatten sowie An-



Ein Hypervisor erlaubt es, eine virtuelle Umgebung mit Hardware-Ressourcen zu definieren, die unabhängig von der tatsächlich vorhandenen Hardware als Basis für die Installation von Gast-Betriebssystemen dient (Bild 1)



Xen ist ein Hypervisor, der direkt auf einer Hardware läuft. Xen kann mehrere Betriebssysteme in virtuellen Maschinen starten. Für diese sind weder der Hypervisor noch andere Domänen sichtbar. Vom Prinzip her ist das Vorgehen vergleichbar mit virtuellem Speicher und Prozessen (Bild 2)

derungen von deren Kapazitäten im Live-Betrieb ermöglicht. Mit der aktuellen Version hat sich zudem der Support für Linux-Gäste erheblich verbessert. So ist Hyper-V nun auch gut dazu geeignet, die gängigen Enterprise-Linux-Systeme zu betreiben. Der Host unterstützt bis zu 320 CPUs und 4 TByte RAM. Einzelnen virtuelle Maschinen können bis zu 1 TByte Arbeitsspeicher und 64 CPUs zugewiesen werden.

VMware ESXi: Seit Anbeginn stellt VMware zusätzlich zu seinem Top-Produkt Vsphere den eigentlichen Hypervisor ESXi kostenfrei bereit. Er besitzt die gesteigerten technischen Fähigkeiten und erweiterten Skalierungskapazitäten seines großen Vsphere-Bruders, ist jedoch durch eine Vielzahl an Limitierungen charakterisiert. So entfällt zwar inzwischen die Obergrenze von 32 GByte RAM je Host, jedoch gilt nach wie vor die Beschränkung des Servers auf zwei CPUs und maximal acht vCPUs in einer VM. Zudem sind die meisten APIs eingeschränkt, sodass der Einsatz vieler Management-Tools verhindert wird. Zudem ist das fehlende Management ein großes Manko des kostenfreien Hypervisors. Setup und Basiskonfiguration gelingen noch sehr einfach über eine Textkonsole. Für alles Weitere wird der Vsphere-Client benötigt, der als Werkzeug für das Remote Management dient. Von einem Windows-Rechner aus lässt sich damit der ESXi Host mit virtuellen Maschinen bestücken und steuern.

Citrix Xen Server: Die Open-Source-Software (www.citrix.de/products/xenserver) wurde ursprünglich die Firma Xen als kommerzielles Paket mit Management-Tools veröffentlicht (Bild 2). Dieses wurde von Citrix übernommen und in verschiedenen Editionen inklusive einer kostenfreien Basisvariante vermarktet. Mit der Version 6.2 hat Citrix den kompletten Software-Stack inklusive der Management-Werkzeuge

als Open-Source-Paket an die Community zurückgegeben. Damit ist Xen Server die kommerziell unterstützte Version des Open-Source-Hypervisors Xen (www.xenproject.org).

Die Liste der Funktionen ist durchaus beeindruckend. So kann das Xen Center als zentrale Management-Oberfläche für Windows dienen und es gibt Live Migration, Site Recovery für Datenrettungsszenarien und Backups auf Basis von VM-Snapshots. Citrix positioniert Xen Server zudem als Cloud-optimierte Virtualisierungstechnologie durch seine direkte Integrationsmöglichkeiten mit Plattformen wie Apache Cloudstack und Openstack. Hierzu gehören auch Skalierungskapazitäten und Netzwerktechniken für den Aufbau mandantenfähiger Virtualisierungsumgebungen, die durch VLANs und IP-Tables entsprechend isoliert und abgesichert werden können. Die kommerzielle Version des Citrix Xen Servers bietet im Gegensatz zum kostenfreien Paket neben dem Hersteller-Support auch die automatische Installation

von Sicherheits-Fixes, Updates und Maintenance-Releases.

Kernel Based Virtual Machine (KVM): Der Open-Source-Hypervisor erfreut sich steigender Beliebtheit und ist direkt in den Linux-Kernel integriert. Ein separater Hypervisor und eine gesonderte virtuelle Partition für den Betrieb virtueller Maschinen sind nicht erforderlich. Der Linux-Kernel selbst stellt unter anderem Scheduling, Speicherverwaltung und Treiber zur Verfügung. Bei der Installation sind im Wesentlichen die Kernel-Module zum bestehenden System sowie Qemu und Management-Tools einzurichten.

Auch bei der Nutzung finden sich Linux-Admins sofort zu recht: Jeder Gast beziehungsweise jede virtuelle CPU verhalten sich wie ganz gewöhnliche Linux-Prozesse und können so beispielsweise auch über normale Kommandos kontrolliert und gesteuert werden. Dies gilt auch für die Gerätelandschaft, speziell für Speichergeräte – da hier die normalen Linux-Treiber genutzt werden, ist eine Umgewöhnung nicht nötig. Zugleich wird klar, dass für den Aufbau und den Einsatz von KVM-basierten Umgebungen tiefer gehendes Linux-Know-how unabdingbar ist. ■



Christoph Hoffmann

ist seit über 20 Jahren als Autor für verschiedene Verlage tätig. Mit seiner Firma stroemung GmbH realisiert er unter anderem Sonderheft-Projekte und ist im Bereich Content-Marketing aktiv.

www.stroemung.de

CLEARSCRIPT UND C#

Das Beste beider Welten

Auch Entwickler und Admins wissen die Vorteile grafischer Oberflächen zu schätzen.

Zeitgemäße Betriebssysteme im Desktopbereich haben eine grafische Bedienoberfläche – analog gilt das auch für die meisten Applikationen. Otto Normalbenutzer erwartet ein solches interaktives GUI (Graphical User Interface) von lokal installierten Anwendungen und Webseiten.

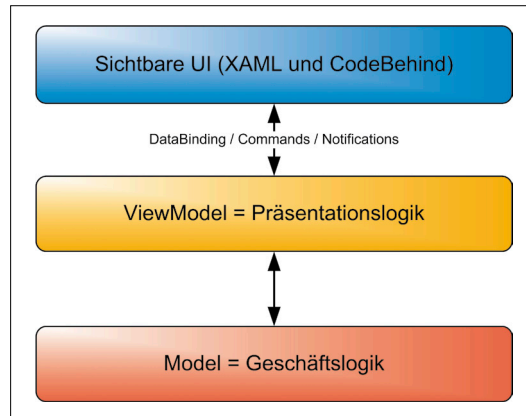
Im Serverbereich und auch bei zahlreichen Entwicklungswerkzeugen verschiebt sich das Bild. Der professionelle und semiprofessionelle Admin, Entwickler oder Softwaretester weiß zwar durchaus auch ein gutes GUI zu schätzen, dennoch liegt sein primäres Interesse in einer möglichst effizienten Erledigung der jeweiligen Arbeitsaufgabe mit Hilfe leistungsfähiger Tools. Die Oberfläche steht dabei erst an zweiter Stelle, da sich viele Dinge auf der Kommandozeile eben besser und vor allem schneller erledigen lassen.

Komplexe Aufrufsyntax

Zwar kann auch die korrekte Zusammenstellung einer komplexen Aufrufsyntax eine nichttriviale Angelegenheit sein. Dies ist aber erstens für viele Tasks nicht der Regelfall, zweitens wird man sich in diesem Fall dann schnell ein entsprechendes Shellskript oder eine Batchdatei schreiben, um solcherlei komplexe oder häufig wiederkehrende Arbeiten reproduzierbar auf Knopfdruck wiederholen zu können beziehungsweise sie als zeitgesteuerte Aktivitäten via Taskplaner oder Cron-Jobs zu automatisieren.

Und das ist genau der Punkt, an dem es mit traditionellen GUI-only-Anwendungen schwierig wird. Um eine grafische Oberfläche automatisiert ansteuern oder testen zu können, sind spezielle Tools erforderlich, deren tatsächliche Leistungsfähigkeit keineswegs immer mit den vom Anbieter geforderten Lizenzpreisen korreliert.

Das liegt allerdings nicht in allen Fällen an den eingesetzten Werkzeugen, sondern häufig eher an der Struktur komplexerer GUI-



MVVM-Pattern: Das Grundprinzip des MVVM-Patterns (Bild 1)

Komponenten, die sich dadurch nicht selten dem Zugriff von außen widersetzen.

Wird eine solche grafische Bedienoberfläche neu entworfen oder nachträglich überarbeitet, müssen diese Änderungen arbeitsaufwendig nachgepflegt werden – ein Faktum, das gerade das automatisierte Testen im Entstehen begriffener Oberflächen nicht selten einfach aufgrund des damit verbundenen Aufwands und Ressourcenbedarfes schon im Ansatz scheitern lässt.

Als Entwicklungsarchitektur für moderne Oberflächen hat sich da-

her in den letzten Jahren neben anderen die sogenannte Model-View-Viewmodel-Architektur, kurz als MVVM bezeichnet, einen Namen gemacht (Bild 1).

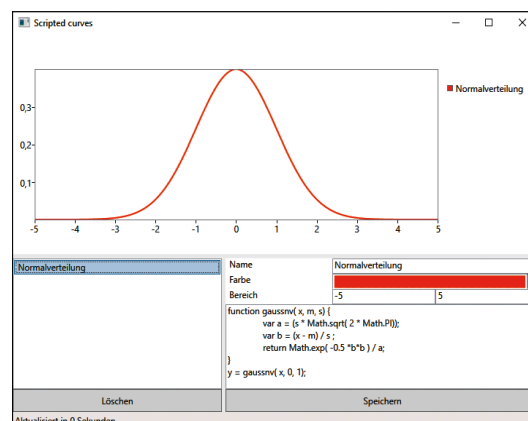
Bessere Testbarkeit der Oberfläche

Erklärtes Ziel dieses Architekturstils ist die möglichst einfache Testbarkeit der Bedienoberfläche. Erreicht wird dies, indem man einerseits die programmseitigen Aspekte der Oberfläche so dünn wie möglich hält, und andererseits, indem man diese als einzige Abhängigkeiten über definierte Schnittstellen und Muster wie Data Binding und Notifications an die UI-Logik anknüpft.

Durch diese Auftrennung lässt sich die Bedienoberfläche relativ einfach in ein geeignetes Testgerüst packen und flexibel aus Testskripts heraus ansteuern,

die sich dann natürlich jederzeit auch automatisiert im Rahmen eines CI-Laufs ausführen lassen

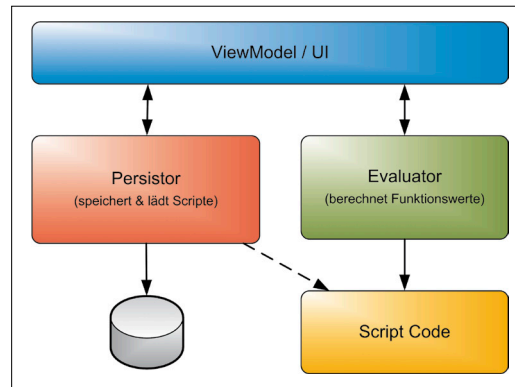
Geht man noch einen Schritt weiter, erscheint es naheliegend, die Möglichkeit zum Scripting gleich direkt in den Anwendungscode aufzunehmen. Indem man kundigen Anwendern ein solches zusätzliches Werkzeug an die Hand gibt, können diese die Funktionalität der Anwendung in gewissen Grenzen selbst erweitern, ohne dass der Entwickler auch nur eine Quelltextdatei neu kompilieren müsste.



Live: So präsentiert sich der Funktionsplotter in Aktion (Bild 2)

Viele Dinge direkt in der grafischen Oberfläche lassen sich so flexibler und effizienter erledigen. Nicht zuletzt ist man damit auch in der Lage, eine dedizierte, saubere, vollständige und konsistente Schnittstelle für Tests und UI-Automation anzubieten.

Die Anwendungsmöglichkeiten sind in der Tat vielgestaltig: Von der bereits angesprochenen automatisierten Ausführung und dem Testen der Anwendung über Formeln und Berechnung, die Validierung von Formularen oder die Auswertung eintreffender Daten nach fast beliebig komplexen Kriterien lassen sich schnell Einsatzfälle und Features finden, deren Kundennutzen mittels Scripting-Möglichkeit deutlich erhöht würde.



Funktionsplotter: Schematische Darstellung der Architektur des Funktionsplotters (Bild 3)

wert herausgehoben. Klare Vorteile von ClearScript sind in erster Linie die Nutzung etablierter Standard-Skript-Engines sowie der vergleichsweise geringe Aufwand, um eigene .NET-basierte Applikationen mit Skriptunterstützung zu versehen.

Zur Installation des Pakets gilt es zunächst auf der Downloadseite den entsprechenden Link anzuklicken und die ZIP-Datei in ein geeignetes Verzeichnis zu entpacken. Will man ClearScript zunächst nur testen oder benötigt man die V8-Engine nicht unbedingt, lädt man

einfach die ClearScript.NoV8 ins eigene Visual Studio und baut dieses Projekt. Die so entstandenen Assemblies werden ganz normal in das jeweilige Zielprojekt eingebunden.

ClearScript

Obwohl sich die folgenden Beispiele und Details primär auf C# und auf das auf Codeplex gehostete ClearScript beziehen, lassen sich die meisten Prinzipien grundsätzlich auch auf andere Produkte übertragen. Dem geeigneten C#-Entwickler bietet ClearScript ein ausgereiftes und aktiv gepflegtes Ökosystem aus mehreren Skript-Engines (<https://clearscript.codeplex.com>).

Da die Firma Microsoft höchstselbst hinter dem Projekt steht, werden natürlich mit JScript und VBScript zwei Microsoft-Engines unterstützt. Aber auch die V8 JavaScript Engine von Google wird unterstützt und sogar als empfehlens-

Integrierte V8-Unterstützung

Für den Build mit integrierter V8-Unterstützung benötigt man etwas mehr Vorarbeit und ein paar zusätzliche Voraussetzungen wie Python 2.7, Git und eine installierte VisualStudio-C++-Umgebung. Die genauen Details lassen sich zusammen mit den nötigen Schritten der Datei *ReadMe.txt* entnehmen. Benötigt man den Zugriff auf die Quelltexte nicht, geht es auch einfacher.

In gewohnter VisualStudio-Manier greift man zur NuGet-Konsole und fügt das ClearScript.Installer-Paket von Eric Swann zum Projekt hinzu. Dieses Paket beinhaltet auch gleich die V8-DLLs. ►

Immediately Invoked Function Expression (IIFE)

Hinter dieser etwas unhandlichen Beschreibung verbirgt sich ein inzwischen gängiges JavaScript-Pattern, dessen primärer Zweck die Kapselung ist.

Traditionell ist JavaScript eher nicht für strikte Kapselung bekannt. Die Namensgebung, die sich inzwischen durchgesetzt hat, bezieht sich auf die syntaktische Eigenheit des Konstrukts, das eben einen Funktionsausdruck ergibt, der unmittelbar ausgeführt wird:

```
(function() {
    /* beliebiger Code */
})();
```

Seine gesamte Power spielt dieses Konstrukt in der Erweiterung als sogenanntes Modulpattern aus. Hier wird die obige *function()* so erweitert, dass sie ein Objekt zurückgibt, dessen Elemente wieder Funktionen sind, oder besser gesagt Methoden:

```
var Foobarizer = (function() {
    var some_data = "I'm private";
```

```
var PrivateFunc(arg) {
    some_data += arg;
};

var DoStuff = function(arg) {
    PrivateFunc(arg);
};

// return instance with public methods
return {
    DoStuff: DoStuff
};

})();

// use the instance
Foobarizer.DoStuff(42);
```

Wesentlicher Vorteil dieser Konstruktion ist eben genau die Kapselung. Alles, was innerhalb des inneren Scopes definiert wird, ist auch nur genau dort sichtbar.

Um einen schnellen Überblick zu bekommen, empfiehlt sich zumindest das Überfliegen des im Ordner *ClearScript\doc* zu findende *FAQtorial.pdf*, in dem die wesentlichen Features kurz anhand einer fiktiven Frage-Antwort-Runde demonstriert werden.

Auch die im selben Verzeichnis befindliche Referenz ist gelegentlich hilfreich. Neben mehr oder weniger ausführlichen Auflistungen von Properties und Methoden findet man darin immer mal wieder den ein oder anderen nützlichen Code-Schnipsel.

Als ersten Einsatzfall wollen wir uns einen einfachen Funktionsplotter bauen (Bild 2). Die Anforderungen sind schnell skizziert: Die Hostanwendung ist natürlich in C# geschrieben und bietet ein simples UI, in dem der Anwender beliebig viele mathematische Funktionen der Form $y = f(x)$ anlegen und mit verschiedenfarbigen Linien als Diagramm darstellen kann.

Für Letzteres kann eine beliebige Chartkomponente zum Einsatz kommen, beispielsweise das Chart-Control aus den Microsoft DataVisualization Controls.

Listing 1: Ansteuerung der Chartfunktionen

```
public static string UpdateChart(
IEnumerable<ChartFunction> functions, Chart chart) {
    var timer = new Stopwatch();
    timer.Start();
    chart.Series.Clear();

    var errors = new StringBuilder();
    var sFunc = String.Empty;
    using (var engine = new JScriptEngine()) {
        try {
            const int STEPS = 200; // plus 1
            const string TEMPLATE = "(function(x) {\n"
                + " var y = 0.0;\n"
                + " {0};\n"
                + " return y;\n"
                + "})({1});";

            foreach (var entry in functions)
            {
                sFunc = entry.Name;
                var range = 1.0 * (entry.RangeMax -
                    entry.RangeMin);
                var code = String.Format( TEMPLATE, entry.Code);

                // add curve points
                var points = new ObservableCollection<Point>();
                for (var stepNr = 0; stepNr <= STEPS; ++stepNr) {
                    var x = entry.RangeMin + (range * stepNr /
                        STEPS);
                    var sx = x.ToString(
                        CultureInfo.InvariantCulture.NumberFormat);
                    var iife = code + "(" + sx + "));";
                    var rslt = engine.Evaluate(entry.Name,
                        true, iife);
                    double y = Math.Min(1e6, Math.Max(-1e6,
                        Convert.ToDouble(rslt)));
                    if (! double.IsNaN(y))
                        points.Add(new Point(x, y));
                }

                // new chart line
                var series = new LineSeries() {
                    DependentValuePath = "Y",
                    IndependentValuePath = "X",
                    Title = entry.Name,
                    IsSelectionEnabled = true,
                    ItemsSource = points,
                    DataPointStyle = new Style(typeof(LineDataPoint))
                };

                // line color
                var brush = new SolidColorBrush(entry.Color);

                // configure the chart line
                var setters = series.DataPointStyle.Setters;
                setters.Add(new Setter(
                    (LineDataPoint.WidthProperty, 0.0));
                setters.Add(new Setter(
                    (LineDataPoint.HeightProperty, 0.0));
                setters.Add(new Setter(
                    (LineDataPoint.BackgroundProperty, brush));

                // add it
                chart.Series.Add(series);
                sFunc = String.Empty;
            }

            catch (Exception e) {
                Console.WriteLine(e);
                if (!String.IsNullOrEmpty(sFunc))
                    errors.Append(sFunc + ": ");
                errors.AppendLine(e.Message);
            }
        }

        timer.Stop();

        if( errors.Length > 0)
            return errors.ToString();
        return String.Format("Aktualisiert in {0}
            Sekunden",Math.Round
            (0.001*timer.ElapsedMilliseconds)/10);
    }
}
```

Die Funktionen selbst sind ansonsten völlig frei per Skriptcode definierbar. Einzige Randbedingung ist, dass die vom Host vordefinierte Variable *x* den Eingangswert enthält, das Ergebnis der Berechnungen muss an die ebenfalls vordefinierte und mit dem Wert 0 initialisierte Variable *y* zugewiesen werden. Was das Skript dazwischen macht, ist tatsächlich gänzlich Sache des Anwenders (Bild 3).

Diagrammpunkte erzeugen

Um die Sache übersichtlich zu halten und genügend Diagrammpunkte zu erzeugen, wird der Skriptcode einfach mehrmals nacheinander mit unterschiedlichen Eingabewerten aufgerufen, die das vorgegebene Intervall mit circa 200 Schritten gleichmäßig abdecken. Die erhaltenen Datenpunkte werden sodann in das Chart übernommen. Der Code dazu könnte so aussehen wie in Listing 1.

Um die eingangs vereinbarte Umgebung ohne Seiteneffekte bereitzustellen, wird der Anwendercode innerhalb einer sogenannten Immediately Invoked Function (IIFE) eingebettet. Das erste und das letzte Statement sind Initialisierung und Rückgabe der lokalen Variablen *y* gewidmet; der vom An-

Listing 2: Gaussssche Normalverteilung

```
function gaussnv( x, m, s ) {
    var a = (s * Math.sqrt( 2 * Math.PI));
    var b = (x - m) / s ;
    return Math.exp( -0.5 *b*b ) / a;
}
y = gaussnv( x, 0, 1);
```

wender eingegebene Skriptcode wird dazwischen platziert. Die Variable *x* mit dem Eingabewert wird stilgerecht an die innere Funktion als Argument *x* übergeben, der aktuelle Wert selbst wird vor dem Aufruf der ClearScript-Methode *Evaluate* an den vorgefertigten Codeblock angehängt.

Weil der Codeblock aber nicht weiter von Bedeutung ist und tatsächlich nur für diesen einen Aufruf Verwendung finden soll, erfolgt der Aufruf von *Evaluate()* mit gesetztem *discard*-Flag. Dies sorgt dafür, dass die Skript-Engine den Codeblock nach der Ausführung komplett verwirft und keine ►

Listing 3: Definition der Formularelemente

```
function GetFormControls() {
    return [
        {
            Id: "Pizza",
            BackColor: { r: 0xff, g: 0xc0, b: 0xc0 },
            LabelText: "Pizza",
            OptionType: OptionEntryType.SelectOne,
            ValueText: "",
            Entries: [
                "Salami",
                "Capricciosa",
                "Hawaii",
                "Quattro Stagioni",
                "Tonno",
                "Spinaci",
                "Prosciutto",
                "Margherita",
                "Diavolo"
            ]
        }, {
            Id: "Size",
            BackColor: { r: 0xff, g: 0xc0, b: 0xc0 },
            LabelText: "Größe",
            OptionType: OptionEntryType.SelectOne,
            ValueText: "",
            Entries: [
                "für Zwischendurch",
                "normaler Pizza",
                "Bärenhunger",
                "Pizza für zwei"
            ]
        }, {
            Id: "Opt1",
            BackColor: { r: 0xff, g: 0xff, b: 0xa0 },
            LabelText: "Option 1",
            OptionType: OptionEntryType.CheckBox,
            ValueText: "Extra Käse",
            ValueBool: false
        }, {
            Id: "Opt2",
            BackColor: { r: 0xff, g: 0xff, b: 0xa0 },
            LabelText: "Option 2",
            OptionType: OptionEntryType.CheckBox,
            ValueText: "Oliven",
            ValueBool: false
        }, {
            Id: "Opt3",
            BackColor: { r: 0xff, g: 0xff, b: 0xa0 },
            LabelText: "Option 3",
            OptionType: OptionEntryType.CheckBox,
            ValueText: "Champignons",
            ValueBool: false
        }, {
            Id: "Coupon",
            BackColor: { r: 0x90, g: 0xa0, b: 0xff },
            LabelText: "Gutscheincode",
            OptionType: OptionEntryType.EditText,
            ValueText: ""
        }
    ];
}
```

Seiteneffekte zurückbleiben. Im Normalfall ist es tatsächlich so, dass man auch mehrere Skripts nacheinander ausführen kann und die Engine zuvor definierte Variablen und Funktionen bei den folgenden Aufrufen noch kennt.

In Module zerlegen

Dieses Verfahren kann man dazu nutzen, Skripts in mehrere Module zu zerlegen. Ebenso lassen sich so Bibliotheken von Drittanbietern einbinden. Oder man unterdrückt dieses Verhalten, wenn es wie in unserem Fall gleich aus mehreren Gründen höchst kontraproduktiv wäre.

Im Skript lassen sich nun alle Funktionen der eingebetteten Skript-Engine wie gewohnt aufrufen und in nahezu beliebig komplexe Routinen verpacken.

Aktuell kann der Funktionsplotter lediglich stetige, zweidimensionale Funktionen plotten. Es ist aber nicht allzu kompliziert, die Schnittstelle auf weitere Dimensionen oder auf parametrisierte Kurven zu erweitern. Das größere Problem dürfte an dieser Stelle tatsächlich sein, eine geeignete Grafikkomponente zu finden, die das auch vernünftig darstellen kann (Listing 2).

Definition und Validierung der Formularelemente erfolgen per Skript (Bild 4)

Unser zweites Beispiel greift einen weiteren Einsatzfall auf. Anwender haben häufig den Bedarf nach Formularen oder Dialogen, die im Hintergrund mit durchaus speziellen und gelegentlich sehr komplexen Geschäftsregeln gespickt sind. Sei es, um den Anwender von einer teuren Fehlkonfiguration seines Wunschcomputers zu bewahren, die Verfügbarkeit aktionsspezifisch verfügbarer Ausstattungsoptionen für KFZ-Modelle zu steuern, Adressen nach bestimmten Kriterien zu validieren oder die Kombina-

tion aus Preis, Gewicht, Abmessungen und Porto eines Online-Einkaufs zu optimieren (Listing 3).

Abhängigkeiten mehrerer Komponenten

Gerade wenn es um Abhängigkeiten mehrerer Komponenten untereinander geht, lassen sich solche Dinge häufig nicht mehr über vordefinierte Tabellen oder vorhandene Optionen mit Bordmitteln abbilden (Bild 4).

Hat der Entwickler das vorhergesehen und eine entsprechend präparierte Skriptumgebung vorbereitet, kann sich der geschulte Anwender dennoch allein behelfen, ohne für jedes Sonderangebot immer gleich eine Programmänderung

Listing 4: Daten können nach beliebig komplexen Regeln validiert werden

```
function ValidateCoupon(ctrIs) {
    if (ctrIs.coupon.length == 0)
        return; // keine Eingabe, Feld leer
    if (ctrIs.coupon.length != 6)
        throw "Ungültiger Coupon-Code";
    var letters = 0, digits = 0, checksum = 0;
    for (var i = 0; i < ctrIs.coupon.length; ++i) {
        c = ctrIs.coupon.charAt(i).toUpperCase();
        if (("A" <= c) && (c <= "Z")) {
            ++letters;
        }
        else if (("0" <= c) && (c <= "9")) {
            ++digits;
            checksum += (c.charCodeAt(0) -
                "0".charCodeAt(0));
        }
    }
    // 3 Ziffern, 3 Buchstaben, Summe der
    // Ziffern durch 9 teilbar?
    if ((digits != 3) || (letters != 3) ||
        ((checksum % 9) != 0))
        throw "Ungültiger Coupon-Code";
};
```

```
function Validate(ctrIs)
{
    var result = new ValidationResult();
    try {
        ValidateOption1(ctrIs);
        ValidateOption2(ctrIs);
        ValidateOption3(ctrIs);
        ValidateOptionCombinations(ctrIs);
        ValidateSize(ctrIs);
        ValidateCoupon(ctrIs);
        result.IsValid = true;
        result.ValidationInfo = "OK";
    }
    catch (e) {
        result.IsValid = false;
        var msg = (e.message || e.toString());
        if (e.number)
            msg = "Error " + e.number.toString() +
                ": " + msg;
        result.ValidationInfo = msg;
    }
    return result;
};
```


und drei zusätzliche Datenbankfelder beantragen zu müssen, die nach Ablauf der Aktion nicht mehr gebraucht werden (Bild 5).

So wäre es denkbar, für Firmenkunden die Eingabe einer Umsatzsteuernummer zu fordern und diese gegen das angegebene Land zu prüfen. Hat sich der Kunde mit einer ge-blacklisteten Mailadressen-Domain angemeldet, ließe sich zusätzlich die Aktualisierung der Mailadresse in den Abschluss der nächsten Bestellung integrieren.

Möglicherweise sind bestimmte Produkte aus irgendwelchen Gründen in bestimmte PLZ-Gebiete nicht lieferbar. Oder der Versand dorthin erfordert besondere Maßnahmen, die wiederum für höhere Transportkosten sorgen. Vielleicht soll dem Kunden automatisch ein spezieller Adapter angeboten werden, sobald er zwei Geräte mit ansonsten inkompatiblen Schnittstellen kauft. All dies ließe sich ad hoc mit Skripts erledigen.

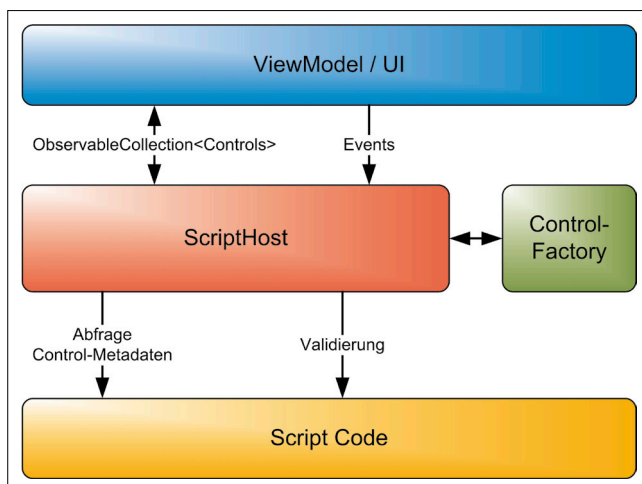
In unserem Beispiel verwenden wir eine zweistufige Logik. Im ersten Schritt definiert ein Teil des Skripts das Layout des Dialogs und die verfügbaren Optionen. Hierzu bedienen wir uns einer Liste mit Einträgen, von denen jeder jeweils eine Dialogoption beschreibt. Im C#-Code der Hostanwendung entsteht aus dieser Beschreibung mittels einer Factory und etwas Data-Binding-Magie eine WPF-Oberfläche.

Sobald nun der Anwender ein Feld editiert, ein Häkchen setzt oder einen Eintrag auswählt wird die Prüfroutine angestoßen. Dazu werden die Formulardaten an eine zweite Skriptfunktion übergeben, die diese auf Konsistenz, Fehlein-gaben und Konflikte prüft.

Hat eine der Prüfroutinen einen Einwand, wirft sie eine Ausnahme. Diese wird an geeigneter Stelle gefangen und in eine Fehlermeldung umgewandelt, die der Host letztlich wiederum via Data Binding in der Oberfläche anzeigt (Listing 4).

Flexible Testautomatisierung

Neben Berechnungen aller Art und flexibler Formular- und Validierungslogik lässt sich Scripting auch an ganz anderer Stelle nutzbringend einsetzen: zur Testautomatisierung. Für



Formular-Tool: Schematische Darstellung der Architektur des Formular-Tools (Bild 5)

Listing 5: Einrichten der Skriptumgebung

```

// create a script engine
using (var engine = new JScriptEngine()) {
    this.Engine = engine;
    // setup env
    var state = new ScriptEnv.SessionState();
    using (var client = new
        FooBar.FooBarServiceClient()) {
        client.Open();
        using (var funcs = new ScriptEnv.Functions
            (state, client, logger)) {
            // Skriptumgebung
            engine.AddHostObject("Client", funcs);
            engine.AddHostObject("State", state);
            engine.AddHostObject("Log", logger);
            // benötigte Datentypen
            engine.AddHostType("Company",
                typeof(FooBar.Company));
            engine.AddHostType("Contact",
                typeof(FooBar.Contact));
            engine.AddHostType("Order",
                typeof(FooBar.Order));
            // run testcase
            logger.WriteLine("Loading "+fname);
            using (var reader = new StreamReader(fname)) {
                var text = reader.ReadToEnd();
                engine.Execute(text);
            }
        }
    }
}

```

den Test von Oberflächen im Webumfeld greifen Entwickler und Tester gern zu Tools wie Selenium; analoge Tools gibt es auch im Desktop-Umfeld. Vorteil ist zweifellos die schnelle Einarbeitung und der attraktive Kostenfaktor.

Analoge Werkzeuge gibt es für SOAP- und REST-Schnittstellen, als bekannter Vertreter der Gattung sei SoapUI genannt. Allen diesen Tools ist eines gemeinsam: Komplexe Abläufe lassen sich nur schlecht testen, da hierzu Logik vonnöten ist und häufig auch Statusdaten zwischengespeichert werden müssen (Bild 6).

Komplexe Abläufe testen

Nehmen wir an, dass unser mittelmäßig komplexes Softwareprodukt unter anderem auch über einen Webservice erreichbar ist, den wir vor der Freigabe gerne auf Herz und Nieren testen möchten. Solche Testsuiten bestehen häufig aus zwei Teilen.

Einerseits werden gezielt einzelne Funktionen geprüft und mit den unterschiedlichsten Parameterkombinationen aufgerufen. Andererseits möchte man auch ganze Testszenarien prüfen, bei denen mehr oder weniger komplexe Abläufe auf Knopfdruck beliebig oft getestet werden können. ►

Ideal wäre es also, wenn unser Testclient als skriptfähige Hostanwendung agiert, sodass wir den komplexen, technischen Teil hinter einer Fassade verbergen können. Diese legen wir in den Testskripts in einer leicht zu programmierenden Schnittstelle offen. Durch diese Strategie lassen sich Flexibilität und einfache Handhabbarkeit optimal kombinieren. Dem Skript werden die erzeugten Objektinstanzen für die Client-Funktionsbausteine und die lokalen Statusdaten wie

gewohnt offeriert. Beide werden im [Listing 5](#) noch durch eine Logger-Klasse ergänzt, über die das Skript seinerseits Ablaufinformationen protokollieren kann.

Kontrolle ist (meist) besser

Für unseren Testclient wäre es in einigen Situationen hilfreich, wenn wir bestimmte Informationen direkt aus dem Skript in eine Datei herunterladen können. So möchten wir

Listing 6: Testskript Webservice-Client

```
(function () {
    var STICHPROBE = 3;
    var Main = function () {
        Log.WriteLine("TestCase start");
        try {
            // Host erfragt und speichert die Credentials
            Client.Login();
            Log.WriteLine("Logged in as " +
                State.Login.FirstName + " " +
                State.Login.LastName);
            Client.GetProductSeries();
            var series = State.ProductData.Series;
            Log.WriteLine(series.Length +
                " series entries found");
            // prod via series
            Log.WriteLine("Series test");
            Log.WriteLine("-----");
            var stichprobe = STICHPROBE;
            for (var i = 0; i < series.Length; ++i) {
                if (--stichprobe < 0)
                    break;
                var idx = Math.round(Math.random() *
                    series.Length);
                var serie = series[idx];
                Log.Write("- Serie " + serie.Display + " ... ");
                Client.SearchProductSeries(serie);
                PrintResultRowsAndErrors();
            }
            // Daten eines Produktes
            if (State.SearchResult.Rows.Length > 0) {
                Log.WriteLine("Product data");
                Log.WriteLine("-----");
                var row = State.SearchResult.Rows[0];
                Client.GetProductDetails(row.ID);
                var img = State.ProductData.Photo.Image;
                var url = img.Url;
                var filename = "c:\\temp\\picture-" +
                    row.ID + img.FileType;
                Log.WriteLine("Downloading " + url);
                DownloadFile(url, filename);
            }
        }
        finally {
            Log.WriteLine("TestCase stop");

            Client.CloseSession();
        }
    };

    var PrintResultRowsAndErrors = function () {
        var msg = [];
        var rowcount = State.SearchResult.Rows.Length;
        if (rowcount > 0)
            msg.push(rowcount.toString() + " row(s)");
        var errcount = State.LastErrors.Length;
        if (errcount > 0)
            msg.push(errcount.toString() + " error(s)");
        Log.WriteLine(msg.join(", "));
    };

    var DownloadFile = function (url, filename) {
        Log.WriteLine("Downloading url " + url);
        Log.WriteLine("Download to file " + filename);
        var webClient = new WebClient();
        webClient.DownloadFile(url, filename);
        Log.WriteLine("Download complete.");
    };

    var HandleException = function (e) {
        var msg = "";
        if (e.message)
            msg = e.message;
        if (e.name)
            msg = e.name + " " + msg;
        if (e.number)
            msg = "Error " + e.number.toString() + ": " + msg;
        if (e.lineNumber)
            msg = "col " + e.lineNumber.toString() + " " + msg;
        if (e.columnNumber)
            msg = "col " + e.columnNumber.toString() + " " +
                msg;
        if (msg == "")
            msg = e.toString();
        Log.WriteLine("ERROR: " + msg);
    };

    // main code
    try {
        Main();
    } catch (e) {
        HandleException(e);
    }
})();
```

Listing 7: WebClient und andere Typen offenlegen

```
// bekannten .NET-Typ für das Skript offenlegen
engine.AddHostType("WebClient",
typeof(System.Net.WebClient));
// Eine Instanz von [Extended]HostFunctions erlaubt
// den vollen Zugriff auf alle Hostfunktionen
engine.AddHostObject("host",
new ExtendedHostFunctions());
// Beispiel aus der Doku: komplette Assemblies
// offenlegen

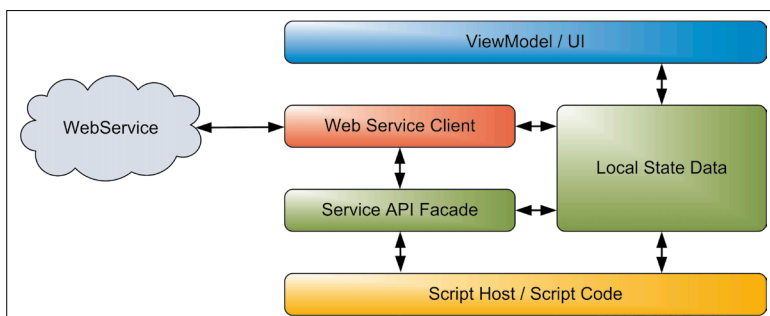
var typeColl = new HostTypeCollection("mscorlib",
"System", "System.Core");
engine.AddHostObject("clr", typeCollection);
// im Skript kann man nun ganz einfach sowas machen:
var guid = clr.System.Guid.NewGuid();
var today = clr.System.DayOfWeek.Friday;
```

zum Beispiel ein Produktfoto herunterladen, um es anschließend mit dem bekannten Muster vergleichen zu können. Man könnte nun eine entsprechende Host-Funktion schreiben, die das erledigt, aber es geht auch anders. Beim Setup fügt man einfach die Zeilen aus Listing 6 hinzu, und schon kann man im Skript einen WebClient erstellen (Listing 7).

Weitreichende Zugriffsrechte

Wie man an den weiteren Beispielen sieht, kann man mit relativ wenigen Statements dem Skript recht weitreichende Zugriffsrechte auf den Host einräumen. Sollte die Quelle des Skripts nicht vertrauenswürdig sein, ist dies aus naheliegenden Gründen allerdings keine gute Idee.

Denn nicht ganz grundlos spricht die Dokumentation davon, dass man mit diesen *HostFunctions*-Klassen die sprichwörtlichen Schlüssel zum Königreich an das Skript aushändigt. Mit diesen Klassen hat das Skript nicht nur Zugriff auf alle möglichen Instanzen und Datentypen in den erreichbaren .NET-Assemblies. Tatsächlich ist es damit leicht möglich, auch ActiveX-Objekte zu instanzieren und sich somit zum Beispiel Zugriff auf das Dateisystem der Hostmaschine zu verschaffen.



Servicetester: Schematische Darstellung der Architektur des Servicetesters (Bild 6)

Listing 8: ContinuationCallback

```
internal class ScenarioRunner {
    private bool AbortFlag;
    internal void Abort() {
        AbortFlag = true;
    }
    internal void Run(string fname) {
        AbortFlag = false;
        using (var engine = new JScriptEngine()) {
            engine.ContinuationCallback +=
                this.ContinuationCallback;
            try
            {
                // ... set up env here ...
                // run testcase (takes ages to complete)
                logger.WriteLine("Loading "+fname);
                using (var reader = new StreamReader(fname)) {
                    var text = reader.ReadToEnd();
                    engine.Execute(text);
                }
            }
            finally {
                engine.ContinuationCallback -=
                    this.ContinuationCallback;
            }
        }
    }
    private bool ContinuationCallback() {
        return (! AbortFlag);
    }
}
```

Um die Sicherheit der Hostanwendung nicht zu kompromittieren, möchte man dem Skript daher lieber eine sehr kontrollierte Umgebung anbieten, die selbigem außerhalb seines intendierten Verwendungszweckes eher wenig Freiheiten und Rechte lässt.

Handelt es sich aber lediglich um ein Tool zur internen Verwendung und bleibt das mögliche Schadenrisiko überschaubar, kann solch weitreichender Zugriff durchaus sinnvoll sein.

Wie häufig bei Security-Aspekten gilt es also, Sicherheitsanforderungen, Zielgruppe und mögliche Risiken auf einer Seite gegen zusätzliche Leistungsfähigkeit und Komfort auf der anderen Seite abzuwägen.

Im Zweifelsfall packt man die offenzulegende Funktionalität besser in ein paar dedizierte Klassen und legt nur diese mittels *AddHostType()* und *AddHostObject()* offen.

Langlaufende Skripts

Ein solches Testskript, einmal gestartet, kann durchaus eine Weile laufen. An einer Stelle hat man den ein oder anderen Parameter etwas zu ►

Listing 9: Skripts abbrechen mittels Interrupt

```

internal class ScenarioRunner {
    private ScriptEngine Engine;
    internal void Abort() {
        try {
            var engine = Engine;
            if (engine != null)
                engine.Interrupt();
        }
        catch (ObjectDisposedException e) {
            // ok, fine.
        }
    }
    internal void Run(string fname) {
        using (var engine = new JScriptEngine()) {
            Engine = engine;
            try {
                // ... set up env here ...

                // run testcase (takes ages to complete)
                logger.WriteLine("Loading "+fname);
                using (var reader = new StreamReader(fname)) {
                    var text = reader.ReadToEnd();
                    engine.Execute(text);
                }
            }
            finally {
                Engine = null;
            }
        }
    }
}

```

großzügig dimensioniert, an einer anderen dauern Aufrufe oder Prüfschritte etwas länger als geplant. Vielleicht möchte man den Test auch einfach nur abbrechen, um die Testsuite mit anderen Einstellungen oder den neuesten Änderungen am Testskript nochmal von vorn zu starten. In solchen Situationen ist es hilfreich, wenn man ein laufendes Skript von außen irgendwie abbrechen könnte.

Dazu bietet ClearScript gleich zwei Mechanismen an. Der eine Weg führt über einen Callback, den man bei der Engine registrieren muss. Dieser Continuation-Callback wird von der Engine in regelmäßigen Zeitabständen aufgerufen und erwartet einen booleschen Rückgabewert. Setzt man ihn auf *false*, wird die Skriptausführung abgebrochen (Listing 8).

Der andere Weg ist die *Interrupt()*-Methode der Engine. Bei einem Aufruf löst diese Methode zeitnah eine *ScriptInterruptedException* aus, die letztlich zum Abbruch der Skriptausführung führt.

Damit der Anwender den Abbruchknopf überhaupt betätigen kann, muss in beiden Fällen die Skriptausführung in einen Sekundär-Thread ausgelagert werden, was bei langlaufenden Skripts generell eine gute Idee ist. Der Aufruf der *Interrupt()*-Methode selbst ist komplett threadsicher und kann somit gefahrlos aus dem UI-Thread erfolgen. Um alle anderen Synchronisierungsmaßnahmen – insbesondere um etwaige Callbacks aus dem Skript-Thread, die UI-relevante Daten verändern – muss sich natürlich wie gewohnt der Entwickler selbst kümmern (Listing 9).

Performance und Alternativen

Trotz moderner V8-Engine ist ClearScript in vielen Szenarien nicht die schnellste Engine am Markt. Unter <http://rushfrisky.com/which-net-javascript-engine-is-the-fastest> finden sich ein Benchmark sowie Links zu alternativen, für den Betrieb in einer .NET-Anwendung geeigneten Engines. Allerdings zeigt ein Benchmark alleine eben nicht die ganze Wahrheit, denn abhängig vom Anwendungsfall kann die Verteilung

durchaus auch anders aussehen. Bereits der Umfang des Skripts spielt hier eine Rolle. Der Einfluss häufiger Zugriffe auf Hostobjekte ist nicht zu unterschätzen.

Clearscript.Manager

Apropos Performance: Empfehlenswert ist in dem Zusammenhang auch der ClearScript.Manager, wiederum von Eric Swann. Dieser Manager verbessert die Performance bei häufiger Skriptausführung, indem präkompilierte Skripts in einem applikationsweiten Pool effizient gecacht werden. Wer sich für die technischen Hintergründe interessiert, sei auf die GitHub-Seite des Projekts verwiesen (<https://github.com/ericswann/ClearScript.Manager>).

Neben der Performance sollten auch weitere Kriterien in die Entscheidung einbezogen werden. Zum Beispiel punktet ClearScript damit, dass die gezielte Offenlegung von Daten und Datentypen so einfach, komfortabel und mächtig zugleich ist. Die Integration in eine existierende Anwendung ist nicht wirklich kompliziert.

Ein weiterer Aspekt ist der Ressourcenverbrauch. Gerade die bei komplexeren Skripts performantere V8-Engine geht nicht zimperlich mit dem verfügbaren Hauptspeicher um. Es hängt also wie so oft vom konkreten Einsatzfall ab, welches Produkt und welche Engine sich für den konkreten Einsatzfall am besten eignet. ■



Jens Geyer

entwickelt bei 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>



Developer Week 2016

20.-23. Juni 2016,
Messe Nürnberg



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

Mit Code

DWX16wmd

€ 149,-
sparen!

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



Gregor Biswanger



Jan Fellien



Dr. Ronald Hartwig



Johannes Hoppe



Patrik Lobacher



Hendrik Lösch



André Krämer



Björn Schotte



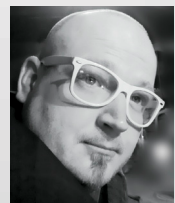
Ulrike Stirnweiß



David C. Thömmes



David Tielke



Holger Wendel

developer-week.de



DeveloperWeek

Aussteller & Sponsoren:

adorsys



brainLight.
LIFE IN BALANCE

Fast Reports
Reporting must be Fast!

Veranstalter:
developer media

Neue
Mediengesellschaft
Ulm mbH



MAXIMAGO

MATHEMA

SDX

Präsentiert von:

SHS VIVEON



TEXTCONTROL

hello, technology

UID

dotnetpro

web & mobile
DEVELOPER

GRAFIK FÜR ENTWICKLER

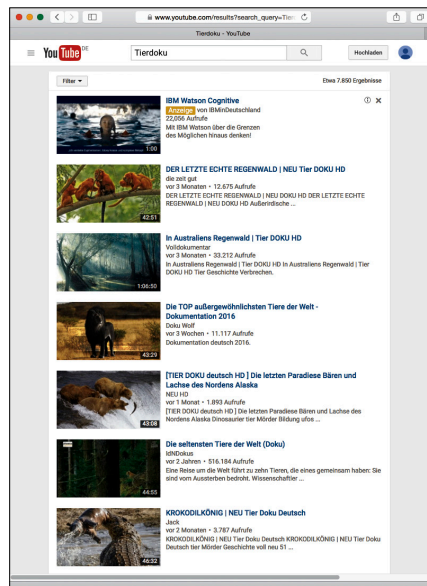
Adobes Videolösungen

Diverse Adobe-Programme helfen bei der Erstellung von Videoclips und Animationen.

Das Internet ist heute nahezu überall verfügbar. Mittlerweile ist eine kabellose Verbindung Standard, ob nun im Büro, daheim oder unterwegs über mobile Geräte. Ist man nicht gerade in einem unbewohnten Tal im Niemandsland, zeigen sich die Funknetze mehr oder weniger flächendeckend. Die Zeiten, in denen Handybesitzer im Urlaub verzweifelt über die Wiesen streiften, um eine Verbindung zu bekommen, gehören fast schon der Vergangenheit an. Und dabei steigt stetig die Geschwindigkeit der Datenübertragung. Zugleich sinken die Kosten für eine Internet-Flat. Die Übertragung von Filmen über das Internet ist also mittlerweile kein Luxus mehr.

Neben Bild und Text liefern Videos zusätzliche Informationen und veranschaulichen so komplexe Themen. YouTube, die wohl bekannteste Plattform für Videos, ist nur eines der vielen prominenten Beispiele (Bild 1).

Vorab lässt sich sagen: Adobe Photoshop CC 2015 liefert eine ansehnliche Arbeitsumgebung für Videoschnitt. Professionelle Werkzeuge hingegen finden sich eher in Premiere Pro. Wer kleinere Animationen erstellen möchte, ist mit dem Doppelpack Adobe Illustrator und Animate gut beraten. InDesign hingegen ist der Spezialist für interaktive PDFs. Zudem lassen sich mit den von Version zu Version wachsenden Mög-



YouTube ist wohl das bekannteste Videoportal, das Videos zur Unterhaltung oder als Informationsträger liefert (Bild 1)

lichkeiten Druckdokumente immer komfortabler in E-Books umwandeln.

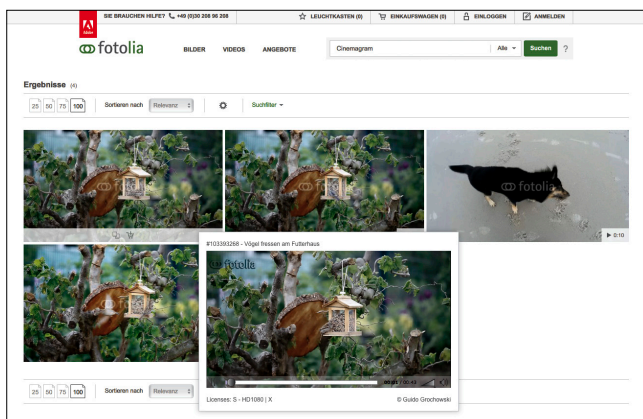
Es gibt verschiedene Möglichkeiten, bewegte Bilder für das Internet und für mobile Anwendungen zu erstellen.

Animiertes GIF: Cinemagram

Bereits im Jahr 1987 stellte der amerikanische Online-Dienst CompuServe das Grafikformat GIF vor. Zwar bot das Format nur eine Palette von 256 Farben, es konnte jedoch Transparenz darstellen – anders als das für das Web damals übliche Ausgabeformat JPG. Mit der verlustfreien LZW-Komprimierung ermöglichte das neue Format auch deutlich kleinere Dateien als andere damals verfügbare Grafikformate, etwa MacPaint. Zudem unterstützte GIF ab der Version GIF89a Animationen.

Zwar wurde GIF ab Ende 1994 von PNG abgelöst, das bei der Darstellung von Einzelbildern deutliche Vorteile im

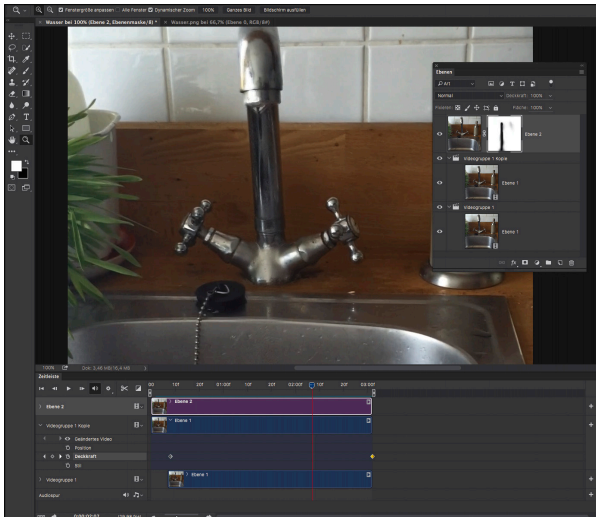
Vergleich zu GIF brachte: Ebenfalls mit einer verlustfreien Kompression versehen, unterstützte PNG neben der Transparenz auch eine Farbdarstellung von acht Bit pro Kanal. Dank der Möglichkeit, Cinemagramme in verschiedene Bilderformen und Datenbanken zu laden, erlebte das Format GIF jedoch ab 2010 ein Comeback: Bei einem Cinemagramm zeigt ein kleiner Bildausschnitt eine Bewegung, die in der Regel in einer Schleife dauerhaft abgespielt wird (Bild 2).



Bilddatenbanken: Auch heute noch bieten verschiedene Bilddatenbanken Cinemagramme zum Kauf an, etwa Fotolia (Bild 2)

Scalable Vector Graphics (SVG)

Das Format SVG (Scalable Vector Graphics) basiert auf XML und kann heute von fast jedem Browser dargestellt werden. Zudem bieten nahezu alle Programme zum Erstellen von Vektorgrafiken SVG als Ausgabeformat an, so auch Illustrator. Lediglich einige Spezialitäten von Illustrator, etwa das Verlaufsgitter, können von der Auszeichnungssprache XML nicht direkt umgesetzt werden; in diesem Fall gehen die Bearbeitungsmöglichkeiten von Illustrator verloren. Grafiken aus Illustrator können zwar im SVG-Format gespeichert werden, lassen sich dort jedoch nicht direkt animieren. Die Bewegung der Formen wird nach dem Erstellen in einem Texteditor festgelegt – hierzu gibt es beispielsweise das `animate`-Element.



Die Herstellung eines Cinemagramms gelingt in Photoshop CC 2015 dank Ebenen und Videobearbeitung (Bild 3)



Soll das GIF animiert sein, muss die Exportoption *Für Web speichern* gewählt sein (Bild 4)

Um diese spezielle Form der GIF-Animation zu erstellen, wird zunächst die entsprechende Szene gefilmt. Am besten gelingt das mit einem Stativ, um Verwackelungen im Hintergrund zu vermeiden. Bei dem Bildausschnitt, der die Bewegung zeigt, sollte darauf geachtet werden, dass Start- und Endpunkt der Animation möglichst identisch sind, um einen überzeugenden Loop zu kreieren. Um die Datenmenge möglichst gering zu halten, sollte sich die Bewegung in einem möglichst kleinen Bildbereich abspielen und darf nicht zu lange dauern. Im Beispiel haben wir einen tropfenden Wasserhahn gewählt.

Der fertige Film wird nun in Photoshop CC geöffnet und erscheint dort auf der Zeitleiste. Hier lassen sich Start und Ende exakt festlegen. Eine überblendete Kopie des Ausschnitts sorgt für einen ruckelfreien Loop. Sollten weitere Bildbereiche Bewegungen zeigen, müssen diese abgedeckt werden. Hierzu dient ein bewegungsfreies Einzelbild der Szene, das

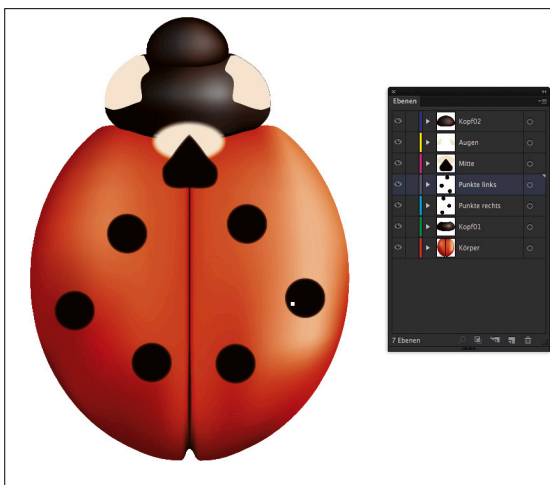
über den Videoclips liegt. Per Maske werden die animierten Bereiche darunter wieder sichtbar gemacht (Bild 3).

Eine Animation kann nicht über den normalen Befehl *Speichern unter* gesichert werden, da hier nur Einzelbilder gesichert werden können. Über *Exportieren* und *Für Web speichern* gelangt man in den Export-Dialog, der auch die GIF-Animation berücksichtigt. Neben den exakten GIF-Einstellungen kann hier die Größe geändert und für die Animation die Anzahl der Wiederholungen festgelegt werden (Bild 4).

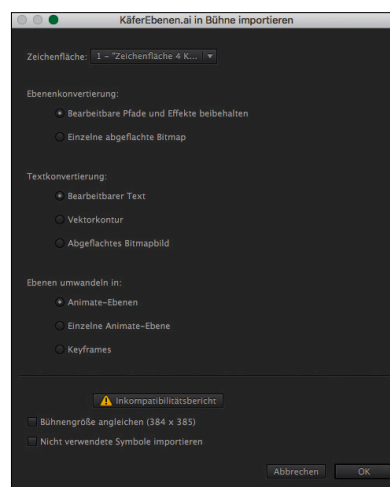
Auch für andere kleinere Videoprojekte hält Photoshop CC verschiedene nützliche Funktionen bereit. Alle Werkzeuge und Funktionen dazu sind in der Arbeitsumgebung *Bewegung* zu finden.

So lassen sich beispielsweise Korrektorebenen auch auf Videoclips anwenden, verschiedene Videos und Bilder miteinander kombinieren, oder auch bereits in der Videodatei verankerten Ton bearbeiten und neuen Sound hinzumischen.

Auch Text lässt sich platzieren und über die Zeitleiste animieren. Neben der zuvor beschriebenen Arbeit mit Videos lassen sich in Photoshop CC 2015 ebenfalls einzelne Frames für eine herkömmliche GIF-Animation erstellen. Wer sich also mit Photoshop bereits auskennt, muss sich für einfache Projekte nicht unbedingt in eine professionelle Videoschnitt-Software einarbeiten.



Illustrator: Um später die einzelnen Teile des Käfers zu animieren, wurden sie in Illustrator auf einzelne Ebenen gelegt (Bild 5)



Animate CC 2015: Illustrator-Ebenen lassen sich in Animate-Ebenen oder in Keyframes umwandeln (Bild 6)

Vektorgrafik aus Illustrator

Zwar gibt es in Photoshop einige Werkzeuge zum Erstellen und Bearbeiten von Vektorgrafiken, für komplexere For- ►

men liefert jedoch Illustrator weitaus ausgefeiltere Tools. Soll eine solche Grafik dann aber animiert werden, zeigen sich die Grenzen des Zeichenprogramms: Funktionen zur Animation fehlen darin schlichtweg.

Hier scheint es sich zunächst anzubieten, die Datei in Photoshop zu bewegen. Denn eine Illustrator-Grafik kann in Photoshop als Smartobjekt platziert werden, sodass auch bei Transformationen, etwa einer Änderung der Größe oder eine Drehung, die originalen Daten weiterhin bereitstehen. Ist so beispielsweise eine Vektor-Figur verkleinert worden, kann sie in den nächsten Frames wieder bis zur ursprünglichen Größe wachsen und zeigt weiterhin gestochen scharfe Kanten.

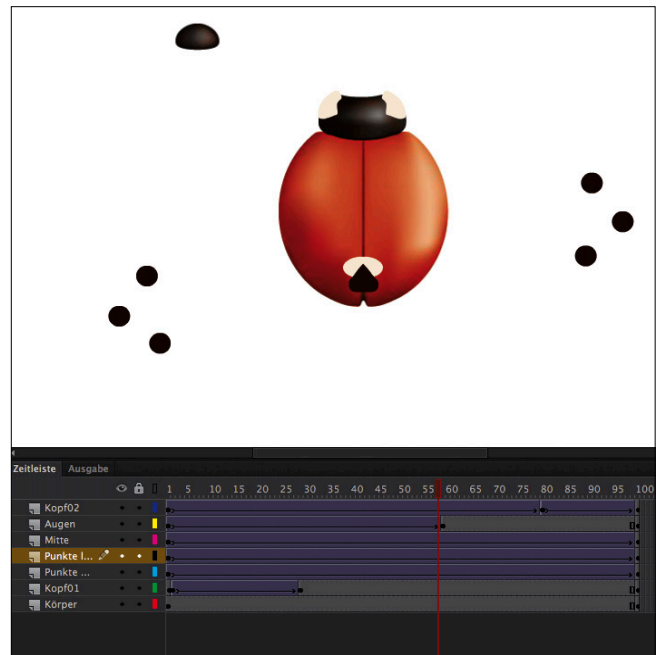
Frame-Animation

Leider lassen sich besagte Vorteile bei einer Frame-Animation nicht uneingeschränkt nutzen. Größenänderungen von Frame zu Frame beispielsweise ignoriert das Bildbearbeitungsprogramm beharrlich. Zudem ist es nicht möglich, einzelne Flächen der Grafik zu animieren. Illustrator-Ebenen erkennt Photoshop nicht.

Für solche Aufgaben ist Adobe Animate CC, ehemals Flash, das Programm der ersten Wahl. Wichtig ist die entsprechende Vorarbeit in Illustrator: Sollen Bestandteile einzeln animiert werden, müssen sie bereits in Illustrator auf extra Ebenen liegen. Je mehr Ebenen es sind, desto wichtiger ist eine schlüssige Benennung (Bild 5).

Diese werden danach in ein neues Animate-Dokument importiert. Hier bietet der Import-Dialog des Programms die Option, alle Illustrator-Ebenen in Animate-Ebenen umzuwandeln (Bild 6).

Eine andere Möglichkeit der Zusammenarbeit zwischen Illustrator und Animate CC besteht darin, die verschiedenen Zustände der späteren Animation auf einzelne Ebenen zu legen. In diesem Fall kommt die Option *Keyframes unter Ebenen umwandeln* in zum Zuge. Handelt es sich jedoch um ers-tere Variante, also um eine Ebenen-Animation, liegen die Bestandteile der einzelnen Ebenen in Adobe Animate untereinander in der Zeitleiste und können so einzeln bearbeitet werden (Bild 7).



Importierte Illustrator-Ebenen können in Animate einzeln animiert werden (Bild 7)

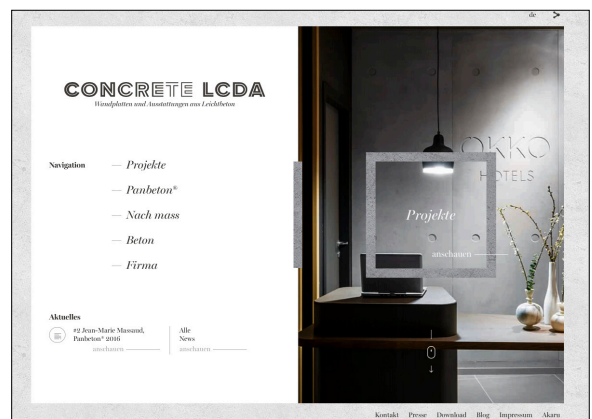
InDesign bietet verschiedene Optionen, Elemente zu bewegen und diese Abfolge an Bewegungen als SWF-Animation zu speichern. Auch das generische Format von Adobe Animate .fla steht bereit, um die in InDesign angelegten Daten weiter in Adobes Animationsprogramm Animate zu bearbeiten.

Bereits beim Erstellen eines neuen Dokuments stehen dem Nutzer als Zielmedium die digitale Veröffentlichung oder die Option *Web* zur Verfügung. Alle zugehörigen Funktionen und Optionen zeigt dann der Arbeitsbereich *Digitale Veröffentlichung*.

Zum Animieren von Bild und Text steht das Bedienfeld *Animation* bereit. Hier finden sich – übernommen von Flash – unterschiedliche Bewegungsformen, deren Bewegungspfade und -Formen sich weiter bearbeiten lassen. Gibt es mehrere Animationen, die zeitlich versetzt ablaufen sollen, kann dies



Für Animationen in InDesign gibt es eine eigene Arbeitsumgebung, zu der auch das Vorschau-Bedienfeld gehört (Bild 8)



Für Bild- und Textanimationen, wie sie diese Webseite zeigt, liefert InDesign die notwendigen Werkzeuge (Bild 9)

im Bedienfeld *Zeitpunkt* eingestellt werden.

Auch der Import von Videodaten in ein InDesign-Dokument ist über den gewohnten *Platzieren*-Dialog möglich. Rahmen und Videoinhalte können dann wie ein Bild transformiert werden. Für die Filmeinstellungen ist das Bedienfeld *Medien* zuständig: Hier kann beispielsweise ein Standbild gewählt werden, das vor dem Abspielen des Films im Rahmen gezeigt wird (**Bild 8**).

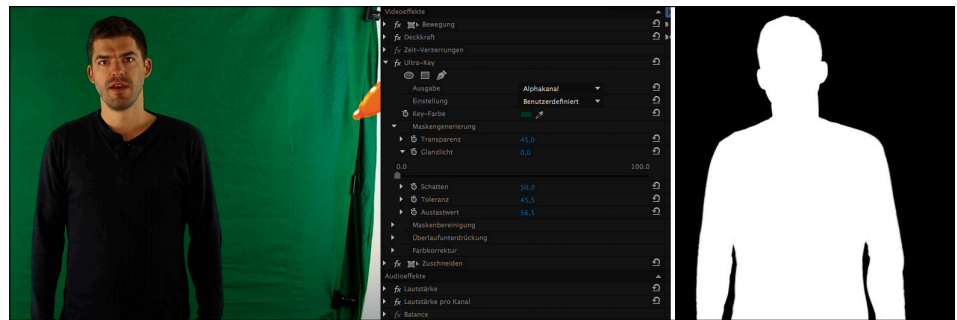
Weiter ist es möglich, in InDesign animierte Seitenübergänge zu erstellen. Zudem gibt es eine Reihe von Schaltflächen, die mit Aktionen belegt werden können.

Nicht nur zum Erstellen von interaktiven E-Books eignen sich die einzelnen Funktionen und Werkzeuge aus InDesign. Webdesigner finden hier auch intuitiv bedienbare Tools für Webanimationen. Hier punktet die Möglichkeit, Bilder und Text gleichermaßen schnell und problemlos bewegen zu können, etwa für eine Startseite. Ein gutes Beispiel liefert die Webseite www.concrete-beton.com: Hier bietet die Startseite eine ästhetisch aufgebaute Animation von Bild und Text, deren Gestaltung und technische Umsetzung sich durchaus in InDesign realisieren ließe (**Bild 9**).

Der Profi: Premiere Pro

Ist eine aufwendigere Videoproduktion angedacht, reichen die Funktionen und Werkzeuge von Photoshop nicht mehr aus. Weitaus professionellere Möglichkeiten bieten etablierte Videoschnitt-Lösungen wie Final Cut Pro von Apple, Lightworks oder Premiere Pro CC. Diejenigen, die ein Adobe-Cloud-Abo haben, sind mit Premiere bestens beraten.

Sieht man sich jedoch die Oberfläche der Schnittsoftware an, ist nur wenig Bekanntes aus der Arbeitsfläche *Bewegung* von Photoshop zu finden. Wer also bislang nichts mit Videoschnitt zu tun hatte, muss mit einer nicht unerheblichen Einarbeitungszeit rechnen (**Bild 10**).



Zum Freistellen der Person wird ein sogenannter Greenscreen eingesetzt (links), der dann über Filter (Mitte) in Premiere transparent geschaltet wird (**Bild 11**)

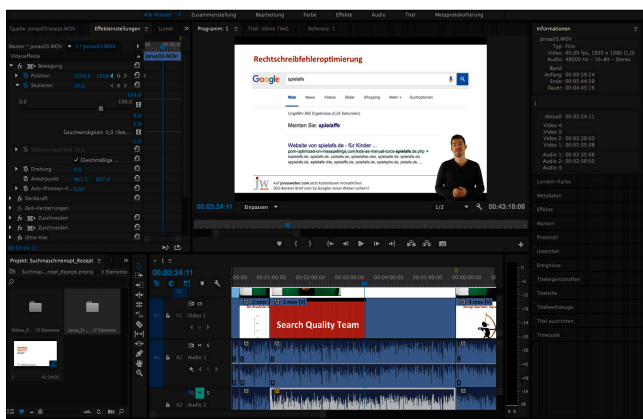
Hatte Final Cut bei einigen Funktionen die Nase vorn, so hat Premiere schon längst aufgeholt. Mittlerweile ist es etwa bequem möglich, verschiedene Tonspuren zu synchronisieren, sollte zum Beispiel die Tonaufnahme eines externen Aufnahmegeräts gegenüber dem Sound der Kamera bevorzugt werden. Zudem gelingen auch anspruchsvolle Aufgaben wie das Freistellen eines Redners, um einen anderen Hintergrund einzublenden. Sinnvoll ist es dabei, eine Bluebox oder einen Greenscreen bei der Aufnahme zu verwenden. Per Ultra-Key wird dann der entsprechende Farbwert transparent geschaltet. Für Abweichungen in der Hintergrundfarbe liefert der Key verschiedene stufenlos einstellbare Parameter, etwa Glanzlicht, Schatten oder Transparenz (**Bild 11**).

Um viele Clips hintereinander in einem Rutsch zu rendern, hat Adobe den Media Encoder in die Cloud gepackt. Hier lassen sich direkt über Drag and Drop alle Premiere-Dateien in den Hauptbereich ziehen. Alternativ dazu gelangt das zu berechnende Projekt in Premiere Pro im Dialog *Exporteinstellungen* per Klick auf *Warteschlange* in den Media Encoder.

Fazit

Adobe liefert mit der Creative Cloud viele Werkzeuge, die professionelles Videomaterial und Animationen ermöglichen. Das hat jedoch seinen Preis: Gerade komplexere Aufgaben sind trotz der durchaus stattlichen Videooptionen in Photoshop nicht mehr machbar. Hier kommt der ambitioniertere Anwender um Premiere Pro CC nicht mehr herum.

Zudem bietet Illustrator noch nicht einmal die rudimentärsten Funktionen zur Videoerstellung. Selbst einfache Animationen einzelner Bestandteile einer Grafik sind so nicht möglich. Auch Photoshop bietet hier keine Lösung; nur Adobes Animate kann mit Illustrator-Ebenen umgehen. ■



Premiere Pro zeigt eine Vielfalt an Funktionen, deren Bedienung man sich jedoch zunächst erarbeiten muss (**Bild 10**)



Katharina Sckommodau

arbeitet als freiberufliche Autorin, Grafikerin und Dozentin, unter anderem für die Akademie der Bayerischen Presse und für Macromedia.

DEN MIKROCONTROLLER ESP8266 PROGRAMMIEREN

Das Internet für die Dinge

Der Baustein ESP8266 befördert Ihr Projekt kostengünstig auf die Datenautobahn.

Arduino und Raspberry Pi sind als einfache Prototyp-Platinen in aller Munde. Neben diesen beiden Boards gibt es aber noch viele andere Mikrocontrollerboards, die auch jenseits von Embedded-Projekten für Software-Entwickler interessant sind. Ein besonders Vertreter dieser Gattung ist der SoC (System on a Chip) ESP8266EX, der von der Firma Expressif Systems aus Shanghai entwickelt wurde.

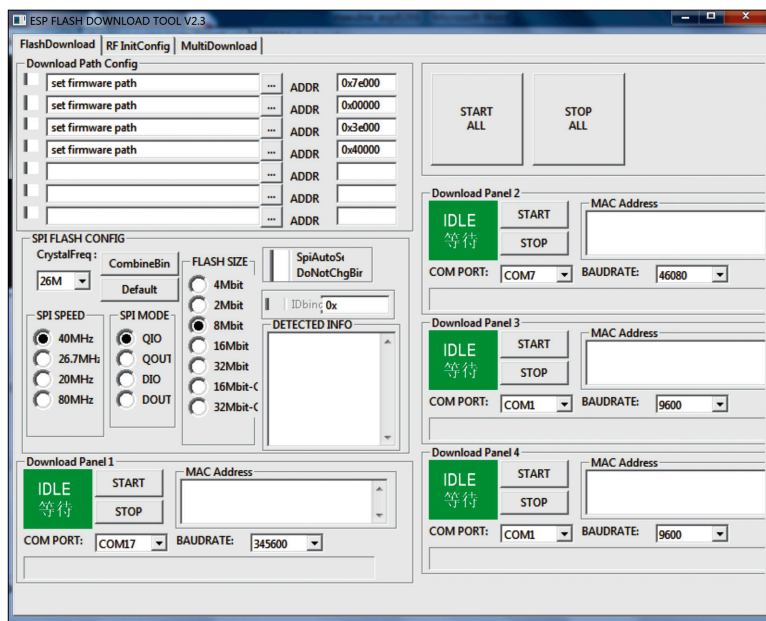
Beim ESP8266EX handelt es sich nicht um einen einzelnen Mikrocontroller, sondern um eine ganze Modellfamilie – so wie die Arduino-Familie. Die einzelnen Ausprägungen des SoC-Moduls tragen Nummern, wie zum Beispiel ESP8266-07 oder ESP8266-12.

Inzwischen werden auf der Webseite von Expressif System insgesamt 15 verschiedene Modelle aufgeführt (Stand: März 2016). Die Mikrocontroller bekommt man meistens auf einer kleinen Platine mit herausgeführten Pins, denn der Chip selbst ist schwierig aufzulöten (Bild 1).

Mikroprozessor mit WLAN-Modul

Die Besonderheit am ESP8266EX-Baustein ist die Kombination von Mikroprozessor mit einem WLAN-Modul auf einem SoC, und das Ganze bekommt man teilweise schon für drei Euro.

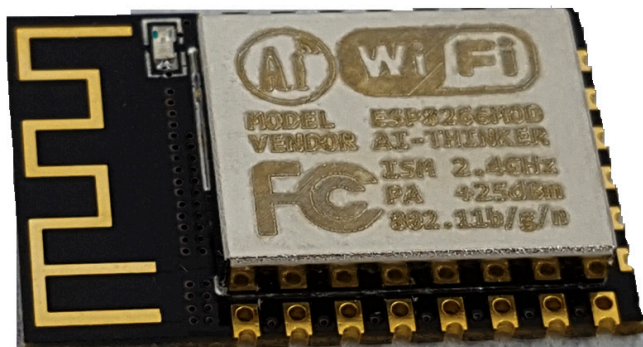
Für Nutzer von Arduino und Raspberry Pi ist besonders interessant, dass der ESP8266EX sehr einfach über die serielle UART-Schnittstelle (Universal Asynchronous Receiver Transmitter) mit diesen verbunden werden kann. Wobei sich das beim Raspberry Pi 3 Model B erledigt hat, denn dieser kommt



FlashTool von Expressif Systems steht nur für Windows zur Verfügung (Bild 2)

bereits mit integrierter WLAN-Schnittstelle. Für die anderen Raspberry-Pi-Modelle ist die Kombination aber weiterhin interessant.

Das WLAN-Modul unterstützt dabei die Standards 802.11 b/g/n und hat einen integrierten TCP/IP-Stack. Als Verschlüsselungsstandards werden WPA und WPA2 genutzt. Bei 802.11n muss erwähnt werden, dass der vom Standard optional spezifizierte 5-GHz-Bereich nicht unterstützt wird, sondern lediglich über 2,4 GHz gefunkelt werden kann.

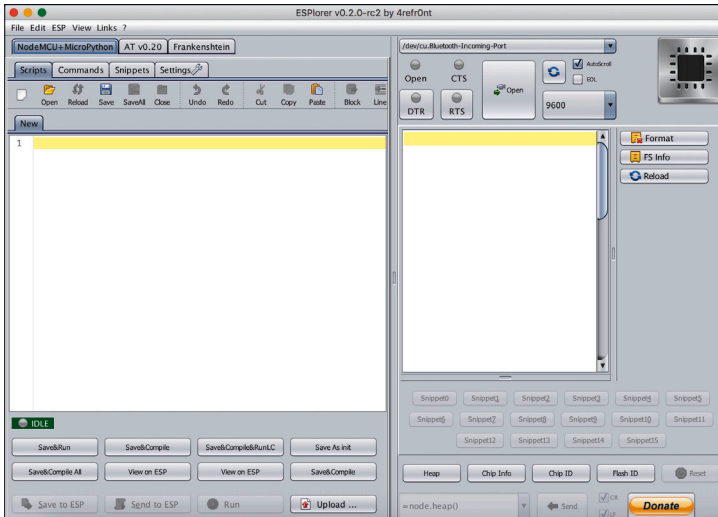


Die ESP8266-Familie hat zahlreiche Mitglieder: hier im Bild ein ESP8266-13 (Bild 1)

Testumgebung

Für den Artikel wurde Mac OS X 10.11.3 mit einer JRE in der Version 8 Update 73 (1.8.0_73-b02) verwendet.

ESPlorer wurde in der Version v0.2.0-rc2 eingesetzt. Als Arduino-IDE kam die Version 1.6.7 zum Einsatz. Als Terminalprogramm diente CoolTerm in der Version 1.4.6. Als Boards für die Tests wurden das SparkFun ESP8266 Thing und das NodeMCU Version 1.0 verwendet. Für die Kommunikation mit dem SparkFun ESP8266 Thing wurde ein 3,3 V FTDI Basic Breakout von SparkFun eingesetzt. Die Spannungsversorgung der Boards erfolgte über die USB-Schnittstelle am Host-PC.



Mit dem ESPlorer kann der ESP8266 direkt programmiert werden (Bild 3)

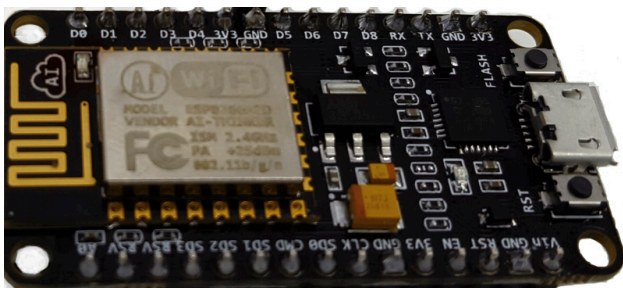
Als Prozessor kommt ein Xtensa LX106 (32 Bit) von Tensilica zum Einsatz. Der Prozessor arbeitet mit einem Systemtakt zwischen 80 bis 160 MHz, wobei der Prozessor auf dem SoC des ESP8266 mit 80 MHz getaktet ist. Der Prozessor besitzt einen Befehlsspeicher von 64 KByte und einen Datenspeicher von 96 KByte.

Der SoC hat keinen ROM für Programme, diese müssen in einem per SPI angebundenen Flashspeicher abgelegt werden. Theoretisch sind bis 16 MByte ansprechbar, empfohlen sind aber 512 KByte bis 1 MByte.

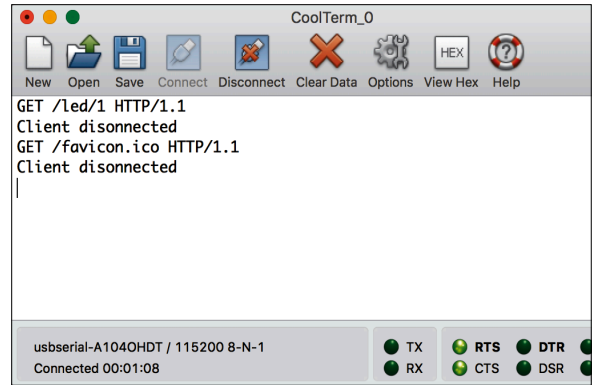
Für die Verbindung mit anderen Mikrocontrollern oder zur Anbindung von Elektronik stehen GPIO-Ports zur Verfügung. Über diese Pins steht I2C, SPI und auch UART zur Verfügung. Zusätzlich bietet der ESP8266 einen analogen Port mit einer Auflösung von 10 Bit.

Hoher Stromverbrauch

Vor der ersten Nutzung des ESP8266 ist wichtig zu wissen, dass die Spannungsversorgung nur 3,3 V liefern darf, denn der ESP8266 ist nicht tolerant gegenüber 5 V. Leider verbraucht der ESP8266 relativ viel Strom: Beim Senden über 802.11 fallen 170 mA an. Wer das WLAN nicht die gesamte Zeit benötigt, sollte daher das Modul in den Tiefschlaf (Deep-Sleep-Modus) versetzen und damit den Stromverbrauch auf 10 µA senken.



Die NodeMCU-Platine kommt direkt mit USB-Anschluss und benötigt keinen separaten USB-Seriell-Wandler (Bild 5)



CoolTerm bietet vielfältige Einstellmöglichkeiten für die Verbindung (Bild 4)

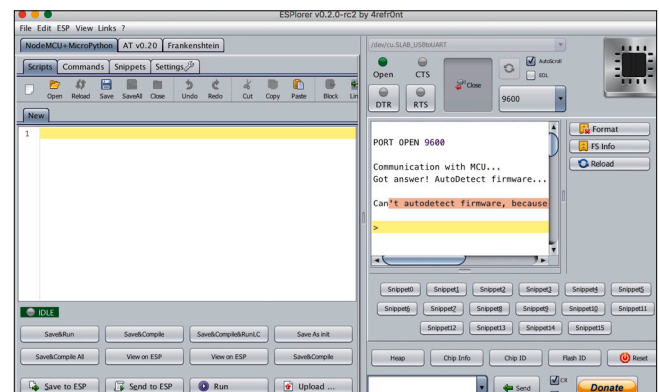
Das Betriebssystem für einen Mikrocontroller wird als Firmware bezeichnet. Für den ESP8266 gibt es nicht nur eine einzige Firmware, sondern es stehen mehrere Varianten zur Verfügung. Die einzelnen Firmwares sind sehr unterschiedlich, und deshalb hängt die Nutzung des Chips stark von der installierten Firmware ab.

Auf die Firmware kommt es an

Die Standardfirmware des Herstellers wird im Rahmen des SDK mitgeliefert. Diese Firmware ermöglicht die Kommunikation über AT-Befehle, die der eine oder andere Leser vielleicht noch aus dem Modemzeitalter kennt. Deswegen wird diese Firmware auch als AT-Firmware bezeichnet.

Daneben gibt es eine Arduino-Firmware, um den ESP8266 über die Arduino-IDE zu programmieren. Mit Micropython kann die Programmiersprache Python eingesetzt werden und bei Smart.js JavaScript. Für Anhänger von BASIC besteht die Möglichkeit, eine BASIC-Firmware zu nutzen. Mittels NodeMCU kann der ESP8266 mit der Skriptsprache Lua programmiert werden.

Um eine bestimmte Firmware auf den ESP8266 zu laden, muss dieser mit der jeweiligen Version geflasht werden. Hierzu bietet Expressif Systems ein Flash-Tool an – leider nur für Windows. Alternativ steht das Python-Skript esptool für das Update der Firmware als Download bereit (Bild 2).

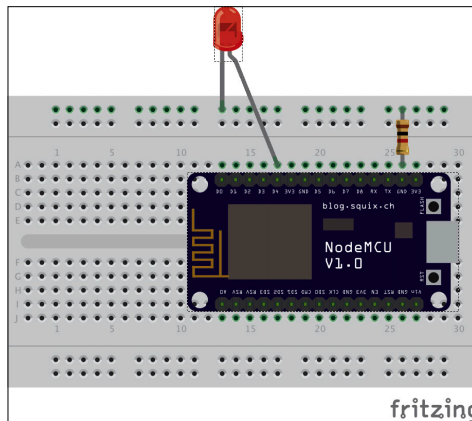


Ausgaben des NodeMCU-Boards sind im rechten Teil des ESPlorers zu sehen (Bild 6)

Wie bereits erwähnt, stellt Expressif Systems ein SDK zur Entwicklung eigener Firmwares zur Verfügung. Für den Einsteiger ist dies sicherlich nicht relevant, wer sich aber dafür interessiert, kann sich schon fertige VirtualBox-Images mit komplett aufgesetzter Toolchain herunterladen.

Wer den ESP8266 mit einer bereits vorhandenen Firmware einsetzen möchte, der kann auf die Installation des SDK zunächst verzichten. Eine Installation der Entwicklungsumgebung ESPlorer (Bild 3) ist dabei aber sehr zu empfehlen. ESPlorer hat eingebautes Syntax-Highlighting für Python und Lua.

Mit dieser Umgebung können Kommandos über die serielle Schnittstelle an den ESP8266 gesendet werden. Bei ESPlorer



Die Arbeit mit einem Vorwiderstand ist zu empfehlen (Bild 7)

programmieren. Hierfür wird ein ESP8266 mit aufgespielter NodeMCU-Firmware benötigt. NodeMCU basiert auf eLua, hierbei handelt es sich um Lua für eingebettete Systeme. Um das nicht selbst aufspielen zu müssen, wird für diese Zwecke das Entwicklungsboard NodeMCU Dev-Kit Version 1.0 verwendet (Bild 5).

Integrierter USB-TTL-Wandler

Das Modul mit NodeMCU-Firmware kommt mit einem ESP8266-12E Mikrocontroller. Es hat bereits einen USB-TTL-Wandler integriert und kann damit ohne zusätzlichen Kon-

verter am USB-Port des PC betrieben werden. Das Modul besitzt 10 GPIO-Ports, die alle als PWM (Pulsweitenmodulation) nutzbar sind. Zusätzlich können darüber die Protokolle I2C und 1-Wire genutzt werden.

Für die Kommunikation mit dem Board muss zuerst der CP210x-USB-Treiber installiert werden; diesen finden Sie für die wichtigsten Betriebssysteme auf der Silicon-Labs-Webseite. Nach der erfolgreichen Treiber-Installation kann das Modul über ein serielles Terminal (CoolTerm oder RealTerm) genutzt werden, oder wie in diesem Fall mit dem ESPlorer.

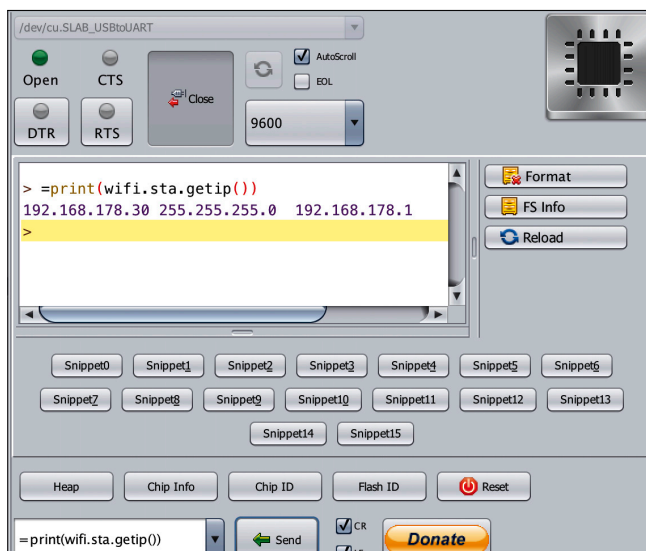
Standardmäßig wird NodeMCU mit einer Einstellung von 9600 Baud ausgeliefert. Somit belassen Sie es bei dieser Einstellung im ESPlorer. Anschließend müssen Sie noch die serielle Schnittstelle auswählen und über *Open* die Verbindung zum Board herstellen (Bild 6).

Bei der ersten Verbindung mit dem Board müssen die Befehle wie im nachfolgenden Listing ausgeführt werden:

```
file.remove("user.lua")
node.restart()
```

Nun kann das Board programmiert werden. Zuerst soll eine an das Board angeschlossene LED zum Blinken gebracht werden. Hierfür wird zuerst die Schaltung gemäß der Skizze in Bild 7 aufgebaut.

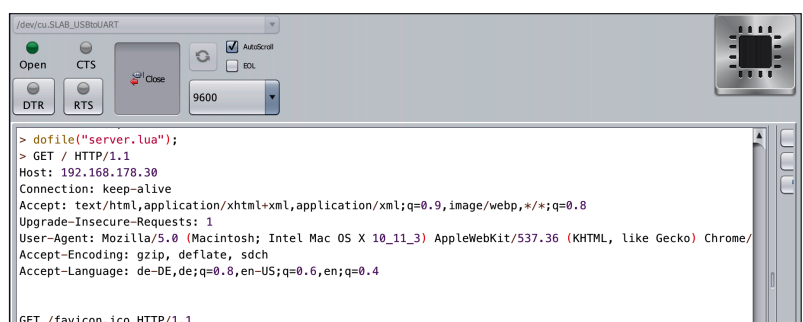
Die Programmierung in Lua ähnelt der C-Programmierung. Für die Nutzung der GPIO stehen bereits vordefinierte Funk-



Befehle werden über die Schaltfläche *Send* ausgeführt (Bild 8)

rer handelt es sich um eine Java-Applikation, daher ist auch die Installation einer JRE erforderlich. Um auf den ESP8266 zugreifen zu können, muss zuvor der entsprechende Treiber für die serielle Kommunikation installiert werden. Welcher Treiber installiert werden muss, hängt vom verwendeten Board ab und auch davon, ob ein separater USB-Seriell-Wandler verwendet wird oder nicht. Zusätzlich ist die Installation eines seriellen Terminalprogramms zu empfehlen. Damit haben Sie jederzeit die Ausgaben des ESP8266 im Blick – unabhängig von der installierten Entwicklungsumgebung. Für Windows bietet sich RealTerm an und für Mac OS X das Programm CoolTerm (Bild 4).

Nun soll der ESPlorer genutzt werden, um den ESP8266 mit der Skriptsprache Lua zu



Ausgabe des HTTP-Servers im ESPlorer (Bild 9)


```
mstaeuble — telnet 192.168.178.30 2000 — 80x24
Markuss-MBP-2:~ mstaeuble$ telnet 192.168.178.30 2000
Trying 192.168.178.30...
Connected to 192.168.178.30.
Escape character is '^]'.
NodeMCU - Lua
print(wifi.sta.getip())
192.168.178.30 255.255.255.0 192.168.178.1
>
```

Direkte Ausführung von Lua-Kommandos über den gestarteten Telnet-Server (Bild 10)

tionen über das Lua-Modul *gpio* bereit. Da die GPIO als Ein- oder Ausgang genutzt werden kann, wird zunächst über *gpio.mode()* festgelegt, dass Pin 4 ein Ausgang (*gpio.OUTPUT*) sein soll. Nun wird im Zweisekudentakt die LED an- und ausgeschaltet:

```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
        gpio.write(pin,gpio.LOW)
    end
end)
```

Als Nächstes soll die WLAN-Fähigkeit des Moduls ausgenutzt werden und dieses auch mit einem vorhandenen WLAN-Netzwerk verbunden werden. Zuerst wird über *wifi.sta.getip()* geprüft, ob das Modul bereits eine IP-Adresse besitzt. Um die Ausgabe einer Konsole im Ausgabefenster von

Listing 1: Telnet-Server für Lua-Kommandos

```
-- Starte Telnet Server auf Port 2000
s=net.createServer(net.TCP)
s:listen(2000,function(c)
    function s_output(str)
        if(c~=nil)
            then c:send(str)
        end
    end
    -- Ausgaben umleiten
    node.output(s_output, 0)
    c:on("receive",function(c,l)
        node.input(1)
    end)
    c:on("disconnection",function(c)
        node.output(nil)
    end)
    print("NodeMCU - Lua")
end))
```

Was ist ein Bootloader?

Programme für einen Mikrocontroller werden in einem Festspeicher abgelegt, zum Beispiel einem EPROM (Erasable Programmable Read-Only Memory).

Die Alternative zu dieser Art der Programmierung ist ein Bootloader, der auch von den Atmel-Mikrocontrollern auf dem Arduino eingesetzt wird. Hierbei handelt es sich um ein Programm, dessen Aufgabe es ist, das auszuführende Programm in den Speicher zu laden. Ein Bootloader führt somit das sogenannte Bootstrapping durch. Der Bootloader steht dabei in einem besonderen Bereich des Speichers, damit dieser nicht von einem Programm überschrieben wird. Bei AVR-Mikrocontrollern (zum Beispiel Arduino) wird der Bootloader dafür genutzt, dass das Board über USB und über die Arduino IDE programmiert werden kann, anstatt über ICSP (Integrated-Circuit Serial Programming). Dies nimmt dem Entwickler einiges an lästiger Arbeit ab und macht die Programmierung dadurch erheblich leichter – einer der Gründe für den großen Erfolg der Arduino-Plattform.

ESPlorer zu sehen, muss diese über die Funktion *print()* ausgegeben werden. In diesem Fall also *print(wifi.sta.getip())*. Sollte keine Adresse zugewiesen sein, so wird als Rückgabewert *nil* zurückgegeben.

SSID des Netzwerks

Über die Funktion *wifi.sta.config()* kann eine Verbindung zu einem erreichbaren WLAN-Netzwerk aufgebaut werden. Hierfür wird als erster Parameter die SSID des Netzwerks übergeben und als zweiter Parameter das Passwort. Nun kann erneut die IP-Adresse über *wifi.sta.getip()* abgefragt werden. Die im nachfolgenden Listing aufgeführten Befehle (beginnen mit einem >) werden im Eingabefeld neben *Send* im ESPlorer eingegeben (Bild 8):

```
> print(wifi.sta.getip())
nil
> wifi.sta.config("meinNetzwerk", "MyPasswort") ▶
```

Listing 2: Blinken der LED 5 auf dem SparkFun Thing

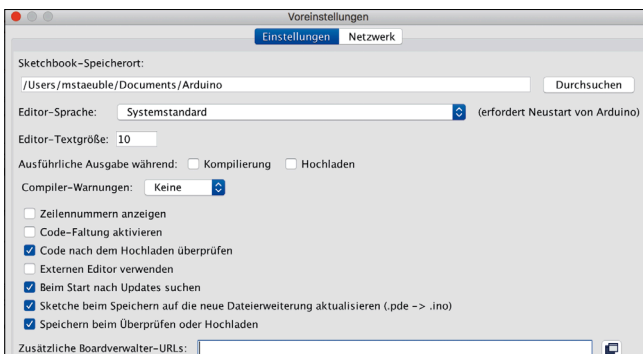
```
#define ESP8266_LED 5
void setup() {
    pinMode(ESP8266_LED, OUTPUT);
}
void loop() {
    digitalWrite(ESP8266_LED, HIGH);
    delay(500);
    digitalWrite(ESP8266_LED, LOW);
    delay(500);
}
```

```
> print(wifi.sta.getip())
192.168.178.30    255.255.255.0    192.168.178.1
```

NodeMCU kann auch einen Server starten. Als erstes Beispiel soll ein einfacher HTTP-Server unter Port 80 gestartet werden. Bei erfolgreicher Verbindung gibt der Server *Hallo, NodeMCU* aus. Dies kann über einen beliebigen Browser getestet werden. Die Ausgabe des HTTP-Servers kann man im Explorer verfolgen (Bild 9):

```
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
  conn:on("receive",function(conn,payload)
    print(payload)
    conn:send("<h1> Hallo, NodeMCU.</h1>")
  end)
end)
```

Mit dem gleichen Prinzip kann auch ein Telnet-Server gestartet werden. Über die Funktion *createServer()* aus dem Lua-Modul *net* wird der Server gestartet. Als Port wird 2000 gewählt (Listing 1). Nach erfolgreicher Verbindung nimmt der Server Lua-Kommandos entgegen, führt diese aus und gibt gegebenenfalls die Ausgaben in der Konsole aus (Bild 10).

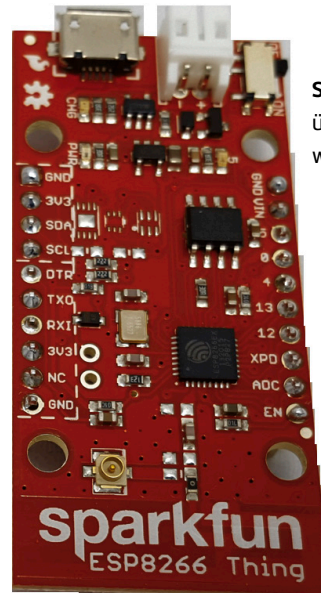


Externen Board-Manager für die ESP8266 konfigurieren (Bild 12)

Neben Lua soll nun noch gezeigt werden, wie der ESP8266 mit der Arduino-IDE programmiert werden kann. Hier wird ebenfalls auf ein bereits fertiges Board mit entsprechender Firmware zurückgegriffen, in diesem Fall das SparkFun ESP8266 Thing (Bild 11).

Das SparkFun ESP8266 Thing ist ein Board, bei dem alle Pins des Mikrocontrollers herausgeführt sind. Daneben hat es eine Micro-USB-Buchse und einen Anschluss für eine LiPo-Batterie. Über einen Schalter kann das Board an- und abgeschaltet werden. Eine rote LED signalisiert, dass das Board angeschaltet ist. Nach Anlöten von zwei Pinleisten kann das Board direkt auf ein Steckbrett aufgesteckt werden. Für die Programmierung wird ein 3,3-V-TTL-USB-Wandler benötigt.

Der ESP8266 hat bereits einen Arduino-Bootloader integriert und kann somit über die Arduino IDE programmiert werden. Hierfür steht eine Erweiterung für



SparkFun ESP8266 Thing kann über USB mit Strom versorgt werden (Bild 11)

die Arduino IDE zur Verfügung. Diese muss über den Board-Manager installiert werden. Für die Installation der ESP8266-Erweiterung muss zunächst ein zusätzlicher Board-Manager installiert werden. Hierfür öffnen Sie die Einstellungen über *Arduino* und *Einstellungen...* (Bild 12) und tragen unter *Zusätzliche Boardverwalter-URLs* den URL

http://arduino.esp8266.com/stable/package_esp8266com_index.json ein und beenden den Dialog mittels *Ok*.

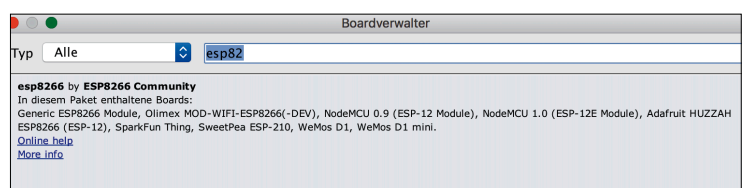
Nun kann die Erweiterung über den Board-Manager installiert werden. Diesen öffnen Sie über die Menüpunkte *Werkzeuge*, *Board* und *Boardverwalter...* (Bild 13).

Über das Suchfeld kann die Anzeige der Boards eingeschränkt werden. Wählen Sie *esp8266* von der *ESP8266 Community* aus und klicken Sie auf *Installieren*, die Version belassen Sie dabei bei 2.0.0. Nach der erfolgreichen Installation sind die unterstützten ESP8266-Boards in der Kategorie *ESP8266* unter den Menüpunkten *Werkzeuge* und *Board* zu finden (Bild 14).

Aus den zur Verfügung stehenden Boards wird in diesem Fall *SparkFun ESP8266 Thing* ausgewählt. Nun muss nur noch der USB-Port unter *Port* ausgewählt werden, und das Board kann programmiert werden. Beim SparkFun Thing ist es wichtig, dass dieses über den auf dem Board vorhandenen Schalter angeschaltet ist. Nach dem Anschalten leuchtet die rote Power-LED. Ein erster Sketch soll die Funktionsweise der Programmierumgebung testen, hierfür wird der klassische Blink-Sketch eingesetzt (Listing 2). Nach der Programmierung blinkt die mit 5 bezeichnete LED auf dem Board.

ESP8266 als Webserver

Das ESP8266 kann nicht nur mit dem Internet interagieren, sondern auch als Access Point (AP) dienen. Darüber können sich andere Geräte mit dem Modul verbinden. Als kleiner Test soll die LED an Pin 5 per URL aktiviert und deaktiviert werden. Hierfür wird auf dem Mikrocontroller ein Webserver gestartet. Über einen REST-URL wird die LED an- und aus-



Per Board-Manager können Sie zusätzliche Boards installieren (Bild 13)

ESP8266 Modules

Generic ESP8266 Module
 Adafruit HUZZAH ESP8266
 NodeMCU 0.9 (ESP-12 Module)
 NodeMCU 1.0 (ESP-12E Module)
 Olimex MOD-WIFI-ESP8266(-DEV)
 SparkFun ESP8266 Thing
 SweetPea ESP-210
 WeMos D1
 WeMos D1 mini

Nach der Installation kann das genutzte ESP8266-Board im Menü ausgewählt werden (Bild 14)

geschaltet. Der HTTP-Server wird auf Port 80 gestartet und der Access-Point wird unter dem Namen *ESP8266 test* und Passwort *webundmobile* gestartet (Listing 3). Um auf den Webserver zugreifen zu können, muss zuerst eine Verbindung mit dem gerade erzeugten WLAN *ESP8266 test* hergestellt werden. Nun kann die Webseite im Browser aufgerufen werden.

Über die IP-Adresse 192.168.4.1 kann auf das Modul zugegriffen werden. In Listing 4 sind zwei Aktionen definiert. Über */led/1* wird die LED an Pin 5 eingeschaltet und über */led/0* wird die LED ausgeschaltet. Debug-Ausgaben erfolgen über die Funktion *Serial.println()* auf die serielle Konsole.

Daten in die Cloud senden

Mit dem Sparkfun Thing können nicht nur Daten empfangen werden, sondern auch an Cloud-Plattformen gesendet werden. Sparkfun bietet hier einen freien Dienst zur Datenablage an. Dieser Dienst nutzt als Basis den Open-Source-Server Phant und ist erreichbar über <http://data.sparkfun.com>. Bei der Nutzung des Dienstes fallen derzeit keine Kosten an und es ist keine Anmeldung erforderlich. Um den Dienst in der Arduino IDE zu nutzen, muss zunächst die Phant-Biblio-

Was ist Lua?

Die in ANSI-C implementierte imperative Skriptsprache Lua wurde im Jahr 1993 veröffentlicht.

Sie eignet sich aufgrund der geringen Binärgröße des Interpreters hervorragend zum Einbinden in andere Programme, unter anderem wird Lua auch in Minecraft verwendet. Lua unterstützt funktionale und objektorientierte Programmierung. Inzwischen liegt Lua in der Version 5.3.2 vor und steht unter der MIT-Lizenz. Die geringe Binärgröße des Interpreters prädestiniert Lua für den Einsatz auf einem Mikrocontroller, denn hier wird Speicher schnell knapp. Die mit Lua erstellten Programme sind plattformunabhängig und werden vor der Ausführung meistens in einen Bytecode übersetzt. Für Kenner einer anderen imperativen Programmiersprache, wie zum Beispiel C, ist das Erlernen von Lua relativ einfach. Wichtig zu wissen: Kommentare werden mit doppeltem Bindestrich eingeleitet.

Listing 3: Access-Point und HTTP-Server starten

```

const int LED_PIN = 5;
WiFiServer server(80);
void setup() {
    Serial.begin(115200);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
    WiFi.mode(WIFI_AP);
    WiFi.softAP("ESP8266 test", "webundmobile");
    server.begin();
}

```

Listing 4: LED über URL an- und abschalten

```

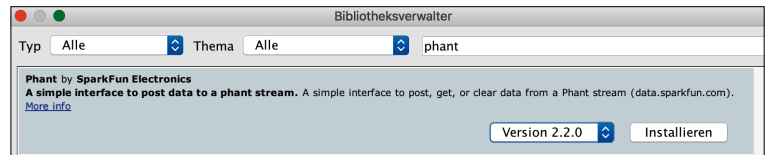
void loop() {
    WiFiClient client = server.available();
    if (!client) {
        return;
    }
    //Request einlesen
    String req = client.readStringUntil('\r');
    Serial.println(req);
    client.flush();
    int val = -1;
    if (req.indexOf("/led/0") != -1)
        val = 0; // LED soll ausgeschaltet werden
    else if (req.indexOf("/led/1") != -1)
        val = 1; // LED soll angeschaltet werden
    // LED anschalten
    if (val >= 0)
        digitalWrite(LED_PIN, val);
        client.flush();
    // HTML-Rueckgabe
    String s = "HTTP/1.1 200 OK\r\n";
    s += "Content-Type: text/html\r\n\r\n";
    s += "<!DOCTYPE HTML>\r\n<html>\r\n";
    if (val >= 0) {
        s += "LED ist ";
        s += (val)?"angeschaltet":"ausgeschaltet";
    }
    else {
        s += "Ungueltige Anfrage.<br>";
        s += "Nutzen Sie /led/1, /led/0";
    }
    s += "</html>\r\n";
    // Antwort senden
    client.print(s);
    delay(1);
    Serial.println("Verbindung geschlossen");
}

```

thek installiert werden. Hier öffnen Sie den Bibliotheksmanager über *Sketch*, *Bibliothek einbinden* und *Bibliotheken verwalten* Suchen Sie dann im Suchfeld nach *phant* und wählen Sie die aktuellste Version aus (Bild 15). Nach der Installation steht neben dem Bibliotheksnamen im Bibliotheksmanager das Wort *INSTALLED*, und damit kann Phant genutzt werden.

Bei Phant werden Daten an einen Stream gesendet. Es gibt bereits einen öffentlich eingerichteten Stream, der genutzt werden kann. Für diesen Test soll aber ein eigener Stream erzeugt werden. Hierfür klicken Sie auf das +-Symbol neben *DATA.SPARKFUN.COM*. Anschließend kann der Stream konfiguriert werden. Es kann ein Name festgelegt werden und auch Datenfelder.

Nach Abschluss der Konfiguration wird der Stream über die Schaltfläche *Save* erzeugt. Das Ergebnis wird in einer



Auswahl der phant-Bibliothek in der Arduino IDE (Bild 15)

Übersicht dargestellt. Wichtig sind die dort angezeigten Schlüssel, diese benötigen Sie für eine Arbeit mit dem Stream. Mit der installierten Bibliothek und den erzeugten Schlüsseln kann dann auf den Stream zugegriffen werden (Listing 5). Über den erzeugten URL können die gesendeten Daten im Browser betrachtet werden.

Durch die große Anzahl an Firmwares ist der ESP8266 vielseitig einsetzbar und auch für Einsteiger nutzbar. Besonders die Arduino-Firmware macht es dem Entwickler sehr einfach.

Listing 5: Daten an Phant übermitteln

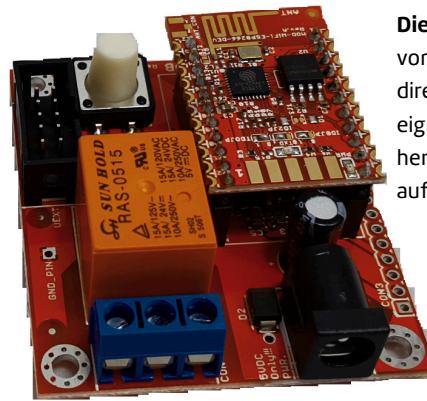
```
#include <ESP8266WiFi.h>
#include <Phant.h>
const char WiFiSSID[] = "SSID";
const char WiFiPSK[] = "PASSWORT";
const int LED_PIN = 5; // Thing's onboard, green LED
const int ANALOG_PIN = A0;
// The only analog pin on the Thing
const int DIGITAL_PIN = 12; // Digital pin to be read
const char PhantHost[] = "data.sparkfun.com";
const char PublicKey[] = "wpvZ9pE1qbFJAjaGd3bn";
const char PrivateKey[] = "wzeB1z0xWnt1YJX27xdg";
const unsigned long postRate = 30000;
unsigned long lastPost = 0;
//Verbindung mit dem WLAN
void connectWiFi() {
    byte ledStatus = LOW;
    WiFi.mode(WIFI_STA);
    WiFi.begin(WiFiSSID, WiFiPSK);
    while (WiFi.status() != WL_CONNECTED) {
        digitalWrite(LED_PIN, ledStatus);
        ledStatus = (ledStatus == HIGH) ? LOW : HIGH;
        delay(100);
    }
}
//Initialisierung der Hardware
void initHardware() {
    Serial.begin(9600);
    pinMode(DIGITAL_PIN, INPUT_PULLUP);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
}
//Daten an Phant senden
int postToPhant() {
    digitalWrite(LED_PIN, HIGH);
    Phant phant(PhantHost, PublicKey, PrivateKey);

    String postedID = "webmobile_Thing";
    phant.add("id", postedID);
    phant.add("analog", analogRead(ANALOG_PIN));
    phant.add("digital", digitalRead(DIGITAL_PIN));
    phant.add("time", millis());
    WiFiClient client;
    const int httpPort = 80;
    if (!client.connect(PhantHost, httpPort)) {
        return 0;
    }
    // Daten senden
    client.print(phant.post());
    //Ausgabe auf die serielle Konsole senden
    while(client.available()) {
        String line = client.readStringUntil('\r');
    }
    // LED wieder ausschalten
    digitalWrite(LED_PIN, LOW);
    return 1
}
void loop() {
    if (lastPost + postRate <= millis()) {
        if (postToPhant())
            lastPost = millis();
        else
            delay(100);
    }
}
// Konfiguration
void setup() {
    initHardware();
    connectWiFi();
    digitalWrite(LED_PIN, HIGH);
}
```


Und mit einem Breakoutboard entgeht man auch dem Aufspielen der Firmware. Die beiden getesteten Boards funktionieren gut und eignen sich ausgezeichnet für den Einstieg in die ESP8266-Programmierung.

Links zum Thema

- Arduino
<https://www.arduino.cc>
- BASIC Firmware für ESP8266
www.esp8266basic.com
- CoolTerm
<http://freeware.the-meiers.org>
- CP210x-USB-Treiber von Silicon Labs
<https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>
- eLua
www.eluaproject.net
- ESP8266 Flash Tool
<http://bbs.espressif.com/viewtopic.php?f=5&t=433>
- ESPlorer
<https://github.com/4refr0nt/ESPlorer>
- Expressif Systems
<http://espressif.com>
- Fritzing
<http://fritzing.org>
- IoT-Datencloud von SparkFun
<https://data.sparkfun.com>
- Lua
www.lua.org
- Micropython
<https://micropython.org>
- NodeMCU
http://nodemcu.com/index_en.html
- phant
<https://github.com/sparkfun/phant>
- Raspberry Pi
<https://www.raspberrypi.org>
- RealTerm
<http://realterm.sourceforge.net>
- SparkFun Electronics
<https://www.sparkfun.com>
- Smart.js
<https://www.cesanta.com/developer/smartjs>
- Tensilica
<http://ip.cadence.com/ipportfolio/tensilica-ip>
- ThingSpeak
<https://thingspeak.com>



Die ESP8266-Platine von Olimex kommt direkt mit Relais und eignet sich somit hervorragend für Schaltaufgaben (Bild 16)

Wer sich mit Lua anfreunden kann, sollte sich das NodeMCU DevKit näher ansehen, denn hier spart man sich den externen USB-TTL-Wandler. Neben den beiden vorgestellten Boards gibt es auch noch zahlreiche andere ESP8266-Boards. Exemplarisch sei noch das Board von Olimex genannt. Dieses kommt bereits mit einem Relais (Bild 16).

Inzwischen hat Expressif Systems die nächste Generation des ESP8266 unter dem Namen ESP32 angekündigt. Dieser Mikrocontroller vereint Bluetooth und WLAN auf einem Chip und soll weniger Strom benötigen als der ESP8266. Ob ESP32 oder ESP8266, es lohnt sich auf jeden Fall, sich mit diesem Mikrocontroller auseinanderzusetzen. Billiger wird man das eigene Projekt nicht mit dem Internet verbinden können. Durch die vorhandenen GPIO-Ports kommt man für kleinere Projekte auch ganz ohne zweiten Mikrocontroller aus. ■

Bauteil zu Fritzing hinzufügen

Für die Dokumentation von eigenen Schaltungen eignet sich sehr gut das Programm Fritzing.

Neben der Einfachheit spricht besonders die große Bauteilbibliothek für die Nutzung des Programms. In der aktuellen Version 0.9.2b sind zwar viele Bauteile enthalten, bei Weitem aber nicht alle Mikrocontrollerboards. Die Bauteilbibliothek lässt sich aber erweitern. Neue Bauteile werden im fzpz-Format importiert und stehen dann in der Bibliothek über die Kategorie *Mine* zur Verfügung. Für die beiden verwendeten Boards, SparkFun Thing und NodeMCU, steht auf GitHub bereits ein fertig gezeichnetes Bauteil zum Download bereit.



Dr. Markus Stäuble ist Informatiker, Conference Chair der Developer Week, Fachautor und Programmleiter Make beim Franzis Verlag. Neben Programmiersprachen und Make beschäftigt er sich mit dem Thema Mobile und hat zu dessen Auswirkungen auf die Arbeitswelt promoviert.

ONLINE-BEILEGUNG VERBRAUCHERRECHTLICHER STREITIGKEITEN

Gepflegt streiten

Die EU macht Vorgaben zur Streitbeilegung zwischen Unternehmen und Verbrauchern.

Seit dem 9. Januar 2016 bestehen neue Handlungs- bzw. Informationspflichten für alle Webshop-Betreiber. Ausgangspunkt hierfür ist die EU-Verordnung Nr. 524/2013 des Europäischen Parlaments und des Rates vom 21. Mai 2013 über die Online-Beilegung verbraucherrechtlicher Streitigkeiten und zur Änderung der EG-Verordnung Nr. 2006/2004 und der Richtlinie 2009/22/EG, oder kurz: Verordnung über Online-Streitbeilegung in Verbraucherangelegenheiten (ODR-Verordnung).

Deren Art. 14 Abs. 1 besagt Folgendes: »In der Union niedergelassene Unternehmer, die Online-Kaufverträge oder Online-Dienstleistungsverträge eingehen, und in der Union niedergelassene Online-Marktplätze stellen auf ihren Websites einen Link zur OS-Plattform ein. Dieser Link muss für Verbraucher leicht zugänglich sein. In der Union niedergelassene Unternehmer, die Online-Kaufverträge oder Online-Dienstleistungsverträge eingehen, geben zudem ihre E-Mail-Adressen an.«

Diese neuen Informationspflichten sind primär dazu geschaffen worden, um Verbrauchern bei grenzüberschreitenden Internet-Verträgen auf die Möglichkeit für kostengünstige, außergerichtliche Verfahren zur Streitbeilegung hinzuweisen. Zu diesem Zweck betreibt die EU-Kommission die



Betroffene Händler finden auf diesem Blog weitergehende Informationen und aktuelle News (**Bild 1**)

sogenannte OS-Plattform als zentrale Anlaufstelle für Verbraucher. Darüber werden dann alle Beschwerden an die für die jeweilige Angelegenheit zuständige alternative Streitbeilegungsstelle (AS-Stelle) weitergereicht. Die Pflicht zur Platzierung eines solchen Hinweises inklusive Verlinkung stellt eine wichtige Ergänzung zu nationalen Informationspflichten über alternative Streitbeilegungsverfahren dar.

Gesetztestexte

Die nachfolgenden Zitate geben die entsprechenden Gesetzestexte im Wortlaut wieder.

§ 36 VSBG lautet:

Allgemeine Informationspflicht

(1) Ein Unternehmer, der eine Webseite unterhält oder Allgemeine Geschäftsbedingungen verwendet, hat den Verbraucher leicht zugänglich, klar und verständlich

1. in Kenntnis zu setzen davon, inwieweit er bereit ist oder verpflichtet ist, an Streitbeilegungsverfahren vor einer Verbraucherschlichtungsstelle teilzunehmen, und
2. auf die zuständige Verbraucherschlichtungsstelle hinzuweisen, wenn sich der Unternehmer zur Teilnahme an einem Streitbeilegungsverfahren vor einer Verbraucherschlichtungsstelle verpflichtet hat oder wenn er auf Grund von Rechtsvorschriften zur Teilnahme verpflichtet ist; der Hinweis muss Angaben zu Anschrift und Webseite der Verbraucherschlichtungsstelle sowie eine Erklärung des Unternehmers, an einem Streitbeilegungsverfahren vor dieser Verbraucherschlichtungsstelle teilzunehmen, enthalten.

(2) Die Informationen nach Absatz 1 müssen

1. auf der Webseite des Unternehmers erscheinen, wenn der Unternehmer eine Webseite unterhält,
2. zusammen mit seinen Allgemeinen Geschäftsbedingungen gegeben werden, wenn der Unternehmer Allgemeine Geschäftsbedingungen verwendet.
- (3) Von der Informationspflicht nach Absatz 1 Nummer 1 ausgenommen ist ein Unternehmer, der am 31. Dezember des vorangegangenen Jahres zehn oder weniger Personen beschäftigt hat.

§ 37 VSBG lautet (voraussichtlich):

Informationen nach Entstehen der Streitigkeit

- (1) Der Unternehmer hat den Verbraucher auf die für ihn zuständige Verbraucherschlichtungsstelle unter Angabe von deren Anschrift und Webseite hinzuweisen, wenn die Streitigkeit über einen Verbrauchervertrag durch den Unternehmer und den Verbraucher nicht beigelegt werden konnte. Der Unternehmer gibt zugleich an, ob er zur Teilnahme an einem Streitbeilegungsverfahren bei dieser Verbraucherschlichtungsstelle bereit ist oder verpflichtet ist.
- (2) Der Hinweis muss in Textform gegeben werden.

Betroffene Händler finden im Internet zum Beispiel unter www.verbraucherstreitbeilegung.de weitergehende Informationen und ständig aktualisierte News (Bild 1).

Hintergrund

Die neuen Pflichten treffen nicht nur deutsche, sondern alle Webshop-Betreiber, die ihren Sitz in einem Mitgliedstaat der Europäischen Union haben. Es kommt nicht darauf an, ob der einzelne Händler an einem solchen Streitschlichtungsverfahren teilnehmen will oder ob er überhaupt davon gehört hat. Die Hinweispflicht trifft ausnahmslos jeden Online-Händler.

Einzig das reine B2B-Umfeld ist ausgenommen, denn die neuen Info-Pflichten treffen lediglich die Rechtsbeziehungen von Unternehmen zu Verbrauchern, also den B2C-Sektor. Jeder Online-Händler, der zumindest auch Verträge mit Verbrauchern abschließt, ist also Verpflichteter im Sinne der ODR-Verordnung. Ausgenommen sind hingegen wohl reine Unternehmenspräsenzen, über die keine Verträge über Waren beziehungsweise Dienstleistungen abgeschlossen werden können; rein private Websites sind ohnehin nicht betroffen (Bild 2).

Die exakte Formulierung des Hinweistextes wird von Seiten der EU nicht vorgegeben, in dieser Hinsicht kann und muss jeder einzelne Betroffene selbst kreativ werden. Entscheidend ist jedoch, dass der Hinweis für Verbraucher »leicht zugänglich« und natürlich auch als solcher erkennbar ist. Der bloße Link dürfte insoweit nicht ausreichend sein, es ist schon ein sprechender Link, also eine Erläuterung erforderlich. Hier gilt der Grundsatz: So viel wie nötig, so wenig wie möglich. Der Hinweistext kann also beispielsweise wie folgt formuliert werden:

»Informationen zur Online-Streitbeilegung: Die EU-Kommission stellt eine Internetplattform zur Online-Beilegung von Streitigkeiten (sogenannte OS-Plattform) bereit. Diese OS-Plattform soll als Anlaufstelle zur außergerichtlichen Beilegung von Streitigkeiten im Zusammenhang mit vertraglichen Verpflichtungen aus Online-Kaufverträgen dienen. Die OS-Plattform ist unter folgendem Link erreichbar: www.ec.europa.eu/consumers/odr. Unser E-Mail-Adresse lautet wie folgt: xyz@mustershop.de. Alternativ kann auch ein Verweis auf die ohnehin im Impressum bereits genannte E-Mail-Adresse erfolgen (»Unsere E-Mail-Adresse finden Sie oben bei unseren Kontaktdaten.«)

Ein Hinweistext in dieser Fassung kann natürlich nur dann gewählt werden, wenn ausreichend Platz für eine solche Zeichenanzahl vorhanden ist beziehungsweise wenn eine Möglichkeit zur Verlinkung von externen Websites besteht (siehe dazu auch unseren Praxis-Hinweis).

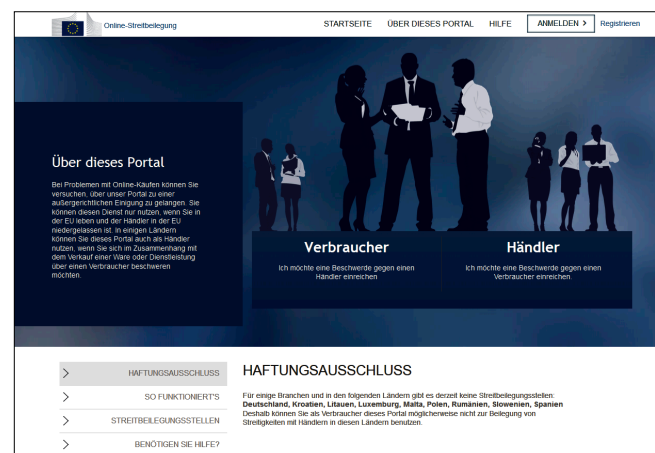
Platzierung

Was bedeutet »leicht zugänglich« nun genau? Wo kann ein von der EU-Regelung betroffener Online-Händler den Hinweis nebst Verlinkung platzieren, um dieser Vorgabe Folge zu leisten?

Für die Lösung dieses Problems gibt es gleich mehrere Möglichkeiten. In erster Linie sticht natürlich eine Platzierung im Impressum ins Auge, da sich hier auch noch andere

juristisch notwendige Angaben finden. Auf Handelsplattformen, wie etwa Ebay oder Amazon, kann der Hinweis beispielsweise in den Abschnitt »Rechtliche Informationen des Verkäufers« beziehungsweise »Verkäuferinformationen -> Impressum« aufgenommen werden.

Auch die Darstellung als eigenständige Klausel (Streitschlichtung) in den eigenen Allgemeinen Geschäftsbedingungen (AGB) ist sicherlich möglich. Die Einrichtung eines entsprechenden, separaten Menüpunkts oder die Einbindung des Hinweises in einen Menüpunkt (zum Beispiel Kundeninformationen), über den noch andere Hinweise für Verbraucher erreichbar sind (zum Beispiel Angaben nach dem Batteriegesetz oder gemäß Altöl-Verordnung et cetera), ist ebenfalls denkbar.



Auf dieser Plattform soll künftig die Streitbeilegung zwischen Unternehmen und Verbrauchern stattfinden (Bild 2)

Zusätzlich kann auch ein Hinweis im Rahmen der Produkteinzelseiten erfolgen. Wird der Hinweis allerdings nur im Produkt- beziehungsweise Auktionsbeschreibungstext eingebunden und nicht zusätzlich zum Beispiel auch im Impressum, dann besteht die Gefahr, dass diejenigen Besucher der Website, die sich keine konkreten Produkte anschauen, den Hinweis nicht bemerken. Daher kann dies sicherlich nur als zusätzliche Option angesehen werden. Dies gilt im Grunde auch für die Einbindung von Hinweis plus Verlinkung in die E-Mails, welche als Bestätigung über den Eingang der Bestellung beziehungsweise als Annahme des Kaufvertrags an Kunden versandt werden. Allerdings ergibt sich hier die gleiche Problematik wie bei den Produktbeschreibungen.

Unter dem Strich ist daher die Einbindung von Hinweis und Link im Impressum und/oder in den AGB zu empfehlen.

Nationale Regelungen

Am 29. Januar 2016 hat der Deutsche Bundesrat das Gesetz zur Umsetzung der Richtlinie über alternative Streitbeilegung in Verbraucherangelegenheiten und zur Durchführung der Verordnung über Online-Streitbeilegung in Verbraucherangelegenheiten, oder kurz: Verbraucherstreitbeilegungsgesetz (VSBG), passieren lassen. Wenn das VSBG auch vom ►

Bundespräsidenten abgesegnet wird, soll es voraussichtlich ab 1. April 2016 in Kraft treten. Das bedeutet, dass ab diesem Zeitpunkt erweiterte Informationspflichten gelten und der Hinweis gemäß ODR-Verordnung nicht mehr ausreichend sein wird – jedenfalls nicht für deutsche Online-Händler.

Intention der EU

Hintergrund für den Beschluss des VSBG durch den deutschen Gesetzgeber ist die Richtlinie 2013/11/EU des Europäischen Parlaments und des Rates vom 21. Mai 2013 über die alternative Beilegung verbraucherrechtlicher Streitigkeiten und zur Änderung der Verordnung (EG) Nr. 2006/2004 und der Richtlinie 2009/22/EG, oder kurz: Richtlinie über alternative Streitbeilegung in Verbraucherangelegenheiten (ADR-Richtlinie). Diese Richtlinie hätte bis zum 9. Juli 2015 in allen EU-Mitgliedstaaten in das jeweilige nationale Recht umgesetzt werden müssen – »hätte« deswegen, weil dies hierzulande bislang noch nicht abschließend geschehen ist. Erst mit dem Inkrafttreten des VSBG wird die Umsetzung endgültig realisiert sein.

Stichtage

Für die Umsetzung der Vorgaben gemäß ODR-Verordnung ist in dieser keinerlei Übergangsfrist vorgesehen, sodass alle Verpflichtungen ausnahmslos seit dem 9. Januar 2016 umgesetzt sein müssen. Dies gilt auch vor dem Hintergrund, dass die OS-Plattform erst ab dem 15. Februar 2016 freigeschaltet worden ist – zuvor existierte lediglich eine Hinweisseite.

Aus dem VSBG werden sich voraussichtlich ab dem April 2017 weitere Informationspflichten für deutsche Online-Händler ergeben. Wie genau diese aussehen werden beziehungsweise wann und in welcher Form sie umzusetzen sind, ist derzeit noch nicht absehbar. Aller Voraussicht nach wird jedoch – einmal mehr – eine nicht unerhebliche Anpassung der AGB beziehungsweise der Website erforderlich sein. So werden Online-Händler ihre (privaten) Kunden insbesondere über ihre Verpflichtung zur Teilnahme an Streitbeilegungsverfahren oder auch über die zuständige AS-Stelle samt deren Anschrift und Internetadresse informieren müssen.

Aufgrund der Tatsache, dass es sich bei den Vorgaben der Richtlinie über alternative Streitbeilegung und auch bei der ODR-Verordnung um zwingendes Verbraucherschutzrecht handelt, dürfte dessen ungenügende oder gar gänzlich feh-

Praxis-Tipp

In speziellen Fällen oder aus technischen Gründen kann eine verkürzte Muster-Formulierung zum Einsatz kommen.

In manchen Fällen, zum Beispiel auf Plattformen wie Ebay oder Amazon, lässt sich ein ausführlicher Hinweis nebst Link auf die OS-Plattform aus technischen Gründen nicht unterbringen, etwa weil es eine Begrenzung der verfügbaren Zeichenanzahl gibt. In solchen Fällen kann folgende, etwas verkürzte Muster-Formulierung zum Einsatz kommen:

Plattform der EU-Kommission zur Online-Streitbeilegung (OS-Plattform): www.ec.europa.eu/consumers/odr.

Auch hierbei ist, falls möglich und zulässig, der angegebene Link zur OS-Plattform anklickbar zu machen, das heißt, mit dem passenden URL zu verknüpfen. Zudem muss auch die E-Mail-Adresse angegeben werden, zum Beispiel im Rahmen der Kontakt- beziehungsweise Impressums-Angaben, also nicht zwingend im Rahmen des Hinweises auf die OS-Plattform.

lende Umsetzung als Verstoß gegen das Wettbewerbsrecht zu werten sein. Als solcher kann er von Mitbewerbern oder auch von bestimmten Verbraucherschutzorganisationen gemäß § 3a des Gesetzes gegen den unlauteren Wettbewerb (UWG) abgemahnt werden. Zudem besteht für deutsche Händler die Gefahr, neben einer kostenpflichtigen Abmahnung auch noch ein Bußgeld zu kassieren. Denn Art. 18 ODR-Verordnung regelt, dass jeder einzelne Mitgliedstaat selbst festlegen kann, ob und welche Sanktionen er bei einem Verstoß gegen diese Verordnung verhängt.

Kosten

Wichtig zu erwähnen ist abschließend noch, dass zwar eine Informationspflicht, aber keine Pflicht zur Teilnahme an einem Streitschlichtungsverfahren besteht. Erfolgt also eine Beschwerde über die OS-Plattform, müssen Händler darauf nicht reagieren. Sofern es jedoch eine freiwillige Teilnahme seitens des Händlers gibt, so hat dieser auch die anfallenden Kosten dafür zu tragen – wie hoch diese sein werden, kann zum jetzigen Zeitpunkt leider noch nicht gesagt werden. Für Verbraucher wird ein solches Verfahren jedoch definitiv kostenfrei sein. ■

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>
- Weitergehende Informationen zum Thema E-Commerce
<http://rechtssicher.info>



Michael Rohrlisch

ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.

www.rechtssicher.info

Impressum

Verlag

Neue Mediengesellschaft Ulm mbH
Bayerstraße 16a,
80335 München
Telefon: (089) 741 17-0,
Fax: (089) 741 17-101
(ist zugleich Anschrift aller
Verantwortlichen)

Herausgeber

Dr. Günter Götz

Chefredakteur

Max Bold
– verantwortlich für den
redaktionellen Teil –
E-Mail: redaktion@webundmobile.de

Schlussredaktion

Ernst Altmannshofer

Redaktionelle Mitarbeit

Philip Ackermann, Christian Bleske,
Jens Geyer, Thomas Hafen,
Tam Hanna, Christoph Hoffmann,
Bernhard Lauer, Scott Love,
Florence Maurice, Michael Rohlich,
Jochen Schmidt, Markus Schraudolph,
Katharina Sckommodau,
Thomas Sillmann, Frank Simon,
Markus Stäuble, Bettina Zimmermann

Art Directorin

Maria-Luise Sailer

Grafik & Bildredaktion

Alfred Agatz, Dagmar Breitenbach,
Catharina Burmester, Hedi Hefele,
Manuela Keller, Simone Köhnke,
Cornelia Pflanzner, Petra Reichensperner,
Ilka Rüther, Christian Schumacher,
Nicole Üblacker, Mathias Vietmeier

Anzeigenberatung

Jens Schmidtmann, Anzeigenleiter
Klaus Ahlering, Senior Sales Manager
Telefon: (089) 741 17-125
Fax: (089) 741 17-269
E-Mail Anzeigenberatung:
sales@nmg.de

Anzeigendisposition

Dr. Jürgen Bossmann
Telefon: (089) 741 17-281
Fax: (089) 741 17-269
E-Mail: sales@nmg.de

Leitung Herstellung/Vertrieb

Thomas Heydn
Telefon: (089) 741 17-111
E-Mail: thomas.heydn@nmg.de

Leserservice

Hotline: (089) 741 17-205
Fax: (089) 741 17-101
E-Mail: leserservice@nmg.de

Kooperationen

Denis Motzko
Telefon: (089) 741 17-116
E-Mail: kooperationen@nmg.de

Druck

L.N. Schaffrath Druckmedien
Marktweg 42-50
47608 Geldern

DVD/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-Entwick-
ler 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 kos-
tet 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 Pro-
zent.

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

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Monatliches Ranking

Fischen in den sozialen Medien

Glaubt man den Aussagen des ICR Recruiting Reports 2015, so müssen sich die meisten Entwickler heute gar nicht mehr darum bemühen, einen neuen Job zu finden. Es genügt, sich in den sozialen Medien möglichst gut zu präsentieren, der Rest geht dann wie von selbst.

Der Report sagt nämlich, dass Software-Entwickler immer seltener über Print- oder Online-Anzeigen gefunden werden und die Unternehmen stattdessen dazu übergegangen sind, selbstständig geeignete Kandidaten aus den sozialen Medien zu fischen und direkt anzusprechen. Active Sourcing heißt diese Methode der Mitarbeiterfindung. Gemäß ICR Recruiting Report nutzen 54 Prozent der IT Unternehmen und 76 Prozent der Internet-Unternehmen bereits Active Sourcing, um geeignete Kandidaten zu finden.

Die Suche nach Jobs für Webentwickler in der Datenbank von Jobkralle.de zeigte am Erhebungsstichtag, dass derzeit die meisten Stellen für Bayern, Baden-Württemberg, NRW und Hessen ausgeschrieben sind. Im Ranking der deutschen Großstädte liegt München (2326 Treffer) deutlich vor Berlin, Hamburg und Frankfurt. Auf dem fünften Platz folgt dann Stuttgart vor Köln und Düsseldorf (Bild 1). Vergleichsweise wenige Webentwickler-Angebote sind in Bremen, Essen und Duisburg zu finden.

Jobs für Mobile-Entwickler

Bei der Verteilung der Jobangebote für Entwickler von Apps für Mobilgeräte auf die Bundesländer sieht es nicht viel anders aus als beim Jobangebot für Webentwickler. Ganz oben stehen Bayern (2407 Treffer), Baden-Württemberg und NRW. Vor Hessen kommen hier aber noch Berlin und Hamburg (Bild 2). Kaum Angebote für Mobile-Entwickler gibt es in Thüringen,

Sachsen-Anhalt, Schleswig-Holstein, Bremen, Saarland und Mecklenburg-Vorpommern. Für sie alle lieferte die Suche auf Jobkralle.de jeweils weniger als 100 Treffer.

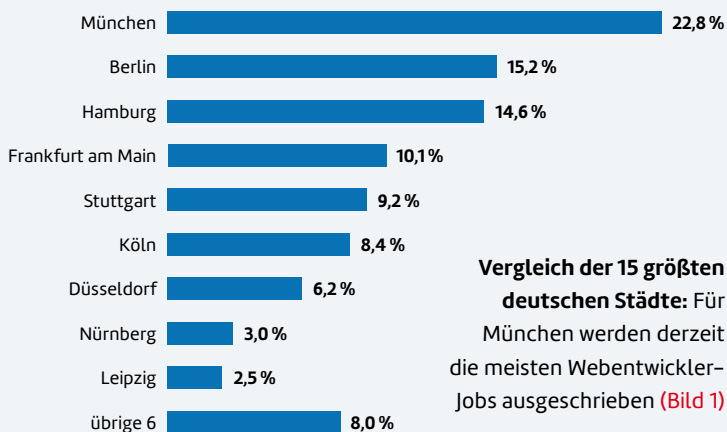
Betrachtet man die Angebote in den 15 größten deutschen Städten, so liegen München (22,2 Prozent), Berlin (16,5 Prozent) und Hamburg (15,7 Prozent) ganz vorne, gefolgt von Frankfurt, Stuttgart und Köln. Am Ende des Rankings stehen Bremen, Essen und Duisburg.

In Tabelle 1 finden Sie die in Stellenanzeigen geforderten Technologiekenntnisse. Dafür wird die Datenbank der Meta-Suchmaschine Jobkralle.de nach ausgewählten Stichworten abgefragt. In den Stellenanzeigen am häufigsten genannt wird derzeit das Stichwort Cloud (über 6300 Treffer). Die am häufigsten genannte Smartphone-Technologie ist Android. Die Suche nach Windows 10 liefert 1200 Treffer. Das sind rund zehn Prozent aller Nennungen von Windows.

Tabelle 1: Technologien

Rang	Technologie	Anteil
1	Cloud	17,4 %
2	MySQL	12,3 %
3	HTML5	10,3 %
4	SharePoint	7,1 %
5	Big Data	6,8 %
6	Microsoft SQL Server	6,4 %
7	Android	6,3 %
8	Responsive Web	5,9 %
9	iOS	5,4 %
10	Angular.js	5,0 %
11	CSS3	4,8 %
12	Windows 10	3,3 %
13	WPF	2,9 %
14	NoSQL	2,4 %
15	Azure	1,9 %
16	WCF	1,8 %

Jobs für Webentwickler

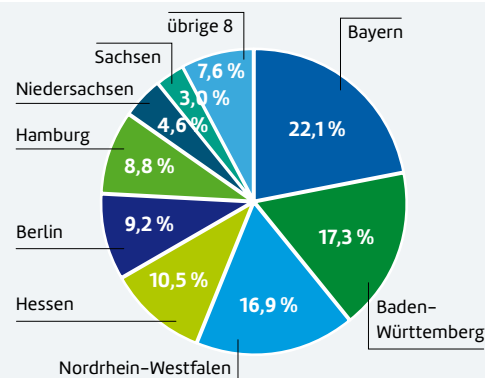


Vergleich der 15 größten deutschen Städte: Für München werden derzeit die meisten Webentwickler-Jobs ausgeschrieben (Bild 1)

web & mobile developer 5/2016

Quelle: Jobkralle.de

Jobs für Mobile-Entwickler



Hochburgen für Mobile-Entwickler liegen in Bayern, Baden-Württemberg und NRW (Bild 2)

web & mobile developer 5/2016

Quelle: Jobkralle.de

Zahl des Monats

Der StepStone-Gehaltsreport nennt **52.000 Euro** als das Durchschnittsgehalt einer Fachkraft in Deutschland. Eine Aufgabe mit Personalverantwortung bringt in der IT im Schnitt **10 Prozent** mehr Gehalt.

Quelle: StepStone-Gehaltsreport

Dice Job Market Report

Wo man die Techies findet

Arbeitgeber, die IT-Fachpersonal suchen, sind nicht zu befechten. Der IT-Arbeitsmarkt ist heiß umkämpft. Arbeitgeber reagieren entsprechend: 58 Prozent der Festangestellten im IT-Bereich haben bei der letzten Gehaltsrunde mehr Geld bekommen. Doch das reicht im aktuellen Markt offenbar nicht aus: 77 Prozent erwarten laut dem gerade erschienenen Dice Job Market Report 2016 in der nächsten Gehaltsrunde eine weitere Steigerung ihrer Bezüge (Bild 3). Dice hat hierzu 320 IT- und Tech-Profis nach ihren Ansichten und Erwartungen befragt, um einen Überblick über die wichtigsten Trends auf dem IT-Arbeitsmarkt zu geben. Dabei wurde auch ermittelt, dass die Bezahlung unter den Gründen, beim aktuellen Arbeitgeber zu bleiben, nur an zweiter Stelle liegt (Bild 4). <http://svy.mk/24rbkXh>

GFN AG

Gratiszugang zu 600 Fachbüchern

Alle Teilnehmer eines der offenen Seminare des Schulungsanbieters GFN AG im Jahr 2016 erhalten zusätzlich zur Schulung einen einjährigen Gratiszugang zu 600 Fachbüchern aus den Bereichen Business, persönliche Entwicklung und IT. Der Zugriff auf die E-Books ist von überall aus sowie mit allen gängigen Geräten und Betriebssystemen möglich – mit dem Smartphone, einem Tablet-PC oder am PC oder Laptop. Den persönlichen Zugangscode erhalten Teilnehmer am ersten Seminartag.

www.gfn.de

Die entscheidende Frage

Wie würden Sie antworten?

Wer diese Frage gestellt bekommt, hat bei seiner Bewerbung bei Facebook bislang alles

richtig gemacht. Die Frage ist nämlich – so hat es Facebook-Chefin für die Personalauswahl Miranda Kalinowski dem Wirtschaftsportal Business Insider verraten – die letzte und für die Einstellung entscheidende Frage. Sie lautet: »An ihrem allerbesten Arbeitstag – dem Tag, an dem Sie nach Hause kommen und überzeugt sind, dass Sie den besten Job der Welt haben – was haben Sie an diesem Tag gemacht?« Für Facebook ist es wichtig, dass in der Antwort zum Ausdruck kommt, dass der ideale Arbeitstag im Einklang mit der Konzernmission steht.

Gut möglich, dass auch andere Arbeitgeber Gefallen an dieser Frage finden. Man sollte sie nicht nur deshalb einmal für sich beantworten. In Ruhe, ohne Bewerbungsstress. In einer ehrlichen Antwort erkennt man die Motivation, mit der man dem Alltagstrott begegnet. Bei so manchem wird die ehrliche Antwort nicht unbedingt für ein Bewerbungsgespräch taugen.

www.facebook.com

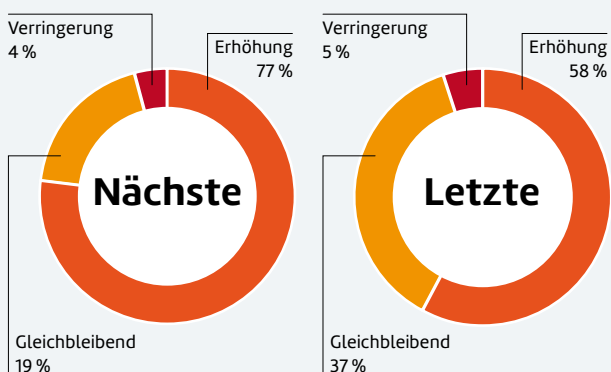
TAE

Berufsbegleitende Weiterbildung

Ab September 2016 starten wieder die berufsbegleitenden Studiengänge der Technischen Akademie Esslingen. Parallel zur Berufstätigkeit besteht die Möglichkeit, nicht nur zusätzliche Fachkenntnisse, sondern auch staatlich anerkannte Studienabschlüsse zu erwerben. Neu hinzugekommen ist das Studium der Wirtschaftsinformatik. Es setzt sich zu 60 Prozent aus Informatikthemen und zu 40 Prozent aus betriebswirtschaftlichen Inhalten zusammen. Regelmäßig wird während des Studiums der Bogen von Wissenschaft zu Praxis geschlagen. Speziell entwickeltes Lernmaterial hilft, Beruf und Studium zu vereinen. Das Wirtschaftsingenieur-Studium ist das Studium für alle, die an den Schnittstellen zwischen Technik und Wirtschaft beschäftigt sind oder dies anstreben.

www.tae.de

Gehaltserwartungen

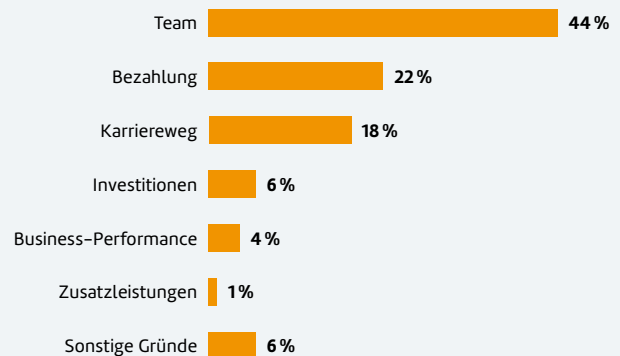


Für die nächste Gehaltsverhandlung erwarten 77 % der Studienteilnehmer eine Gehaltssteigerung (Bild 3)

web & mobile developer 5/2016

Quelle: Dice Job Market Report 2016

Gründe, zu bleiben



Das funktionierende Team ist für Arbeitnehmer der wichtigste Grund, beim aktuellen Arbeitgeber zu bleiben (Bild 4)

web & mobile developer 5/2016

Quelle: Dice Job Market Report 2016

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

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 6/2016 erscheint am 12. Mai 2016

JavaScript-Frameworks im Vergleich



Dank zahlreicher leistungsstarker Frameworks, welche die Entwicklung mobiler Apps beziehungsweise die Entwicklung von Webapplikationen unterstützen, gewinnt JavaScript laufend an Bedeutung. Die aktuellen JavaScript-Frameworks fallen in eine von zwei Kategorien: Innovatoren oder Traditionalisten. Erfahrene Entwickler möchten vorzugsweise auf ausgereifte, bewährte Technologien setzen und gleichzeitig neue Trends wie die bevorstehende groß angelegte Umstellung auf ECMAScript 6 (ES6) im Auge behalten, um sich darauf rechtzeitig vorzubereiten. Bei JavaScript ist es schwierig, angesichts der zahllosen Frameworks und verschiedener Ansätze den Überblick zu behalten. Das Schwerpunktthema der nächsten Ausgabe gibt einen Überblick über das Angebot.

Scrolleffekte mit CSS3

CSS bietet verschiedene Möglichkeiten, das Scrollverhalten einzelner Elemente zu beeinflussen – und mit CSS3 sind neue Optionen dazugekommen. Der Artikel stellt neue Features wie position: sticky, Snappoints und scrollbehavior vor und zeigt schicke Effekte in Kombination mit JavaScript. Beim horizontalen Scrollen kommt meist JavaScript zum Einsatz. Mit den Snappoints gibt es in CSS3 eine reine CSS-Alternative.

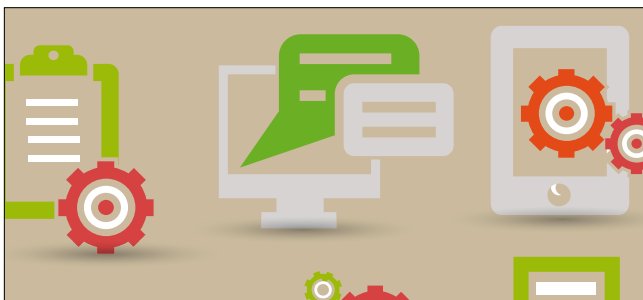
Microsoft ASP.NET 5 und Docker

Das Tool Docker ist eine Container-Technologie, die eine leichtgewichtige virtuelle Maschine darstellt und auf Prozesslevel-Ebene arbeitet. Das heißt, Docker ermöglicht es, Anwendungen in Containern auszuführen. Auch das Management von Linux-Containern wird vereinfacht. Ein Workshop zeigt, wie Sie die Docker-Erweiterungen von Visual Studio nutzen, um in Azure eine ASP.NET 5-Anwendung in einem Docker-Container zu installieren.

Der neue Raspberry Pi 3

Der Einplatinen-PC Raspberry Pi ist in der neuen Version 3 Model B erstmals mit Funkmodulen für WLAN und Bluetooth ausgerüstet. Außerdem haben die Entwickler dem Micro-PC einen flotten Quad-Core-Prozessor mit 64-Bit-Support spendiert. Dank der neuen Funkmodule unterstützt der Raspberry Pi 3 nun ohne Zubehör die WLAN-Standards 802.11 b/g/n sowie Bluetooth 4.1 mitsamt Bluetooth Low Energy.

dotnetpro



Ausgabe 5/2016 ab 21. April am Kiosk

Der Schwerpunkt der dotnetpro 4/2016 befasst sich mit der Frage, welche Tools die Favoriten von Entwicklern sind. Außerdem wird das aktuelle RAD Studio von Embarcadero vorgestellt.

www.dotnetpro.de

Unsere digitalen Angebote

 **Wöchentlicher Newsletter**
webundmobile.de/newsletter

 **Shop**
shop.webundmobile.de

 **YouTube**
youtube.com/user/developermedia

 **Facebook**
facebook.com/webundmobile

 **Google +**
gplus.to/webundmobile

 **Twitter**
twitter.com/webundmobile

Stellenmarkt

dotnetpro + web & mobile Developer

○ 25.800 Exemplare Gesamtauflage

○ 25.300 Newsletter-Empfänger

○ 66.600 PI'S



○.NET ○Architektur ○HTML5/JavaScript ○iOS/Android ○

Kontakt:

Jens Schmidtman, Klaus Ahlering • Tel. 089/74117-125 • sales@nmg.de

Ihr Partner für mehr Online-Wachstum

Wir planen, entwickeln und steuern
Websites, Apps und Kampagnen.

Unsere Ziele sind Ihre Ziele:

- ⊕ **Mehr Sichtbarkeit**
- ⊕ **Mehr Traffic**
- ⊕ **Mehr Leads**
- ⊕ **Mehr Conversions**

.....

➔ **Mehr Kunden**

Besuchen Sie uns unter
www.digitalmobil.com

