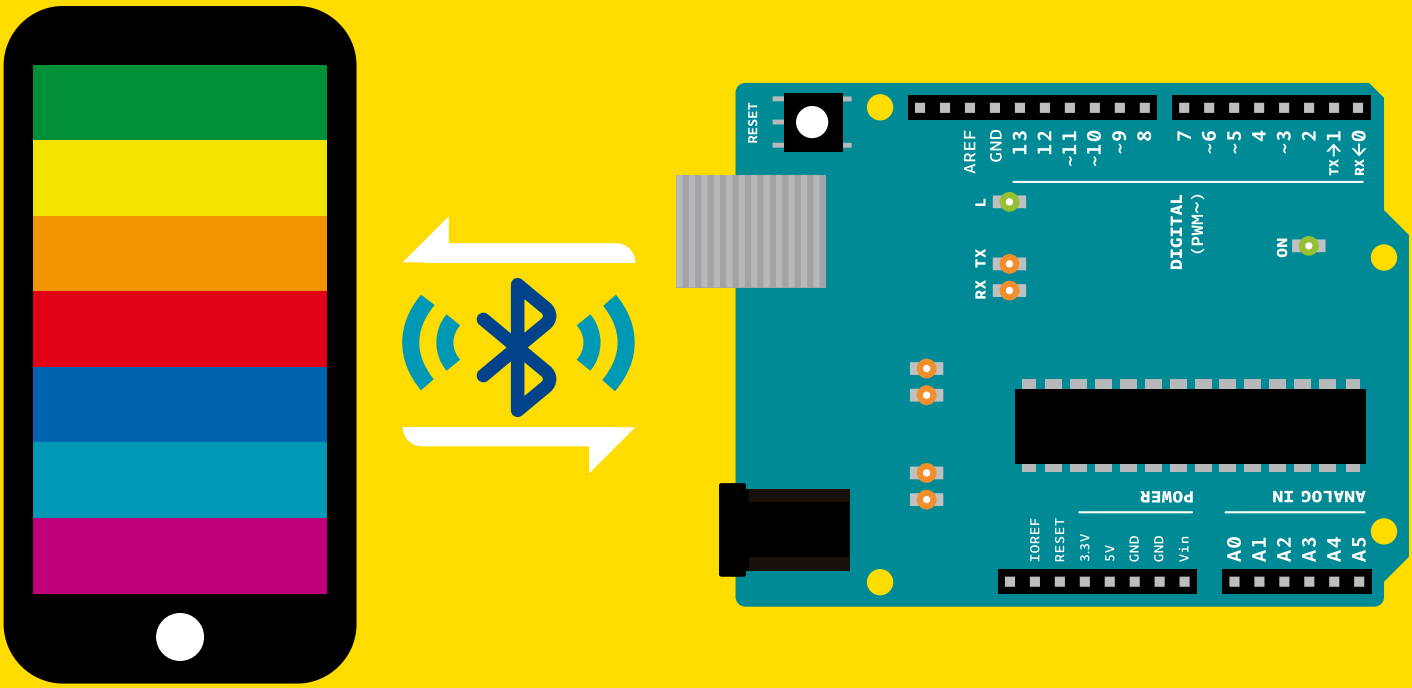
web & mobile DEVELOPER

webundmobile.de

Arduino trifft Smartphone

Arduino 101 einfach und effizient mit Android-Smartphones verbinden und IoT-Geräte ansteuern **ab S. 22**



WordPress-Formulare bauen

Markus Schraudolph zeigt, wie man in WordPress elegant Formulare mit Formidable realisiert **S. 100**

Ausgabe 10/16

Deutschland 14,95 EUR

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

4 198255 014951 10

Smart Services sorgen für nahtlose Business-Prozesse

GRATIS Whitepaper

Was Unternehmen über Daten- und Prozessintegration wissen müssen, wenn sie überleben wollen

Lesen Sie im kostenfreien Whitepaper von >eurodata

- Alles im Griff – mit innovativen Smart Services
- Hype oder Realität? – Wie das Internet der Dinge und Industrie 4.0 die IT-Welt verändern
- Was tun mit dem Internet der Dinge?
- Die Zukunft – smarte Datenintegration
- Best Practice I – Smart Services sorgt für Prozessübersichtlichkeit
- Best Practice II – Smartes Daten-Matching unterstützt reibungslose Transaktionsprozesse
- Best Practice III – Smartes Partnermanagement dank digitaler Businessprozesse
- Blick in die Kristallkugel

Jetzt kostenlos downloaden:
http://digital.com-magazin.de/eurodata-smartservices-wump/

Ein Service von: FÜR IT-ENTSCHEIDER com! professional

Ihr Whitepaper bei uns?
Anfragen an sales@nmg.de
oder Tel. 089 / 7 41 17 – 124.

Neue Techniken

Diese Ausgabe bietet jede Menge Know-how zum Thema Webdesign mit Techniken wie Flexbox oder WebSockets oder funktional reaktiver Programmierung in JavaScript.

Auch wenn andere Arduinos und Selbstlaborate mit Bluetooth-Modulen etwas billiger sind: Der Arduino 101 spart insbesondere bei Kleinserien wertvolle Zeit. Wenn die zu lösende Aufgabe in das von Bluetooth LE vorgegebene Kommunikationsschema passt, sollten Sie dem 101 eine Chance geben. Es ist in vielen Fällen unbezahlbar, wenn man sich um die Funkhardware keine Gedanken machen muss. Tam Hanna erläutert in einem Schwerpunkt ab Seite 22, wie man den beliebten Einplatinencomputer am besten mit einem Android-Smartphone verbindet, um damit IoT- oder Smart-Home-Projekte zu realisieren.

»Wie man den Arduino am effektivsten mit einem Android-Smartphone verbindet.«

Die HTML5-WebSocket-Spezifikation ermöglicht den Aufbau einer persistenten bidirektionalen Verbindung zwischen Client und Server über das WebSocket-API, was den Anforderungen heutiger Webseiten entspricht. Das WebSocket-Protokoll ist Teil des HTML5-Standards und umgeht das HTTP-Protokoll, über das der Client den Server regelmäßig anfragen muss, um Daten zu erhalten. Das WebSocket-Protokoll, dessen Ansteuerung mit JavaScript über das WebSocket-API erfolgt, ermöglicht das schnelle Senden und das sofortige Empfangen von Texten, binären Arrays oder Blobs. Die Details erläutert Philippe Bénard in einem Artikel ab Seite 40.

Wäre es nicht schön, wenn das Schreiben verteilter, skalierbarer Anwendungen einfacher wäre und man den dafür jedes Mal erforderlichen Overhead so weit wie irgend möglich einfach hinter einer schönen Abstraktion verschwinden lassen könnte? Genau das ist der Anspruch, den das in Microsoft Research beheimatete Projekt Orleans erfüllen will. Nach seiner Veröffentlichung als Open Source Anfang 2015 erreichte das Projekt daher auch schnell eine gewisse Popularität unter .NET-Entwicklern. Jens Geyer stellt das Projekt in einem Artikel ab Seite 84 vor.

Ihr Max Bold
chefredakteur@maxbold.de

Philip Ackermann

erläutert die Grundsätze der funktional reaktiven Programmierung in JavaScript (S. 50)

Jens Geyer

stellt das in Microsoft Research beheimatete Projekt Orleans vor (S. 84)

Markus Schraudolph

präsentiert das WordPress-Plug-in Formidable zum Erstellen von Formularen (S. 100)

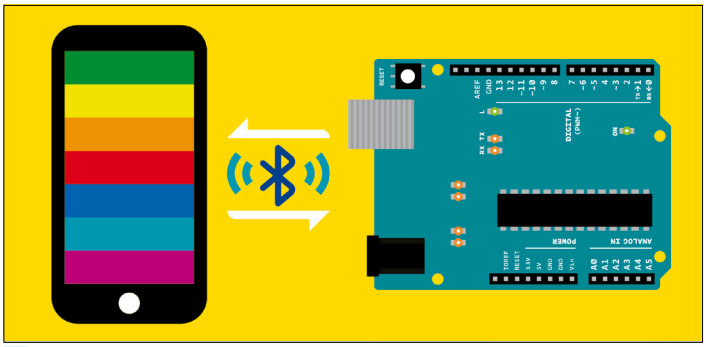
INHALT

Update

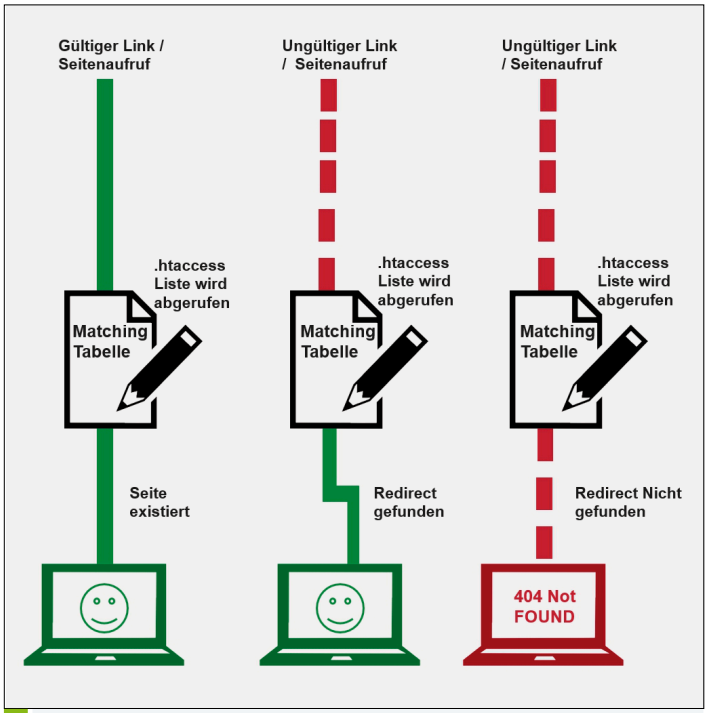
Feature

HTML, CSS & JavaScript

Mobile Development



Der Arduino 101 spart insbesondere beim Entwickeln von Kleinserien wertvolle Zeit **S. 22**

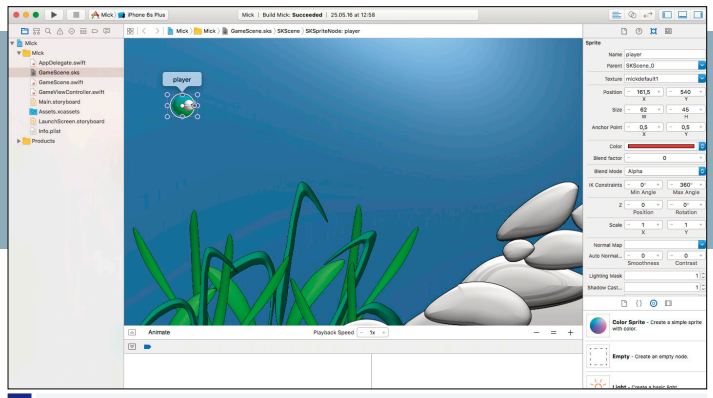


Wird eine Website neu aufgesetzt, gibt es zahllose neue URLs und externe Verlinkungen **S. 56**

Experten in dieser Ausgabe

Dr. Florence Maurice rät Webdesignern bei Layout-Entwürfen zum Umstieg auf Flexbox 34

Christian Bleske zeigt, wie man mit dem SpriteKit Scene Editor Spielszenen visuell entwickeln kann 58



SpriteKit ist die Bibliothek, um Spiele für iOS, tvOS, watchOS und macOS zu schreiben **S. 58**

Performance wird bei Anwendungen eine immer kritischere Anforderung. Deswegen ist es besser, wenn man recht früh die Grenzen seines Systems kennt, um geeignete Gegenmaßnahmen einzuleiten **S. 96**

Ein guter Raumplaner – ob App oder Website – muss leicht zu bedienen und optisch ansprechend sein **S. 124**

Backend

Beyond Dev

Standards

NEWS & TRENDS

AKTUELLE NEWS FÜR ENTWICKLER

Weltweite Dynatrace-Studie

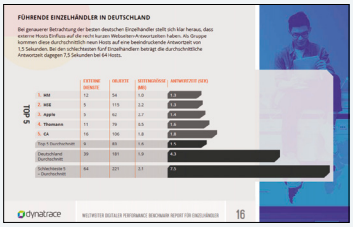
Wie langsam sind Online-Shops?

E-Commerce-Webseiten werden immer langsamer. Die durchschnittliche Antwortzeit der Homepages von Online-Shops nahm im letzten Jahr weltweit von 4,2 auf 4,5 Sekunden zu, in Deutschland von 3,7 auf 4,3 Sekunden. Das zeigt eine weltweite Studie zur digitalen Performance im Einzelhandel, die Dynatrace veröffentlicht hat. Ursache für den langsameren Webseitenaufbau sind steigende Komplexität und Größe der Shop-Webseiten.

Deutschland liegt weltweit mit 4,3 Sekunden Ladezeit hinter den USA im unteren Mittelfeld und auf einem der letzten Plätze in Europa. So erreichen Kunden die E-Shops in Spanien (3,3 Sekunden), Frankreich (3,5 Sekunden) und Großbritannien (3,7 Sekunden) deutlich schneller. Skandinavien liegt mit 4,4 Sekunden nur knapp dahinter, konnte aber die Antwortzeiten im Vergleich zum Vorjahr (5,2 Sekunden) entgegen dem globalen Trend deutlich verbessern.

Die 40 weltweiten Top-Retailer in Bezug auf Performance weisen eine durchschnittliche Antwortzeit der Homepage von 2,2 Sekunden auf. Sie integrieren dort im Durchschnitt nur Angebote von 13 verschiedenen Drittanbietern. Die Seiten enthalten 74 Objekte und haben eine Größe von 1,4 MByte, um die Antwortzeiten niedrig zu halten.

In Deutschland stehen H&M sowie HSE24 mit einer Antwortzeit von 1,3 Sekunden an der Spitze der Performance-Liste. Auf weiteren Top-Plätzen folgen Apple (1,4 Sekunden), Thomann (1,6 Sekunden) und C&A (1,8 Sekunden). Davon haben HSE24 und Apple die wenigsten Drittanbieter (5) integriert, H&M die wenigsten Objekte (54) und Thomann die geringste Seitengröße (0,5 MByte).



Quelle: Dynatrace

Online-Shops werden laut einer Studie immer langsamer

Ein weltweites Best-Practice-Beispiel bildet Apple. Das Unternehmen zeigt, wie ein international tätiger Einzelhändler die Performance kontinuierlich in verschiedenen Ländern hoch halten kann. Es erstellt die Webseiten mit sehr wenigen Drittanbietern, und trotz einer einfachen Navigation sprechen sie Kunden weltweit an. Vor allem sind die Seiten sehr schnell.

www.dynatrace.com/de

1&1 Webhosting

Cloud Hosting auf Docker-Basis

Das 1&1 Managed Cloud Hosting setzt auf die 1&1-Cloud-Server-Infrastruktur auf. Kunden profitieren damit laut 1&1 von einer noch einfacheren Cloud-Lösung, die keinen Administrationsaufwand mit sich bringt. Wartung und kontinuierliche Aktualisierung der Kundensysteme übernimmt 1&1, damit die Webprojekte immer auf dem aktuellen Stand sind.

Das Management kostet im Vergleich mit der Root-Variante, dem 1&1 Cloud Server, lediglich einen Aufpreis von fünf Euro pro Monat. Die genutzten Server-Ressourcen werden minutengenau abgerechnet.

Robert Hoffmann, CEO 1&1 Internet SE: »Mit unserem 1&1 Managed Cloud Hosting unterstützen wir unsere Kunden noch stärker bei der Verwaltung ihrer Webprojekte. Da sie sich nicht mehr mit der Administration ihrer Systeme aufhalten müssen, können sie sich ganz ihrem Kerngeschäft widmen. Die eingesetzte Docker-Container-Technologie macht ihre Cloud-Infrastruktur agiler, steigert die Verfügbarkeit ihrer Online-Dienste und hilft ihnen, noch erfolgreicher im Internet zu sein.«

1&1 Managed Cloud Hosting setzt auf die Docker-Container-Technologie. Damit sie die passende Server-Infrastruktur für ihr Projekt erhalten, haben erfahrene Kunden, Webentwickler, Experten und Agenturen die Option, die Softwarekomponente frei zu konfigurieren. Kunden können zwischen mehreren vorkonfigurierten Stacks wählen oder sich ihren Stack individuell zusammenstellen. Aktuell sind bereits über 20 Kombinationen möglich, die stetig erweitert und auf Kompatibilität geprüft werden.

Zudem können sie bei jedem Server den jeweiligen Stack individuell konfigurieren und weitere Ressourcen minutengenau hinzubuchen. Zusätzlich steht es den Nutzern frei, Server-Ressourcen wie CPU, Memory

Der Webhoster 1&1 bietet ab sofort Managed Cloud Hosting auf der Basis von Docker an

Zahl des Monats

Public-Cloud-Dienste werden ihren Umsatz von **1,6 Milliarden Euro** im Jahr 2015 auf **4,4 Milliarden Euro** im Jahr 2019 steigern. Dabei wird das Segment Software as a Service (SaaS) von 2015 bis 2019 jährlich um **23 %** wachsen.

Quellen: eco, Arthur D. Little

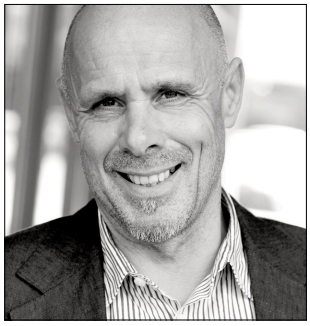
und SSD-Speicher und damit die Performance ihrer Infrastruktur nach ihren Vorstellungen anzupassen. Eine Erweiterung der Komponenten ist sogar im laufenden Betrieb möglich.
https://hosting.1und1.de

eco Verband

Frankfurt wächst zur Daten-Hauptstadt

Frankfurt am Main ist schon heute die Daten-Hauptstadt Deutschlands und wird diese Position in den nächsten Jahren weiter ausbauen können.

Diese Einschätzung vertritt eco – Verband der Internetwirtschaft e. V., der selbst in Köln



Harald A. Summa sieht Frankfurt auf dem Weg zum globalen Superzentrum der digitalen Ökonomie

ansässig ist, und verweist beispielhaft auf das neue C-Lion1-Kabelsystem zwischen Frankfurt und Helsinki.

Dieser sogenannte Northern Digital Highway bietet eine Kapazität, die etwa einer Milliarde ISDN-Leitungen entspricht. »Unsere Entscheidung vor über 20 Jahren, den Deutschen Internet Exchange DE-CIX in Frankfurt aufzubauen, hat sich für die Main-Metropole als nachhaltiger Glücksfall erwiesen«, sagt eco-Geschäftsführer Harald A. Summa selbstbewusst.

Weil die Betreiber der Datenzentren die Nähe zum DE-CIX suchten, stünden heute mehr als die Hälfte aller deutschen Rechenzentren im Großraum Frankfurt. Seit mehr als einer Dekade kann die hessische Großstadt laut eco einen Zuwachs an Rechenzentrumsleistung von 3 Megawatt pro Quartal verzeichnen.

»Die RZ-Branche wächst in Hessen um etwa zehn Prozent jährlich. Mittlerweile haben sich über 35 RZ-Betreiber mit mehr als 50 Lokationen im Rhein-Main-Gebiet angesiedelt«, erläutert Dr. Béla Waldhauser, Leiter der eco Kompetenzgruppe Datacenter Infrastructure.

Die durch den anstehenden Brexit ausgelöste Verlagerungswelle der Finanzwirtschaft von London nach Frankfurt kann auch dem Datenstandort Deutschland zugutekommen, analysiert eco. Allerdings sieht der Verband die Rhein-Main-Metropole hierauf datentechnisch völlig unvorbereitet.

»Wenn die Banker scharenweise nach Frankfurt kommen sollen, wie es sich die Stadt erhofft, dann muss sie auch ganz zügig den Weg für neue Datenzentrums- und Infrastrukturprojekte frei machen. Es gibt kaum eine Branche, die ein so schnelles Internet und so große Datenmengen benötigt wie die Finanzwirtschaft«, klärt eco-Geschäftsführer Harald A. Summa auf. ▶

Information Service Group

Quelle: Information Service Group

As-a-Service-Aktivitäten boomen, während traditionelles Outsourcing abflaut

As-a-Service liegt im Trend

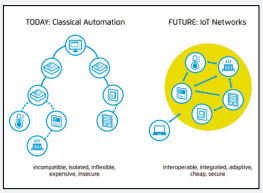
Die Information Services Group (ISG) hat die Ergebnisse des EMEA ISG Index für das zweite Quartal 2016 veröffentlicht. Sie gewähren zum ersten Mal Einblicke in den wachsenden As-a-Service-Markt. Der EMEA ISG Index erfasst kommerzielle Outsourcing-Deals mit einem jährlichen Vertragsvolumen (ACV) von mindestens vier Millionen Euro. Ihm zufolge fiel das ACV des Gesamtmarktes in EMEA (Europa, Naher Osten und Afrika) im zweiten Quartal im Jahresvergleich um 18 Prozent auf 2,2 Milliarden Euro. Das traditionelle Outsourcing büßte 28 Prozent ein und liegt nun bei 1,6 Milliarden Euro. Grund ist die geringe Zahl großer Abschlüsse sowie ein auffälliger Rückgang bei der Restrukturierung von Verträgen. Im gleichen Zeitraum legte der As-a-Service-Markt um 38 Prozent auf nun 600 Millionen Euro zu. In den ersten sechs Monaten dieses Jahres erzielte der Gesamtmarkt in EMEA ein ACV von insgesamt 4,9 Milliarden Euro. Obwohl der Umfang des traditionellen Outsourcings in diesem Zeitraum zurückging, wuchs das ACV des As-a-Service-Marktes um 38 Prozent im Vergleich zum Vorjahr, überschritt zum ersten Mal die Marke von einer Milliarde Euro und erzielte damit einen zuvor noch nie erreichten Rekordwert. Dieses Wachstum speiste sich aus einem eindrucksvollen Aufschwung bei Infrastructure-as-a-Service-Aktivitäten, die um 63 Prozent zulegten. Software as a Service verzeichnete ein Plus von neun Prozent im gleichen Zeitraum.
www.isg-one.de

Internet of Things

Neuer Kommunikationsstandard Lemonbeat

Die Management- und Technologieberatung Cassini Consulting und der Energieversorger RWE treiben das Thema Internet of Things (IoT) voran: Mit der Ausgründung des hauseigenen Sprachprotokolls Lemonbeat in eine gleichnamige Gesellschaft will sich der Konzern international neue Geschäftsfelder erschließen.

Kern der Geschäftsidee ist es, Lemonbeat als Kommunikationsstandard zu etablieren, neue Partner zu gewinnen und das Einsatzspektrum für die Anwendung weiterzuentwickeln. Cassini Consulting ist in renommierten Internetgremien vertreten und hat die Lemonbeat GmbH unterstützt, das Lemonbeat-Sprachprotokoll in das World Wide Web Consortium (W3C) einzubringen.



Quelle: Cassini Consulting

Cassini Consulting und RWE wollen das Thema Internet of Things (IoT) forcieren

Vom Lichtschalter über den Smart Meter bis hin zu komplexen Maschinen und Anlagen – intelligente Lösungen für die vernetzte Kommunikation zwischen unterschiedlichsten Geräten und Anwendungen bilden die Grundlage neuer Geschäftsmodelle für Industriekonzerne.

www.cassini.de

Er blickt in die Zukunft: »Heute wird Frankfurt noch gerne als Banken-Metropole bezeichnet. Aber wenn es gelingt, die positive Rechenzentrums-Entwicklung zu halten oder gar zu beschleunigen, wird sich der Begriff der Internet-Metropole als Synonym für Frankfurt durchsetzen.«

www.eco.de

Stack Overflow

Dokumentation durch die Community

In der diesjährigen Entwickler-Studie 2016 von Stack Overflow kam heraus, dass die technische Dokumentation zu den größten Herausforderungen der täglichen Arbeit im Berufsstand zählt: 36,7 Prozent der Befragten geben die unzureichenden Dokumentation als eine der größten Herausforderungen im Job an. Stack Overflow Documentation ergänzt und verbessert bereits die existierenden Dokumentationsvorgänge: Bedienungsanleitungen werden in lebendige Anleitungen umgewandelt und Entwickler ergänzen in Echtzeit ihre Anwendungsfälle, Implementierungen und Integrationen.

»Mit Documentation verfolgen wir ebenso wie mit der Stack Overflow Q&A-Plattform das Ziel, reale Probleme der Entwickler zu lösen«, sagt Jay Hanlon, Stack-Overflow Vizepräsident. »Durch die geballten Programmierkenntnisse auf der ganzen Welt können wir qualitativ hochwertigere Referenz-APIs und Architekturen erstellen, verglichen mit den Möglichkeiten einzelner Personen.«

Im Gegensatz zu bestehenden Dokumentationslösungen, die oft nur wenige bis gar keine Beispiele bieten, wird mit Documentation ein neuer Weg eingeschlagen: Für jeden Anwendungsfall werden Beispiele gesammelt – auch für solche, die Unternehmen heute noch gar nicht auf dem Radar haben. Documentation liefert den Entwicklern die richtige Information im richtigen Kontext und somit mehr als nur die standardisierten technischen Definitionen. Unternehmen wie Microsoft, PayPal, Twilio, Xamarin und Dropbox sind als Partner an Bord. Sie unterstützen den Launch von Documentation und erweitern zugleich ihre eigenen Dokumentations-Ressourcen.

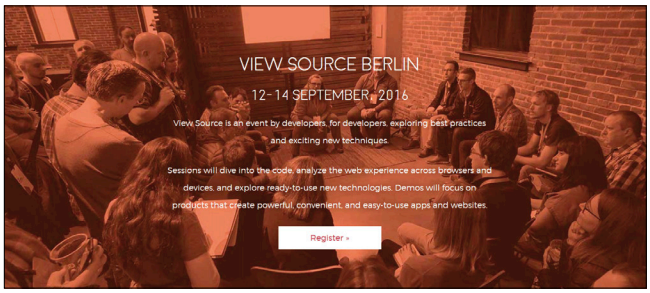
http://stackoverflow.com/documentation

Stack-Overflow-Vizepräsident Jay Hanlon: Mit dem Produkt Documentation reale Probleme der Entwickler lösen

Mozilla

Entwickler-Konferenz erstmals in Berlin

Nach dem erfolgreichen Start in Portland, Oregon, im Jahr 2015, holt Mozilla die Entwickler-Konferenz View Source zum ersten Mal nach Deutschland und unterstreicht damit das große Interesse an Europa und Berlin als wachsende Tech-Spots. Die View Source ist eine Veranstaltung von Entwicklern für Entwickler – mit jeder Menge Zeit und Raum für Diskussionen, Demos und Gespräche. Sie findet vom 12. bis zum 14. September in den Veranstaltungsräumen des Radialsystem V statt. An den drei Veranstaltungstagen wird es Vorträge von 16 international angesehenen Sprechern geben. Die Teilnehmer lernen darin Best-Practice-Modelle kennen und diskutieren über neue, spannende Technologien



Coding und Game Development, Design, Diversity und die Zukunft des Webs gehören zu den Themen der Mozilla View Source

Die Vorträge auf der Hauptbühne werden jeweils 25 Minuten dauern und richten sich an Frontend-Entwickler und alle, die Webseiten sowie Web-Apps designen und erstellen. An den Nachmittagen jedes Konferenztages haben die Teilnehmer in kleinen Diskussionsrunden die Gelegenheit, mit den Referenten zu sprechen und Themen zu vertiefen – egal ob es um die Entwicklung von Web-Apps oder den aktuellen Stand von Open-Source-Projekten in der Tech-Community geht. Zusätzlich werden an jedem Nachmittag interessante Technologien, wie DevTools, VR, Rust oder Web Games, in verschiedenen Demo-Bereichen vorgeführt.

https://viewsourceconf.org/berlin-2016

Jetzt kostenlos testen!

2 x gratis!

Praxiswissen für Entwickler!

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

webundmobile.de/probelesen

Die Digitalwelt in Zahlen

App-Nutzung in Deutschland

In diesem Jahr soll der weltweite Umsatz mit Apps die 50 Milliarden-Dollar-Marke knacken. Kein Wunder, dass sie für jedes Unternehmen immer interessanter werden. Aber welche App lohnt sich? Ein Blick auf die Zahlen gibt Aufschluss.

Auch wenn Deutschland nicht gerade Vorreiter-Status hat, was Digitalisierung anbelangt, steigt auch hierzulande die Anzahl der Smartphone-Besitzer. Laut einer Studie von Comscore MobiLens zählen dazu schon 51 Prozent der männlichen Bevölkerung über 13 Jahre und 49 Prozent der Frauen in der gleichen Altersgruppe. eMarketer prophezeit sogar, dass die Zahl noch in diesem Jahr um 11,1 Prozent steigen wird – das ist mehr als der westeuropäische Durchschnitt.

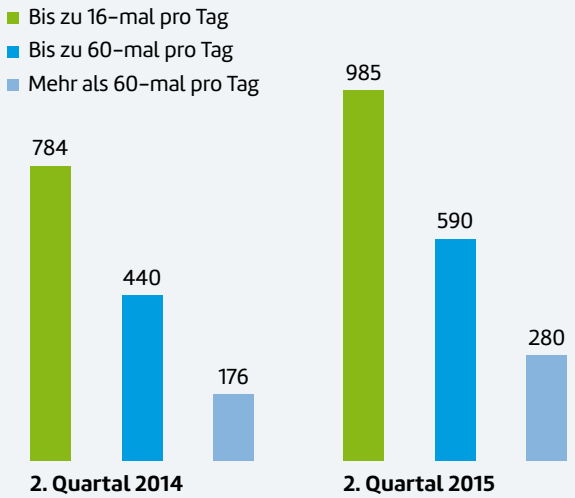
Von den zwei Stunden und 42 Minuten, die täglich durchschnittlich mit dem Smartphone verbracht werden, fällt der Löwenanteil auf die Nutzung von Apps. Überhaupt: Wer ein Smartphone hat, der nutzt (fast immer) auch Apps. In Deutschland steigt auch dieser Anteil noch. In den ersten drei Monaten dieses Jahres stieg er laut Comscore auf 86,9 Prozent – im Vorjahreszeitraum lag er noch bei 85,5 Prozent.

Laut Flurry Analytics ist der Anstieg der App-Nutzung ein globales Phänomen. Eine Viertelmilliarde Menschen ruft täglich mehr als 60-mal eine App auf ihrem mobilen Gerät auf (Stand 2. Quartal 2015). Das entspricht einem Plus von 59 Prozent gegenüber der letzten Erhebung.

Die laufend steigende App-Nutzung heißt aber noch nicht, dass jedes Unternehmen eine eigene neue App braucht. Denn die Deutschen sind wählerisch, was die Anwendungen auf ihren Smartphones betrifft. Die Mehrheit (29,7 Prozent) hat laut Forward AdGroup zwischen 11 und 20 Apps installiert, 18,8 Prozent haben zwischen 21 und 30 Apps auf dem Smartphone und 16,7 Prozent haben sogar mehr als 31 Anwendungen auf ihr mobiles Telefon geladen.

www.flurry.com

App-Nutzung: Die tägliche Nutzung nimmt zu:



App-Nutzer weltweit nach Häufigkeit der täglichen App-Nutzung (alle Zahlenangaben in Millionen)

Quelle: Flurry Analytics

Host Europe

Mehr Leistung für Virtual Server

Host Europe hat die Leistungen seiner virtuellen Server signifikant nach oben geschraubt und damit noch stärker auf die Bedürfnisse von Experten und professionellen Anwendern zugeschnitten.

Um Spitzenleistungen zu gewährleisten, kommen ausschließlich SSD-Datenspeicher zum Einsatz. Mit Plesk 12.5 steht die neueste Version für die Administration zur Verfügung. Für maximale Sicherheit sorgen automatische Backups, Monitoring-Service und SSL.

Ob für Entwicklungsumgebungen, Online-Communities oder Business-Applikationen: Die virtuellen Server in sechs Leistungsstufen sind auf die Bedürfnisse von anspruchsvollen Anwendern ausgelegt. »Das Leistungsplus für die Virtual Server ist eines von mehreren Produkt-Updates, mit denen wir unser Portfolio noch stärker auf die Bedürfnisse von Experten und professionellen Anwendern ausrichten. Dabei setzen wir konsequent auf die Formel Performance plus starke Sicherheits-Features und umfangreiche Service Level Agreements«, so Dr. Claus Boyens, Geschäftsführer bei Host Europe.

Dr. Claus Boyens, Geschäftsführer bei Host Europe, sieht in Experten und professionellen Anwendern die Hauptzielgruppe des Hosters

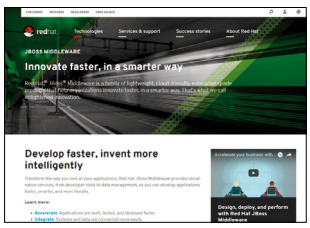
Die SSD-Datenspeicher ermöglichen dank des hohen Datendurchsatzes sehr schnelle Lese- und Schreibzugriffe und eignen sich daher besonders für Datenbanken und dynamische Webanwendungen. Administratoren haben vollen Zugriff auf das System und können die Konfigurationen jederzeit anpassen. Als Betriebssysteme stehen Windows Server 2012 R2, Debian 8, Ubuntu 14.04 oder CentOS 7 zur Auswahl. Die Service Level Agreements garantieren eine Verfügbarkeit von 99,95 Prozent.

www.hosteurope.de

Red Hat JBoss Data Grid 7

Plattform für Real-Time Data Analytics

Red Hat liefert ab sofort Red Hat JBoss Data Grid 7 aus. Die führende In-Memory-Datenbanktechnologie von Red Hat lässt sich als verteilter Cache, NoSQL-Datenbank oder Event Broker nutzen.



Von JBoss Data Grid gibt es eine neue Version

Die neueste Version bietet Erweiterungen, mit denen Unternehmen mit Hilfe von Real-Time Data Analytics neue Einblicke in ihre Geschäftsabläufe erhalten und so eine höhere Agilität und eine bessere Wettbewerbsfähigkeit erzielen können.

JBoss Data Grid 7 unterstützt Apache Spark, ein Open-Source-Framework zur Entwicklung datenintensiver Applikationen. Spark verarbeitet die Daten direkt im Hauptspeicher.

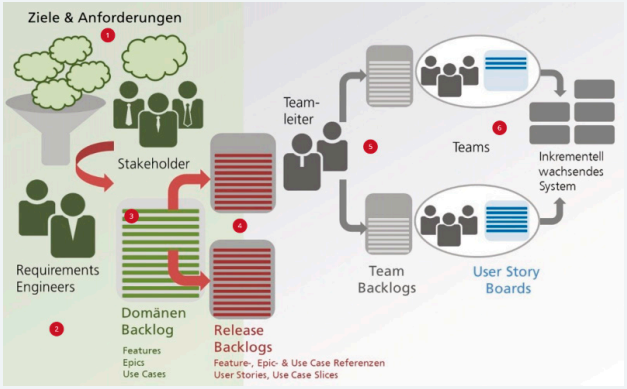
Microtool Objectif RPM

Neue Software für agile Teams

Der Umgang mit Anforderungen ist ein wesentlicher Faktor für die erfolgreiche agile Entwicklung von Software, Systemen und Services. Dafür hat die Berliner Microtool GmbH eine neue Software veröffentlicht: Objectif RPM. Das Produkt ist eine Client-Server-Lösung, die lokale und verteilte, skalierbare Teams bei der Entwicklung von Anwendungen unterstützt. Der Fokus bei Objectif RPM liegt im Zusammenspiel zwischen Requirements Engineering und agiler Entwicklung: Für die agile Entwicklung ist es nicht notwendig, dass alle Anforderungen vollständig definiert werden, bevor Entwicklerteams mit ihrer Arbeit beginnen können. Aufgabe des Requirements Engineerings für die agile Produktentwicklung ist es, Anforderungen zu »produzieren«, die innerhalb kurzer Zeit realisierbar sind und damit von den Entwicklungsteams für die Sprintplanung verwendet werden können. Das Requirements Engineering muss im Projektverlauf regelmäßig für Nachschub an Anforderungen sorgen und dabei vorzugsweise solche liefern, die einen hohen Wert für die Stakeholder schaffen.

Objectif RPM adressiert den gesamten Ablauf zur Produktion von Software, Systemen und Services – von der Anforderungserfassung über die Anforderungsanalyse und Modellierung mit SysML- und UML-Diagrammen, mit der Erfassung von Stakeholdern und Zielen sowie der Definition von Systemkontext, Systemgrenzen und Systemarchitektur bis hin zur Projektplanung und Projektdurchführung auf Basis der erhobenen Anforderungen. Für die Projektdurchführung bietet die Software ein umfassendes Backlog-Management an. Die Software unterstützt Requirements Engineers, Business-Analysten, Systemarchitekten, Projektleiter, sowie Projektmitarbeiter in lokalen oder verteilten Teams und Tester.

www.microtool.de



Objectif RPM unterstützt lokale und verteilte Teams bei der Entwicklung von Anwendungen

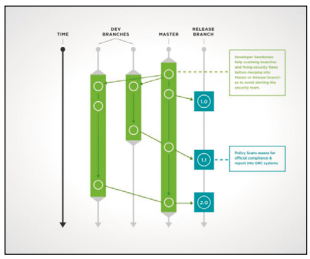
In Kombination mit der In-Memory-Speicherung von JBoss Data Grid können Unternehmen deutliche Geschwindigkeitsvorteile gegenüber konventionellen Systemen erzielen. Zudem lässt sich JBoss Data Grid jetzt als Hadoop-kompatible Datenquelle einsetzen. Durch die Unterstützung des Hadoop InputFormat und OutputFormat können Anwender jetzt viele Analytics-Tools einsetzen, die das Hadoop-I/O-Format verwenden.

www.redhat.com

Veracode

Kontrolle bei der Erstellung sicherer Software

Mit der Developer Sandbox stellt Veracode eine neue, patentierte Funktion seiner Plattform für Anwendungssicherheit vor. Sie gibt Entwicklern bereits in einem frühen Stadium der



Mit der neuen Developer Sandbox können Entwickler Sicherheitsrisiken in ihrem Code frühzeitig erkennen und beheben

Entwicklung mehr Kontrolle über die Anwendungssicherheit und macht gleichzeitig formale, richtlinienbasierte Softwareprüfungsverfahren genauer und effektiver. So verändert die Developer Sandbox auch die Dynamik zwischen Entwickler- und Sicherheits-/Risikoteams.

Mit der Developer Sandbox können Entwickler komplette Anwendungen oder einzelne Komponenten schon während der Erstellung scannen und verbessern, bevor sie die Software an eine formelle Richtlinien- oder Sicherheitsüberprüfung übergeben. Die bisher vorherrschende Konzentration auf die Schwachstellen in der Arbeit des Entwicklungsteams, wenn Sicherheits- und Risikoteams nur frühe Versionen des Codes zu sehen bekamen, entfällt damit. Dies hatte bisher dazu geführt, dass Softwarerisiken oder Compliance-Fehler Thema waren, lange bevor die Anwendung überhaupt gestartet wurde oder der Entwickler eine Chance hatte, Korrekturen vorzunehmen.

Die Developer Sandbox unterstützt Entwickler bei der Arbeit in agilen oder DevOps-Umgebungen. Dass bereits während der Entwicklung früher häufigere Tests des Codes auf Sicherheitsrisiken möglich sind, passt zu kürzeren Entwicklungszyklen und häufigeren Neuversionen. Im Endeffekt durchläuft nur hochwertiger Code die formale Prüfung. Damit sinkt auch die Wahrscheinlichkeit, dass kritische Sicherheitsrisiken erst spät im Entwicklungsprozess entdeckt werden und eine Entscheidung zwischen verspäteter Fertigstellung einer Version und möglichem Geschäftsrisiko getroffen werden muss.

»Die Entwickler wurden in der Vergangenheit oftmals aus der Sicherheitsdiskussion ausgeschlossen«, kommentiert Julian Totzek-Hallhuber, Solution Architect bei Veracode. »Tatsache ist, dass sie durchaus guten und sicheren Code schreiben wollen, aber häufig keinen Zugang zu Tools haben, die zu ihrer Arbeitsweise passen. Die Developer Sandbox ändert diese Gleichung, indem sie Entwicklern praktikablen Zugang zur branchenweit leistungsstärksten Plattform für Anwendungssicherheit bietet.«

Software-Entwickler sind häufig ungenügend in den Grundregeln des sicheren Co-

dings geschult. Der Statusbericht zur Softwaresicherheit von Veracode zeigt, dass Sicherheitsrisiken oft aus falsch konfiguriertem SSL oder fehlerhafter Verschlüsselung herrühren – Funktionalitäten, die eigentlich die Sicherheit verbessern sollten. Die Veracode Developer Sandbox nutzt hier das Tool für die statische Analyse von Veracode, das durch die Abtastung von fast zwei Billionen Zeilen Code abgestimmt und verbessert wurde. Dies bietet Entwicklern mit unzureichenden Sicherheitskenntnissen eine wertvolle Hilfestellung, um die Codesicherheit zu verbessern, sowie die Möglichkeit, sicheres Coding zu erlernen und praktisch anzuwenden.

Ergänzend kommen weitere Tools von Veracode hinzu. Dazu zählt die Analyse der Softwarezusammensetzung, die Risiken in heute häufig in der Software-Entwicklung genutzten Open-Source-Komponenten identifiziert, sowie integrierte Schulungs-Tools. Diese helfen, Schwachstellen bereits während der Code-Erstellung zu beheben. Damit macht Veracode die Entwicklung von sicherer Software einfacher.
www.veracode.com

DomainFactory

CloudServer kostenfrei für Hackathons

DomainFactory unterstützt Hackathons künftig durch die kostenlose Bereitstellung von JiffyBox CloudServern. Damit möchte DomainFactory insbesondere Gründer, kleine Unternehmen und Vereine unterstützen, Innovationen zu fördern. Organisatoren können sich dazu für das JiffyBox-Hackathon-Programm anmelden.

Hackathons haben als Veranstaltungsformat mittlerweile viele Bereiche erobert, vom Bildungs-, Gesundheits- und Bankwesen bis hin zu Journalismus und Kultur. Kaum ein anderes Format eignet sich so gut, in kurzer Zeit viele realisierbare Ideen zu generieren. »Für Gründer, Organisationen und gemeinnützige Initiativen halten wir Hackathons für das beste Mittel, in kurzer Zeit wertvolle Impulse für Innovationen zu erhalten. Doch gerade sie verfügen oft über begrenzte Ressourcen. Wir möchten neue Lösungsansätze und den Mut zu Innovationen in Deutschland fördern und haben uns deshalb entschlossen, Hackathons künftig stärker zu unterstützen«, so Stephan Wolfram, Geschäftsführer bei DomainFactory.

Für Hackathons stellt DomainFactory künftig kostenlose JiffyBox CloudServer zur Verfügung

Außer eine produktive Arbeitsatmosphäre zu schaffen und gutes Essen bereitzustellen, zählt es zu den wichtigsten Aufgaben für Organisatoren, für moderne technische Ausstattung zu sorgen, damit die Teams reibungslos arbeiten können. Dazu gehören auch Server-Kapazitäten, mit denen die Projekte umgesetzt werden können. Aus diesem Grund bietet DomainFactory Organisatoren von Hackathons an, JiffyBox CloudServer kostenlos für ihre Veranstaltung zu nutzen.

Die JiffyBox eignet sich sehr gut für kurzfristige Projekte, wie sie bei Hackathons umgesetzt werden. Mit zwei Klicks können die Projektteams eine neue Linux-Umgebung aufsetzen und innerhalb von fünf Minuten in der eigenen Umgebung mit der Umsetzung ihrer Idee beginnen.
www.df.eu/hackathon

Smartphone-Markt

Samsung baut die Führung weiter aus

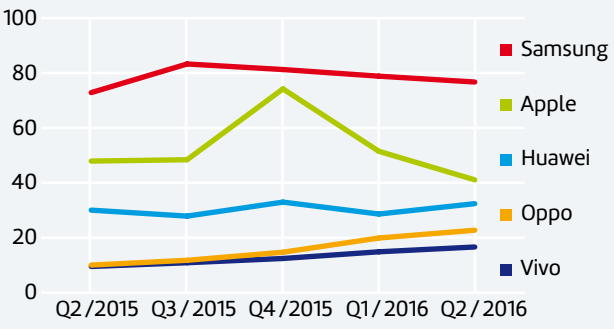
Auch im zweiten Quartal konnte sich der globale Smartphone-Markt nur wenig verbessern: Gemäß den Marktforschern von IDC wurden 343,3 Millionen Geräte verkauft, das waren gerade einmal 0,3 Prozent mehr als im vergleichbaren Vorjahreszeitraum. Im Vergleich zum sehr schwachen ersten Quartal dieses Jahres stiegen die Verkäufe um 3,1 Prozent.

Dabei konnte der Marktführer Samsung dank des Erfolgs seiner Galaxy-S7-Flaggschiffe seine Position klar ausbauen: Die Koreaner verkauften 77 Millionen Smartphones, das waren 5,5 Prozent mehr als ein Jahr zuvor. Der Marktanteil stieg von 21,3 auf 22,4 Prozent.

Zudem schwächelt der Verfolger Apple, dessen Verkäufe im Jahresvergleich um 15 Prozent auf 40,4 Millionen Stück fielen. Das entspricht einem Rückgang des Marktanteils von 13,9 auf 11,8 Prozent. Das zweite Quartal ist für Apple traditionell schwach, da viele Kunden auf ein neues iPhone warten, das im dritten Quartal kommen soll. Laut den Analysten von IDC war die Markteinführung des iPhone SE in der Mitte des zweiten Quartals ein Erfolg, allerdings fiel dadurch der durchschnittliche Verkaufspreis für Apple-Smartphones von 662 auf 595 US-Dollar.

Die Kultmarke spürt sogar schon den Atem des Dritten, Huawei. Bei den Chinesen hat sich zwar das rasante Wachstum der letzten Quartale etwas verlangsamt, doch trotzdem stiegen die Verkäufe im Jahresvergleich um 8,4 Prozent auf 32,4 Millionen Stück, was einem Marktanteil von 9,4 Prozent entspricht. In der Heimat ist Huawei Marktführer und in Europa konnte laut IDC ebenfalls ein starkes Wachstum verzeichnet werden. Mit dem P9 und dem Mate 8 ist der Hersteller zudem auch in der Oberklasse erfolgreich.

Auf den weiteren Plätzen folgen mit Oppo mit 6,6 Prozent Anteil und Vivo mit 4,8 Prozent zwei hierzulande weitgehend unbekannte Hersteller aus China, die beide stark zulegen konnten.
www.idc.com



Im zweiten Quartal stagnierten die weltweiten Smartphone-Verkäufe (Verkaufszahlen in Millionen Stück)

Quelle: IDC

DOMAINS | E-MAIL | **HOSTING** | SHOPS | SERVER

NEU! 1&1 MANAGED CLOUD HOSTING

Das beste aus zwei Welten!

Jederzeit skalierbare und flexible Server-Ressourcen kombiniert mit einem leistungsstarken Hostingpaket: das **neue 1&1 Managed Cloud Hosting ist da!** Ideal für Online-Projekte mit höchsten Ansprüchen an Verfügbarkeit, Sicherheit und Flexibilität.

- ✓ **Dedizierte Ressourcen**
- ✓ **20+ Stack Variationen**
- ✓ **Von 1&1 Experten gemanagt**
- ✓ **Flexibel skalierbar**
- ✓ **Bereitstellung < 1 Minute**

Cloud Vendor Benchmark 2016
Rising Star Germany
experton GROUP

intel XEON inside™
Trusted Performance.
Intel® Xeon® processors.

1 MONAT KOSTENLOS TESTEN!*

Internet made in Germany

1 TEST 30 TAGE AUSPROBIEREN
1 MONAT VERTRAGS-LAUFZEIT
1 ANRUF EXPERTEN-RAT

DE: 02602/9691
AT: 0800/100668

1&1

1und1.info

*1&1 Managed Cloud Hosting 1 Monat kostenlos testen. Danach ab 9,99 €/Monat. Kündigung im ersten Monat jederzeit möglich. Keine Bereitstellungsgebühr. Preise inkl. MwSt. 1&1 Internet SE, Elgendorfer Straße 57, 56410 Montabaur

Musik-Streaming

Vier von zehn Usern streamen Musik im Internet

Musik-Streaming-Dienste wie Spotify, Apple Music, Napster, Deezer und Soundcloud werden auch in Deutschland immer populärer. 39 Prozent der Internetnutzer in der Bundesrepublik ab 14 Jahren nutzen diese Dienste. Das hat eine am Mittwoch in Berlin veröffentlichte repräsentative Befragung im Auftrag des Branchenverbands Bitkom ergeben.

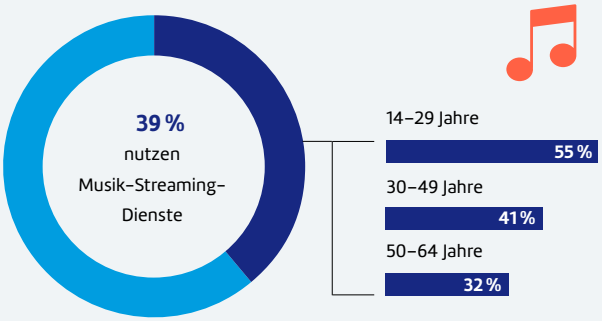
Die große Mehrheit der Streaming-Nutzer setzt auf werbefinanzierte Angebote. Nur rund ein Fünftel (22 Prozent) von ihnen bezahlt für die Musikangebote.

Anbieter von Musik-Streaming seien inzwischen mehr als nur riesige Online-Plattenläden, sagte Timm Lutter, Bitkom-Experte für Consumer Electronics & Digital Media. »Nutzer finden dort kuratierte Playlists, Konzerttickets ihrer Lieblingskünstler oder die Möglichkeit, Musik mit Freunden zu teilen.« Die Attraktivität der Musik-Streaming-Dienste komme auch der Musikindustrie zugute: Laut Global Music Report 2016 ist Streaming die am schnellsten wachsende Einnahmequelle der Branche. Dank dieser Einnahmen steige auch der weltweite Umsatz mit Musik um 3,2 Prozent. Musiker sehen von den Einnahmen aber, glaubt man Berichten, sehr wenig.

Das Nutzungsverhalten ist in den verschiedenen Altersklassen recht unterschiedlich. Über die Hälfte der 14- bis 29-jährigen Internetnutzer streamt Musik online (55 Prozent). Unter den 30- bis 49-Jährigen geben 41 Prozent an, Musik zu streamen, unter den 50-bis 64-Jährigen sind es dagegen nur 32 Prozent.

Gegenüber einem klassischen Download hat Streaming den Vorteil, dass die Inhalte auf nicht erst langwierig auf die Endgeräte heruntergeladen werden müssen, sondern nur vorübergehend zwischengespeichert werden. Dies ermöglicht es, Audiodateien direkt und ohne Wartezeit wiederzugeben. Musik-Streaming-Dienste bieten sowohl kostenlose als auch kostenpflichtige Streaming-Angebote. Die kostenlose beziehungsweise werbefinanzierte Variante begrenzt oft die Funktionen oder spielt zwischen einzelnen Titeln Werbespots ab. Kostenpflichtige Premiumaccounts geben dem Nutzer werbefreien und unbegrenzten Zugang zu den Musik-Archiven, oft in besserer Abspielqualität. Andere Streaming-Dienste bieten Musikern die Möglichkeit, eigene Stücke hochzuladen und zu teilen.

www.bitkom.org



Immer mehr Internet-User nutzen Musik-Streaming

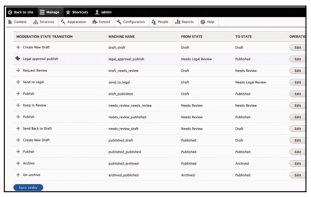
web & mobile developer 10/2016 Quelle: Bitkom

Acquia Lightning

Schneller Einstieg in Drupal 8

Acquia stellt mit Acquia Lightning eine kostenlose Open-Source-Distribution von Drupal 8 vor. Sie beschleunigt Projekte, vereinfacht die redaktionelle Tätigkeit von Mitarbeitern in den Fachabteilungen und ermöglicht es auch technisch nicht versierten Anwendern, Inhalte zu verwalten und aufzubereiten.

Acquia Lightning ist das perfekte Starterkit für Drupal 8 und enthält alle wesentlichen Funktionen für die unternehmensweite Bereitstellung von Digital Experiences. Acquia Lightning ist ab sofort auf der Hosting-Plattform Acquia Cloud Free (www.acquia.com/de/free) verfügbar, auf der Entwickler ihre Drupal-8-Projekte unmittelbar starten können.



Das Starterkit Acquia Lightning beschleunigt die Entwicklung von Drupal-8-Projekten

Agenturen und Unternehmen, die mit Drupal 8 arbeiten, bietet Acquia Lightning Best Practices für den Start neuer digitaler Initiativen, die Reduktion von Komplexität und die Verkürzung des Go-to-Market.

»Acquia Lightning setzt einen neuen Maßstab für den Einstieg in Drupal-8-Projekte und bewältigt viele der größten Herausforderungen, die üblicherweise am Anfang dieser Projekte warten«, sagt Karl Kedrovsky, Solution Architect bei der Full-Service-Digital-Marketing- und Werbeagentur VML. »Mit Lightning ist die Basisarbeit schnell erledigt und wir können uns voll auf unsere Stärke konzentrieren: Mehrwert für unsere Kunden zu schaffen.«

»Mit Lightning stellt Acquia eine optimal integrierte Software bereit, die hohe Sicherheitsstandards erfüllt und mit der Unternehmen schneller neue Drupal-8-Webseiten erstellen können«, sagt Chris Stone, Chief Product Officer bei Acquia.

»Lightning ermöglicht Entwicklern, leistungsfähige, einfach einsetzbare Publishing-Tools für die Redaktionsteams zu erstellen. Drag-and-Drop-Layouts, Medien-Management, konfigurierbare Workflows und Preview-Funktionalitäten schaffen die Basis für eine attraktive Digital Experience. Auf Hunderten von Websites kommt Acquia Lightning bereits zum Einsatz. Wir schätzen, dass sich mit Lightning die Entwicklungszeit in jedem Drupal-8-Projekt um 20 Prozent verkürzen lässt.«

www.acquia.com/lightning

ownCloud 9.1

Zwei-Faktor-Authentifizierung integriert

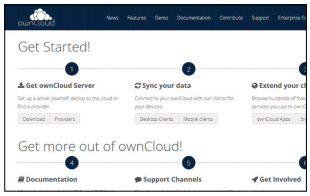
Die neue Version 9.1 von ownCloud kommt mit zahlreichen Detailverbesserungen für Community- und Enterprise-Anwender.

Benutzer der ownCloud-Community-Version können jetzt ownCloud Contacts, ownCloud Calendar und ownCloud Mail bequem über den App Store updaten und installieren. Insbesondere für die Anwendung im Unternehmen, aber auch für private Nutzer, wurde das Thema Authentifizierung angegangen.

»In ownCloud v9.1 haben wir zahlreiche Optimierungen und Neuerungen integriert, die die

Nutzung stabiler, schneller und sicherer machen«, kommentiert Holger Dyroff, als COO für die Produktstrategie von ownCloud zuständig.

»Neben der erweiterten Zwei-Faktor-Authentifizierung und besseren Behandlung von Windows Network Drives betrifft das beispielsweise Permalinks für interne Shares oder das vor allem von Unternehmen gewünschte Tagging auf Gruppenebene. Das Besondere: Die meisten Änderungen kommen sowohl in der Community- als



ownCloud stellt das neue Release 9.1 seiner Software vor

auch in der Enterprise-Version zum Tragen.«

Bisher konnte eine sichere Zwei-Faktor-Authentifizierung via SAML/Shibboleth realisiert werden. In der neuen Version wurde das Authentifizierungsverfahren geändert, sodass jetzt über Plug-ins weitere Technologien sowie Tokens genutzt werden. Das erhöht die Zugangssicherheit, zudem erhalten Administratoren die Möglichkeit, einzelne Tokens zu deaktivieren.

Mit sogenannten Time Based One-Time Passwords (TOTP) können Anwender selbstständig die Sicherheit ihrer Accounts erhöhen, indem sie Dienste wie Google Authenticator oder eine Open-Source-Implementierung des TOTP-Standards verwenden.

Externe Anbieter können auf Basis des neuen Frameworks Apps entwickeln, die den Benutzer dann bei der Anmeldung zusätzlich zu dem ownCloud Passwort nach einem zweiten Faktor fragen. Klassische Passwort-Angriffe werden somit unterbunden, sensible Daten noch zuverlässiger geschützt. Die Benutzer können nur noch auf die Daten zugreifen, wenn sie das Passwort wissen und im Besitz des zweiten Faktors sind.

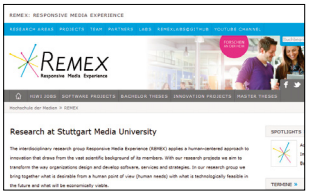
https://owncloud.org

Remex Summit 2016

Remex-Forschungsgruppe präsentierte Ergebnisse

Im Juli trafen sich die Professoren, Doktoranden und Projektmitarbeiter der Remex-Forschungsgruppe der Hochschule der Medien (HdM) zum jährlichen Workshop auf Schloss Monrepos nahe Ludwigsburg. Präsentiert wurden Ergebnisse aus Forschungsprojekten, Doktorarbeiten und aktuelle Forschungsansätze.

Der Forschungsleuchtturm »Responsive Media Experience« (Remex) besteht seit 2013 als ein Zusammenschluss diverser Forschungsschwerpunkte. Remex konzipiert und erforscht innovative Anwendungen und Technologien, die sich an den Benutzer, den Nutzungskontext und die benutzten Endgeräte adaptieren, um eine optimale Usability und User Experience zu erreichen. Schwerpunkte innerhalb des



Die Remex-Forschungsgruppe will eine optimale Usability und User Experience erreichen

Forschungsleuchtturms befassen sich zum Beispiel mit personalisierten und barrierefreien Benutzerschnittstellen, adaptiven E-Learning-Plattformen,

Technikmarkt in Westeuropa

Unterschiedliche Tendenzen

Der westeuropäische Markt für technische Gebrauchsgüter entwickelte sich im ersten Quartal trotz unterschiedlicher Tendenzen in den einzelnen Sektoren im Vergleich zum Vorjahreszeitraum konstant. Das ist das Ergebnis einer Erhebung der GfK. In den Bereichen Elektrogroßgeräte, Elektrokleingeräte und Telekommunikation stiegen die Umsätze verglichen zum ersten Quartal 2015. In den übrigen Sektoren fiel der Umsatz dagegen, wenn auch nur leicht.

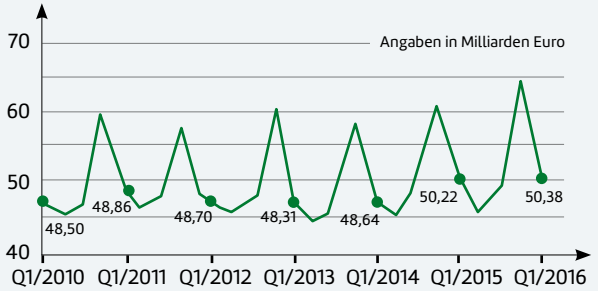
Mit einem Minus von 1,4 Prozent im Vergleich zum Vorjahresquartal entwickelt sich der Umsatz im Sektor Unterhaltungselektronik im ersten Quartal negativ. Videospielkonsolen, Lautsprecher sowie Mini-Dockingstation-Lautsprecher wiesen ein zweistelliges Umsatzwachstum auf. Im ersten Quartal 2016 wurden im Bereich Foto 1,1 Milliarden Euro umgesetzt.

Mit einem Umsatz von circa 13,3 Milliarden Euro stellte der IT-Sektor im ersten Quartal 2016 den größten Bereich im westeuropäischen Technikmarkt dar. Im Vergleich zum Vorjahresquartal waren die Umsätze etwas geringer. In Griechenland fiel der Umsatz um 38,1 Prozent im Vergleich zum Vorjahreszeitraum. Die Umsätze in den Kernkategorien des IT-Sektors entwickeln sich schleppend, auch, wenn sich das Segment Mobile Computing – unterstützt durch Verkäufe im Bereich Convertibles und Tablets – gut entwickelt. Der Umsatz mit größeren Modellen sowie im Bereich Desktop-Computer ging zurück. Auch der Umsatz in der Kategorie Media-Tablets fiel weiter. Das limitierte Wachstum im Hardware-Bereich wirkt sich ebenfalls negativ auf den Verkauf von Zubehör aus.

Im ersten Quartal 2016 ist der Umsatz im Sektor Telekommunikation im Vergleich zum Vorjahreszeitraum um 2,5 Prozent gewachsen. Mit einem Gesamtwert von 12,1 Milliarden Euro handelt es sich bei dem Bereich Telekommunikation um den zweitgrößten Sektor des westeuropäischen Technikmarktes.

Im Markt für Smart- und Mobilephones gab es ein gemischtes Bild. Verschiedene Länder leiden an der Marktsättigung im Smartphone-Segment. In Ländern wie Großbritannien, Frankreich, Italien und den Benelux-Staaten steigt der Umsatz in dem Segment dagegen. Wearables gehört zu der Produktgruppe im Telekommunikationssektor, die wächst.

www.gfk.com



Im ersten Quartal 2016 wurden im westeuropäischen Technikmarkt insgesamt 50,8 Milliarden Euro umgesetzt

web & mobile developer 10/2016 Quelle: GfK

Tablet-Verkäufe

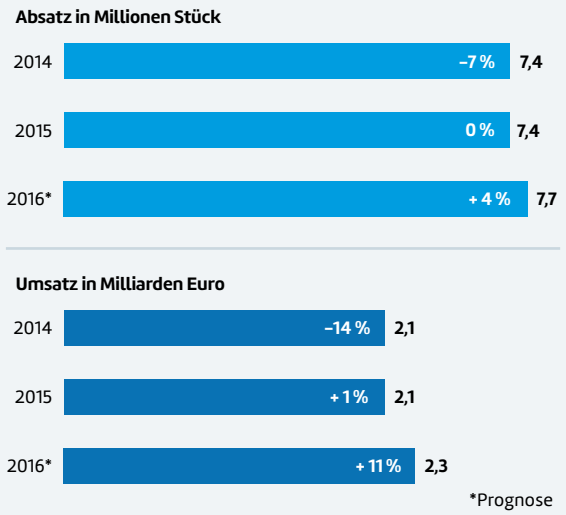
Starke Nachfrage nach leistungsfähigen Tablets

Nachdem im Frühjahr noch mit rückläufigen Umsätzen und Verkaufszahlen für Tablet-Computer gerechnet wurde, soll sich das Geschäft nun nach neuesten Berechnungen des Digitalverbands Bitkom sehr positiv entwickeln. Die Verkaufszahlen sollen demnach 2016 um vier Prozent auf 7,7 Millionen Geräte steigen. Der Umsatz dürfte sogar um elf Prozent auf 2,3 Milliarden Euro zulegen. »Treiber für dieses starke Wachstum sind die sogenannten Detachables, also Tablet-Computer, die fest mit einer Tastatur verbunden werden können und dann für die gleichen Aufgaben eingesetzt werden wie ein vollwertiges Notebook oder ein PC«, sagt Bitkom-Chef Dr. Bernhard Rohleder. Das Interesse an klassischen Tablets ist dagegen sowohl im Privatkunden- als auch im Geschäftskundenbereich rückläufig.

»Die Detachables sind erst seit etwa einem Jahr breit am Markt verfügbar und werden stärker nachgefragt als erwartet«, so Rohleder. »Dabei ist das Interesse von Privatanwendern und von Geschäftskunden gleichermaßen hoch. Ein Grund dafür: Je nach Betriebssystem sind Detachables sehr einfach in die bestehende IT-Infrastruktur zu integrieren«, so Rohleder. Da diese Geräte aktuell noch deutlich teurer sind als einfachere Tablet-Computer, steigt auch der Durchschnittspreis pro Gerät von 280 Euro im vergangenen Jahr auf nun 300 Euro.

Aktuell benutzen 41 Prozent der Bundesbürger ab 14 Jahren einen Tablet-Computer, vor zwei Jahren lag der Anteil gerade einmal bei 28 Prozent. »Die Nachfrage hat sich durch das Angebot an Smartphones mit immer größeren Displays etwas abgeschwächt. Für viele Nutzer ersetzt ein solches Phablet die Anschaffung eines eigenen Tablet-Computers«, so Rohleder. Hinzu kommt, dass durch Software-Updates auch ältere Tablets mit neuen Funktionen ausgestattet werden und so eine Neuanschaffung verschoben werden kann.

www.bitkom.org



Der Tablet-Umsatz legt 2016 voraussichtlich um 11 Prozent auf 2,3 Milliarden Euro zu

web & mobile developer 10/2016 Quelle: Bitkom

Technologien zur Integration mobiler Endgeräte in Fahrzeug und Infotainment, sowie Car-2Car- und Car2X Technologien. Studenten wirken im Forschungsleuchtturm als wissenschaftliche Hilfskräfte und in Form von Lehrprojekten und Abschlussarbeiten mit.

Der Forschungsleuchtturm verfügt über ein Usability-Labor und über ein Mobile Lab. Beide Labors stehen auch den Studenten für ihre im Rahmen der Lehre ausgeübten Projekte zur Verfügung.

http://remex.hdm-stuttgart.de

Microsoft

Neue Sicherheitsfunktionen für Windows 10

Mit dem Anniversary Update für Windows 10 kommen umfangreichen neue sowie verbesserte Sicherheitsfunktionen für Anwender wie Unternehmen.

So erhalten Firmen mit Windows Defender Advanced Threat Protection (WDATP) eine Lösung auf Basis von Machine-Learning, die Verhaltensmuster erkennt und Unternehmen die Möglichkeit gibt, komplexe Angriffe oder Datenlecks im Netzwerk schneller selbst zu erkennen, zu untersuchen und zu beseitigen.

Mit dem Anniversary Update erhält Windows 10 umfangreiche neue Sicherheitsfunktionen

Darüber hinaus schützt Windows Information Protection vor unbeabsichtigten Datenverlusten durch die Trennung von personenbezogenen und Unternehmensdaten. Zudem unterstützt Windows Hello nun auch Anwendungen und wird im Browser Microsoft Edge verfügbar, sodass Nutzer sich über eine biometrische Authentifizierung ohne Passwort auf Webseiten anmelden können.

Der mitgelieferte Malwareschutz Windows Defender wird verbessert und durch eine Option erweitert, anhand derer sich automatisch regelmäßige und schnelle Computerscans durchführen lassen.

Nach Abschluss der Scans erhalten Anwender eine Zusammenfassung und werden im Fall einer Bedrohung umgehend benachrichtigt.

Windows Defender ATP ist eine neue entwickelte Post-Breach-Schutzebene, die den Windows-10-Sicherheitsstack vervollständigt. Diese Lösung ergänzt die bestehende Endpunktsicherheit durch Windows Defender, SmartScreen und verschiedenste Features zur Betriebssystemhärtung. Der neue Dienst wurde zur Erkennung von hochentwickelten Angriffen sowie zur Reaktion auf diese Bedrohungen entwickelt.

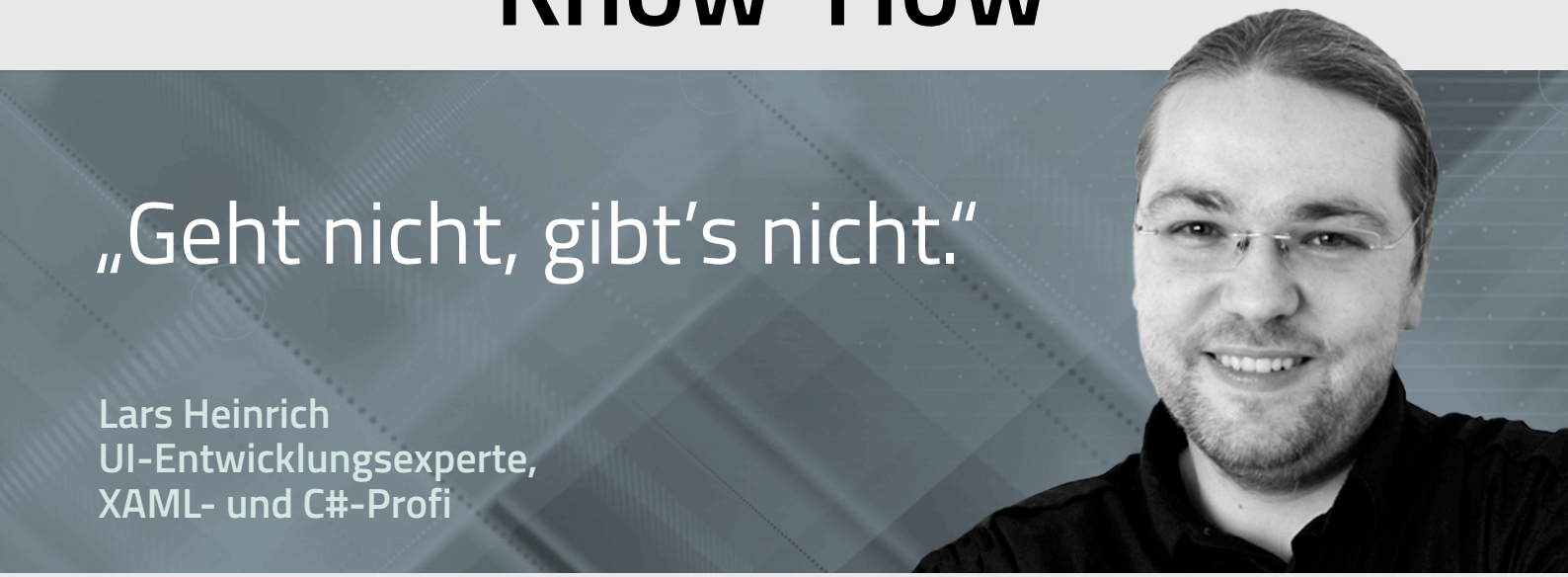
Mit Windows Information Protection schützen sich Unternehmen vor unbeabsichtigten Datenverlusten und -weitergaben. Der Datenschutz funktioniert über die Trennung von personenbezogenen Daten und Unternehmensdaten. Dokumente lassen sich über Labels klassifizieren, um den Zugriff darauf einzuschränken. Diese Klassifizierung ist mit dem Dokument verknüpft und nicht davon abhängig, wo es gespeichert oder von wo aus darauf zugegriffen wird. Außerdem kann der Ersteller eines Dokuments über die Labels nachvollziehen, wer auf ein Dokument zugegriffen hat.

www.microsoft.com

developer media

developer media Trainings

Updates für Ihr Know-How



Effiziente Softwarearchitekturen mit PHP

Trainer: Stefan Priebsch

2 Tage, 14.-15.11.2016, München
Ab 1.799 EUR zzgl. MwSt.

Cross-Plattform-Apps mit C# und Xamarin

Trainer: Sebastian Seidel

3 Tage, 22.-24.11.2016, Köln
Ab 2.199 EUR zzgl. MwSt.

Apps für Windows 8/10 entwickeln

Trainer: Lars Heinrich

2 Tage, 08.-09.12.2016, Köln
Ab 1.799 EUR zzgl. MwSt.

PHPUnit erfolgreich einsetzen

Trainer: Sebastian Bergmann

2 Tage, 07.-08.11.2016, Köln
Ab 1.799 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

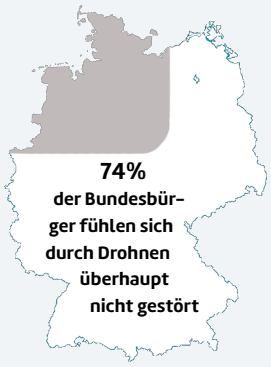
developer-media.de/trainings

Drohnen-Umfrage 2016

Keine Angst vor Drohnen

Die Deutschen stehen Drohnen positiv gegenüber: Weder die Diskussion um Lieferdrohnen noch die inzwischen weite Verbreitung der Multikopter im Privatbereich hat die Bundesbürger in ihrer Einstellung gegenüber Drohnen negativ beeinflusst. 74 Prozent der Deutschen fühlen sich durch Drohnen überhaupt nicht gestört. Das zeigt eine repräsentative Umfrage von TNS Emnid im Auftrag von Reichelt Elektronik.

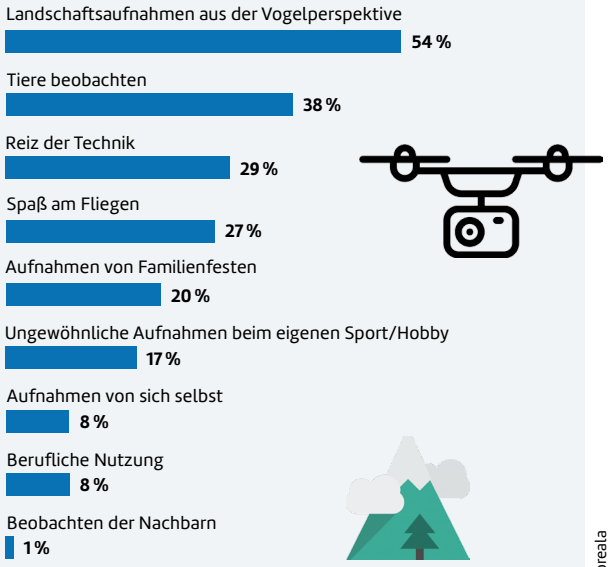
Multikopter lassen sich immer leichter handhaben, ihre Akkukapazität steigt und die Preise sinken. Das führt dazu, dass die Deutschen zunehmend über die Anschaffung einer Drohne nachdenken. Immerhin jeder zwölfte Umfrageteilnehmer kann sich vorstellen, in den nächsten zwölf Monaten eine Drohne anzuschaffen, oder plant es bereits.



Bei den verschiedenen Einsatzmöglichkeiten von Drohnen stehen Landschafts- und Tieraufnahmen ganz oben in der Gunst der Umfrageteilnehmer. Immerhin jeder Zwölfte kann sich auch eine berufliche Nutzung eines Multikopters vorstellen.

Die Sorge, dass Drohnen zum Beobachten der Nachbarn eingesetzt werden könnten, ist weitestgehend unbegründet: Nur ein Prozent der Befragten bekundet Interesse, sein Umfeld aus der Luft auszuspionieren. Bei den 14- bis 29-Jährigen besteht sogar überhaupt kein Interesse.

www.reichelt.de



Drei Viertel der Deutschen fühlen sich durch Drohnen nicht gestört

web & mobile developer 10/2016 Quelle: www.reichelt.de

Smartphone löst in Zukunft TV ab

Vor allem für die junge Generation ist das Smartphone unverzichtbar – TV und Radio spielen hier kaum eine Rolle. Das ergab eine repräsentative Umfrage durch TNS Emnid im Auftrag des Bundesverbands Digitale Wirtschaft (BVDW) e. V. unter 1004 Internetnutzern in Deutschland.

Fernseher und Smartphones sind die technischen Geräte, auf die die wenigsten verzichten möchten: Für jeden dritten Befragten (33 Prozent) ist das TV

Für BVDW-Vizepräsident Thorben Fasching ist das Smartphone aus unserem Leben nicht mehr wegzudenken

unentbehrlich – dicht gefolgt vom Smartphone mit 27 Prozent. Auf Platz drei liegt das Radio mit 20 Prozent. Ein radikal anderes Bild zeigt sich bei der jungen Generation: Für satte 67 Prozent der 14–29-Jährigen ist das Smartphone nicht mehr wegzudenken. Klassische Medien wie der Fernseher (acht Prozent) oder das Radio (fünf Prozent) spielen in dieser Generation kaum eine relevante Rolle. BVDW-Vizepräsident Thorben Fasching erklärt: »Das Smartphone ist aus unserem Leben nicht mehr wegzudenken. Dass über Jahrzehnte etablierte Medien wie TV und Radio in der jüngeren Generation kaum noch stattfinden und durch das Smartphone abgelöst werden, ist ein Sinnbild des digitalen Wandels. Natürlich heißt es nicht, dass Fernsehen und Radio dem Ende nah sind – vielmehr werden sich Übertragungsweg und Konsumart grundlegend verändern.«

Bei aller Begeisterung für Smartphones gibt es auch Schwächen zu beklagen. Die Top-3-Nachteile sind ein schwacher Akku (29 Prozent) sowie eine langsame Internetverbindung (24 Prozent) und Empfangsprobleme (24 Prozent). Das Design (zwei Prozent), die Größe (fünf Prozent) und eine schlechte Kameraqualität (neun Prozent) werden nur selten als Nachteil empfunden. Dabei nutzen viele Hersteller gerade die Güte der Kamera als Werbeargument für Neuanschaffungen. Nur neun Prozent der Befragten sind in dieser Hinsicht unzufrieden. Auch zu geringer Speicherplatz zählt nur bei 14 Prozent zu den größten Nachteilen.

www.bvdw.org

Foto: Shutterstock / Boreala

IBM Sicherheits-App

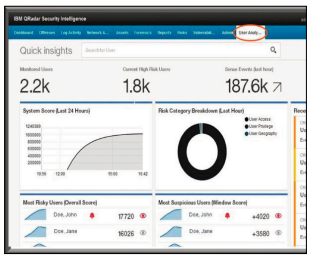
Schwachstellen in den eigenen Reihen

QRadar User Behavior Analytics heißt die neue App von IBM, mit der sich Organisation effektiver vor Cyberattacken aus den eigenen Reihen schützen können.

Das Add-on für das IBM Sicherheitsinformations- und Ereignismanagement QRadar erleichtert die Arbeit von IT-Abteilungen enorm, denn es schlägt bei verdächtigen Systemzugriffen automatisch Alarm. Insider sind für rund 60 Prozent aller Cyberangriffe auf Unternehmen verantwortlich. Dabei ist nicht immer böse Absicht am Werk, denn meist spielen Anwenderfehler den

kriminellen Hackern Passwörter von Mitarbeitern oder andere sensible Daten in die Hände.

Die neue User Behavior Analytics App klinkt sich in QRadar ein, das IBM Sicherheitsinformations- und Ereignismanagement, und warnt so vor verdächtigem Systemverhalten und davon ausgehenden Cyber-



Eine IBM-App schützt Organisationen vor Cyberattacken durch Insider

gefahren. So müssen IT-Abteilungen zur Analyse nicht Daten aus unterschiedlichen Quellen zusammenziehen, sondern haben Informationen zentral auf dem Schirm.

www.ibm.de

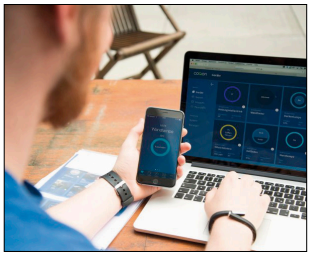
Umfrage zu Smart Home

Deutsche möchten ihr Zuhause intelligent vernetzen

74 Prozent würden künftig gerne in einem Smart Home nach ihren Vorstellungen leben. So das Ergebnis einer Studie des Marktforschungsinstituts YouGov im Auftrag des deutschen Smart-Home-Anbieters Coqon. Für 83 Prozent sind Datensicherheit und Datenschutz elementar. 79 Prozent erwarten, dass die Technik einfach und unkompliziert im Hintergrund läuft, ohne dass sie sich selbst damit auseinandersetzen müssen. Fast jeder Vierte (23 Prozent) ist bereit, mehr als 2000 Euro für den Smart-Home-Ausbau zu investieren. Dass die eigenen vier Wände smart sind – für 63 Prozent der Befragten wird dies bald so selbstverständlich sein wie der Umgang mit dem Smartphone.

Einen besonderen Fokus hat die YouGov-Studie auf die Einstellung der Geschlechter verschiedener Altersgruppen zum Thema Smart Home gelegt. Insbesondere Frauen zwischen 18 und 35 Jahren weisen eine überdurchschnittlich hohe Affinität zur intelligenten Vernetzung ihres Zuhauses auf. So würden 8 von 10 Frauen dieser Altersgruppe die smarte Technik nutzen und wesentlich aktiver als der durchschnittliche Deutsche mitentscheiden wollen, welche Bereiche in den eigenen vier Wänden intelligent vernetzt werden.

www.coqon.de



Die intelligente Vernetzung der eigenen vier Wände steht bei den Deutschen hoch im Kurs

Training

Know-how für Java-Entwickler

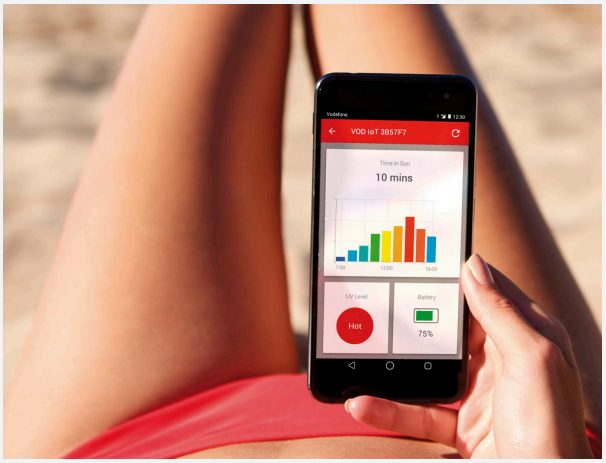
Bei professionellen Anwendungen und Android-Apps ist die Programmiersprache Java bei den meisten Entwicklern erste Wahl. Im Tiobe-Index liegt Java, konkurrierend mit C, stets auf Platz 1 oder 2 des Rankings. Wer in die Sprache neu einsteigen will oder als Fortgeschrittener sein Know-how vertiefen will, für den bietet Developer Media im November zwei Trainings an, die jeweils in Köln stattfinden.

Am 21. November kommen die Einsteiger auf ihre Kos- ▶

Vodafone

Der intelligente Bikini

Laut einer Online-Umfrage des Marktforschungsinstituts YouGov vergisst die Hälfte der befragten deutschen Urlauber, sich mit Sonnenschutz einzucremen. Jeder Vierte gibt zu, auch sein Kind nicht immer ausreichend gegen UV-Strahlen zu schützen. Um solche und ähnliche Probleme zu vermeiden, hat Vodafone das Projekt »Smart Summer« ins Leben gerufen: In Zukunft kann der Bikini der Trägerin Bescheid geben, wenn es Zeit für die Sonnencreme ist. Und wer sich genau wie zehn Pro-



Smarte Sommer-Accessoires können künftig bei Sonnenbrand und verlorenem Gepäck helfen

zent der Befragten über verlorenes Urlaubsgepäck ärgert, dem helfen künftig smarte Taschen, die dem Besitzer ihren Standort melden.

60 Prozent der befragten Urlauber gaben an, dass ihnen ein Warnsignal helfen würde, an Sonnencreme zu denken und die Intensität der Sonne besser einzuschätzen. Deshalb sind smarte Bikinis und Badehosen zukünftig mit Sensoren ausgestattet, die die UV-Strahlung messen. Wird die Belastung zu hoch, warnen App und ein kleiner vibrierender Chip in der Bademode den Nutzer, dass es besser wäre, sich mit Sonnencreme zu schützen oder den Schatten aufzusuchen. Für Kinder könnte es am Strand der Zukunft einen intelligenten Sonnenhut mit UV-Sensoren geben, der zusätzlich als Ortungsgerät fungiert. Entfernt sich das Kind aus einem individuell festgelegten Radius, meldet die App den Eltern einen Alarm und zeigt den Standort des Kindes an. Und auch verloren gegangene Gepäckstücke könnten schon bald der Vergangenheit angehören: Dank IoT-Technologie kann die Strandtasche zukünftig per Ortungs-Chip lokalisiert werden.

Auf kurzen Distanzen erfolgt die Ortung über die energieeffiziente Bluetooth-Technologie, bei größeren Entfernungen per GPS und GSM. Die Accessoires beinhalten eine Batterie, die circa eine Woche lang hält. Ende 2017 wird diese IoT-Hardware noch leistungsfähiger sein: Die Batterie wird nach einmaligem Laden bis zu zehn Jahre halten, klein und kostengünstig sein.

www.vodafone.com

web & mobile developer 10/2016 Quelle: www.vodafone.com

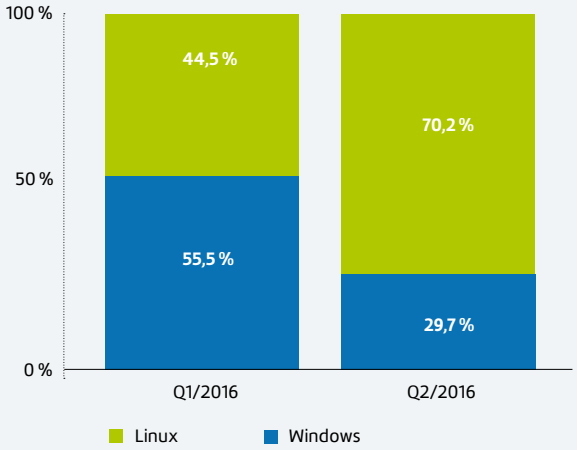
Kaspersky-Report

DDoS-Attacken über Linux-Server

Die von Kaspersky Lab zwischen April und Juni 2016 gemessenen DDoS-Attacken (Distributed Denial of Service) dauerten länger als zu Beginn des Jahres. Außerdem stellte der IT-Sicherheitsexperte einen signifikanten Anstieg bei über Linux-Botnetze durchgeführten DDoS-Attacken fest. Diese Ergebnisse gehen aus dem Kaspersky-Report »DDoS-Attacken im zweiten Quartal 2016« hervor.

Die Anzahl von DDoS-Angriffen stieg von April bis Juni 2016 an. Die beliebteste Methode zur Durchführung von DDoS-Attacken sind nach wie vor sogenannte SYN-DDoS-Attacken. Hierbei handelt es sich um Angriffe, die den Internetzugang, das Betriebssystem oder Host-Dienste belasten. Im Vergleich zum Vorquartal stiegen SYN-DDoS-Angriffe um das 1,4-Fache an und machten drei Viertel (76 Prozent) aller Angriffe aus. Der Grund: Der Anteil von Attacken auf Basis von Linux-Botnetzen hat sich fast verdoppelt und stieg auf 70 Prozent an. Linux-Botnetze gelten als das effektivste Mittel zur Durchführung von SYN-DDoS-Angriffen. Zum ersten Mal konnte die Kaspersky DDoS Intelligence ein solches Ungleichgewicht zwischen Aktivitäten von Linux-basierten im Vergleich zu Windows-basierten DDoS-Bots ausmachen.

»Auf Linux-Servern befinden sich oft verbreitete Schwachstellen, aber keine zuverlässige Sicherheitslösung. Das macht sie zu einem geeigneten Tool für Botnetzbetreiber«, sagt Oleg Kupreev, Lead Malware Analyst bei Kaspersky Lab. »Attacken über Linux-basierte Bots sind einfach, aber effektiv. Sie können wochenlang andauern, während die Server-Besitzer nichts davon mitbekommen, dass sie der Ursprung einer Attacke sind. Zudem sind Cyberkriminelle in der Lage, über einen einzigen Server einen Angriff durchzuführen, der die Stärke Hunderter einzelner Computer hat. Unternehmen sollten sich präventiv auf derartige Szenarien einstellen, indem sie zuverlässige Schutzlösungen gegen DDoS-Angriffe integrieren.«
https://de.securelist.com



Signifikanter Anstieg von DDoS-Attacken über Linux-Botnetze

Quelle: Kaspersky Lab

ten. Mit diesem Training wird ihnen eine Einführung in die objektorientierte Programmiersprache Java geboten. In zahlreichen praktischen Beispielen stellt der Trainer Alexander Salvanos die imperativen Sprachkonzepte wie auch die wichtigsten Entwurfsprinzipien der Objektorientierung unter Java vor. Hierzu zählen unter anderem

Von Alexander Salvanos stammt das Standardwerk für Java-EE-Technologien im deutschsprachigen Raum

Abstraktion, Kapselung, Vererbung und Polymorphie. Zusätzlich werden auch die Besonderheiten der wichtigsten vordefinierten Klassen der Java-Standard-Bibliothek erklärt. Das Training ist als Schnelleinstieg für Einsteiger in die Java-Programmierung konzipiert.

Das Training am 28. November richtet sich an die Profis. Ob Abstraktion, Kapselung, Vererbung oder Polymorphie – jeder Entwickler, der die Objektorientierung einmal verstanden hat, fühlt sich bald sattelfest. Um aber das Potenzial von Java voll auszuschöpfen, reichen Grundlagen nicht aus – das Wissen um Details der Sprache ist unabdingbar: Wie lassen sich Typparameter mit *super* und *extends* einschränken, wie werden sie erweitert? Wie ersetzt man alten Java-Code durch Lambda-Ausdrücke? Welche funktionalen Schnittstellen hat das API vordefiniert? Wann übergibt man eine Method-Reference als Parameter? Wie werden Streams verwendet? Und wie funktioniert hierbei das Multi-Threading?

Diese und viele weitere Fragen beantwortet Alexander Salvanos in diesem Training, das sich an all diejenigen Java-Entwickler richtet, die die Sprachspezifika kennenlernen möchten, um Java künftig optimal einsetzen zu können.

Alexander Salvanos ist Mitglied der Java EE Expert Group und seit 1998 auf Java-Technologien spezialisiert. Neben seinen Tätigkeiten als IT-Berater und zertifizierter Java-EE-Trainer (IHK) ist er auch der Autor des Buches »Professionell entwickeln mit Java EE 7«, dem Standardwerk für Java-EE-Technologien im deutschsprachigen Raum.
http://developer-media.de

Facebook

Werbung trotz Adblocker

Facebook will künftig auch an User mit Werbeunterdrückern Anzeigen ausspielen. Auch wenn Facebook nicht so stark von der Adblocker-Debatte betroffen ist, muss es sich dennoch mit der Frage auseinandersetzen, wie sich die Nutzer wieder mit Werbung anfreunden können. Dafür hat das Unternehmen nun einige Neuerungen in einem Blogpost vorgestellt. Spannend ist vor allem das Update, mit dem das Blockieren bestimmter Werbeanzeigen umgangen werden soll. Facebook will Personen, die Software zur Werbeanzeigenblockierung nutzen, auf dem Desktop dennoch Ads auf der Facebook-Seite ausspielen.
www.facebook.com

PROGRAMMIERUNG DES ARDUINO/GENUINO 101

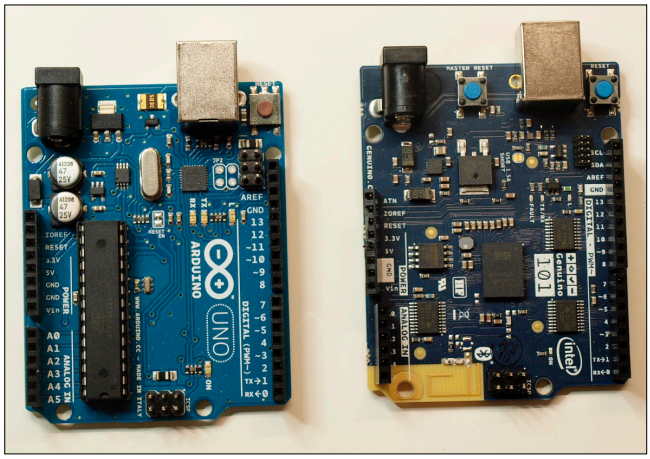
Arduino 101 an Android

Intels neuer Arduino erleichtert Entwicklern das Realisieren von IoT-Geräten.

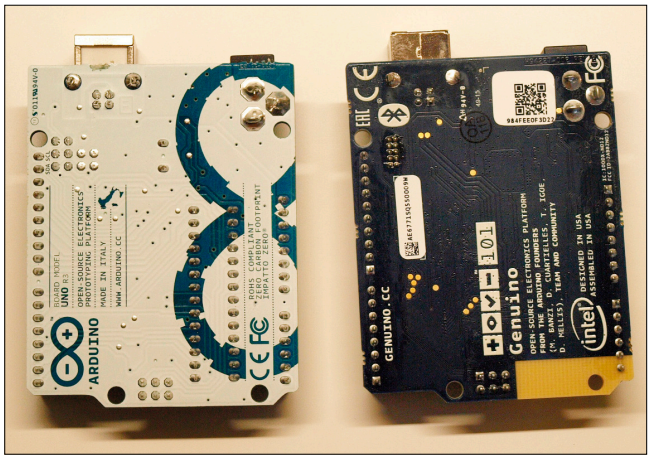
Bluetooth-Verbindungen zwischen Arduino und Smartphone sind per se keine Neuigkeit. Der jüngste Spross aus dem banzischen Teil der Arduino-Familie unterscheidet sich von seinen Vorgängern insofern, als er ein Bluetooth-LE-Modul mitbringt.

Die Gefechte zwischen den einst vereinten Arduino-Brüdern involvieren mittlerweilen auch Chiphersteller. Während Intel die Mannen um Massimo Banzi versorgt, zeigen sich die nobleren Herren aus dem Hause STM eher dem Team um Frederico Musto zugewandt.

Wie dem auch sei, der Arduino 101 enthält einen der neuen Curie-Zweikernprozessoren aus dem Hause Intel, die eine x86-CPU und einen RISC-Prozessor miteinander koppeln.



Die Steckplätze sind da, wo man sie vermuten würde (Bild 1)



Die Antenne sollte zur Optimierung der Reichweite mindestens 10 cm von metallischen Flächen entfernt bleiben (Bild 2)

Vom Aussehen her besteht zwischen dem 101 und seinem Vorgänger Arduino Uno nur wenig Unterschied: Bild 1 zeigt beide Geräte von oben, während Bild 2 die Unterseite zeigt.

Das Deployment der notwendigen Software erfolgt wie gewohnt: Laden Sie die unter https://www.arduino.cc/en/Main/Software bereitstehende IDE herunter und installieren Sie diese auf einer Windows-Workstation. Beachten Sie dabei, dass die Version von Arduino.org wahrscheinlich nicht mit dem 101 funktioniert.

Beide Arduino-Teams liefern die IDE seit einiger Zeit nur mehr mit Unterstützung für einige wenige MCU-Familien aus. Die Toolchains für seltener verwendete Prozessrechner müssen nach dem Download über einen von Microchip beigesteuerten Downloadmanager bezogen werden. Klicken Sie dazu auf *Tools, Board:* * und *Boards Manager*. Suchen Sie im daraufhin erscheinenden Dialog nach dem String *101*. Nach getaner Installation präsentiert sich der Bildschirm wie in Bild 3 gezeigt. Die Combobox erlaubt das gezielte Herunterladen bestimmter Versionen der Toolchain.

An dieser Stelle ein kleiner Hinweis: Auf vielen Computern funktioniert nur die Version 1.0.4 der Toolchain. Ältere und neuere Varianten werfen beim Brennen von Images Fehler. Wenn jedoch eine neuere Variante in Ihrer Systemkonfiguration funktioniert, so können Sie sich die im folgenden Abschnitt beschriebene manuelle Integration der Bibliotheken ersparen.

Elektronische Spielerei

Aus prozessrechentechnischer Sicht ist der Arduino 101 im Großen und Ganzen ein Arduino wie jeder andere. Die Signalspannung von 3,3 Volt kennt man von Zero oder Due, während der beim Ausgeben schneller Rechteckswellenformen auftretende Jitter von den Galileos bekannt ist.

Leistungselektroniker sollten wissen, dass die PWM-Signale hier durch eine Zeitgebereinheit entstehen. Wer dieses Modul für eine andere Funktion einspannt, muss sich nicht darüber wundern, wenn die PWM-Signale zu schwingen aufhören.

Sowohl der Bluetooth-Transmitter als auch das Gyroskop werden am 101 durch spezielle Bibliotheken ansprechbar gemacht, die zum Zeitpunkt der Drucklegung folgende Namen aufweisen:

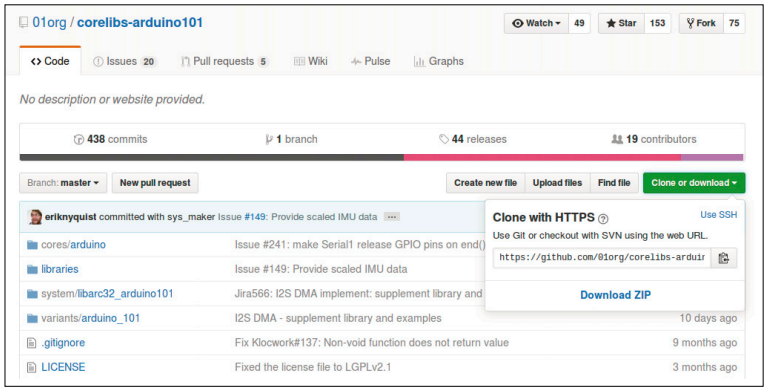
- CurieBLE,
- CurieIMU,
- CurieTimerOne.

Leider enthält die an sich stabil funktionierende Version 1.0.4 des Compilerkits einen Fehler, der die Bibliotheken nicht au-

Die Toolchain für den Arduino 101 ist nicht von Haus aus verfügbar (Bild 3)

tomatisch einbindet. Zur Behebung dieses Problems müssen Sie im ersten Schritt den URL https://github.com/01org/corelibs-arduino101 aufrufen, um das dort befindliche Repository durch Anklicken des in Bild 4 gezeigten Buttons herunterzuladen.

Im nächsten Schritt müssen Sie die Bibliotheken in die Arduino-IDE integrieren. Im Unterordner *corelibs-arduino101-master\libraries* finden sich die diversen Bibliotheken. Wir gehen in den folgenden Schritten davon aus, dass Sie *CurieTimerOne* integrieren wollen. Wechseln Sie in das Unterverzeichnis und zippen Sie alle dort befindlichen Dateien. Wechseln Sie sodann in die Arduino-IDE zurück und laden Sie das Archiv über *Sketch, Bibliothek* einbinden und *.ZIP-Bibliothek hinzufügen*.



Unter diesem Button finden Sie in GitHub das ZIP-Archiv (Bild 4)

Zahlenausgabe

Als erstes Beispiel wollen wir uns der Ansteuerung einer Siebensegmentanzeige zuwenden. Das in Bild 5 gezeigte Bauteil ist im Elektronikfachhandel vergleichsweise preisgünstig erhältlich und lässt sich über die in Bild 6 gezeigte Gruppe von Widerständen mit dem Arduino 101 verbinden. Auf der Softwareseite beginnt unser Sketch mit der Einbindung der Bluetooth-Bibliothek *CurieBLE*. Der Arduino 101 unterscheidet sich von seinen Vorgängern insofern, als er durch ein vergleichsweise komplexes Echtzeitbetriebssystem betrieben wird. Daraus folgt eine längere Startzeit des Prozessrechners, die wir im Rahmen des Tests dieses Programms messtechnisch ermitteln wollen:

```
void setup() {
  pinMode(13, OUTPUT);
  pulseLED();
```

Nach der Initialisierung des Ausgangspins nutzen wir *pinMode* abermals, um die für die Ansteuerung der Siebensegmentanzeige notwendigen Pins zu initialisieren:

```
  for(int i=2;i<=8;i++)
  {
    pinMode(i, OUTPUT);
  }

  //BLE-Infrastruktur hochbringen
  ...
  pulseLED();
}
```

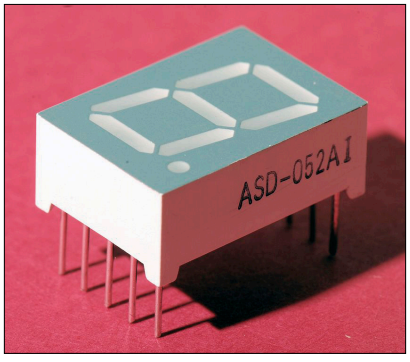
An dieser Stelle ein kleiner Hinweis: Beim Testen von Microcontrollern ist es immens hilfreich, wenn man einen überschüssigen PIN als Ausgabemedium für Statusinformationen nutzt.

Die Methode *pulseLED()* ändert den Status des Ausgangspins, was sowohl das Triggern eines Oszilloskops als auch Rückschlüsse auf den Zustand des Mikrocontrollers erlaubt:

```
void pulseLED()
{
  digitalWrite(13, !digitalRead(13));
}
```

Damit können wir uns dem Rest des Setups zuwenden. Neben dem Konfigurieren der Pins – der Arduino startet wie alle Prozessrechner inert – müssen wir auch die für die Kommunikation mit dem Smartphone erforderliche Bluetooth-LE-Infrastruktur hochbringen:

```
void setup() {
  ...
  //BLE-Infrastruktur hochbringen
  blePeripheral.setLocalName
  ("NMGOutput");
  blePeripheral.
  setAdvertisedServiceUuid
  (ledService.uuid());
  blePeripheral.
  addAttribute(ledService);
  blePeripheral.addAttribute(
```



Siebensegmentanzeigen geben die Zahlen 0 bis 9 – und mit etwas Fantasie die Buchstaben A bis F – aus (Bild 5)

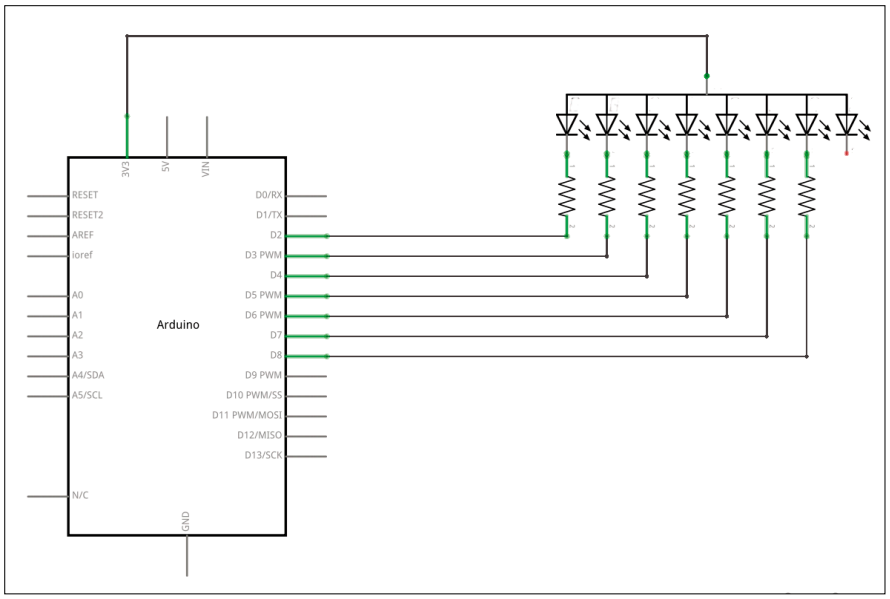
```
  switchCharacteristic);
  switchCharacteristic.setValue(0);
  blePeripheral.begin();
  pulseLED();
}
```

Damit fehlt uns an dieser Stelle nur noch der Programm-Header, der neben der weiter oben nicht abgedruckten Einbindung der Bibliothek auch die Speicherbereiche für die diversen Objekte deklariert. Services und Charakteristika bekommen an dieser Stelle zudem ihre GUIDs eingeschrieben, die sie gegenüber sonstiger Hardware eindeutig ausweisen:

```
#include <CurieBLE.h>

BLEPeripheral blePeripheral;
BLEService ledService
("19B10000-E8F2-537E-4F6C-D104768A1214");
BLEUnsignedCharCharacteristic switchCharacteristic
("!19B10001-E8F2-537E-4F6C-D104768A1214",
BLERead | BLEWrite);
```

Der von den jeweiligen Charakteristiken verwaltete Datentyp wird durch Auswahl der *BLECharacteristic*-Unterklasse festgelegt. Zum Zeitpunkt der Drucklegung unterstützt der Arduino 101 folgende Charakteristika:

- BLEBoolCharacteristic (UUID, properties),
- BLECharCharacteristic (UUID, properties),
- BLEUnsignedCharCharacteristic (UUID, properties),
- BLEShortCharacteristic (UUID, properties),
- BLEUnsignedShortCharacteristic (UUID, properties),
- BLEIntCharacteristic (UUID, properties),
- BLEUnsignedIntCharacteristic (UUID, properties),
- BLELongCharacteristic (UUID, properties),
- BLEUnsignedLongCharacteristic (UUID, properties),



Die Widerstände schützen die LEDs vor zu viel Strom – und die Pins des Arduinos vor zu geringer Impendanz (Bild 6)

Listing 1: Loop-Methode

```
void loop() {
  BLECentral central = blePeripheral.central();
  if (central) {
    while (central.connected()) {
      if (switchCharacteristic.written()) {
        char was=switchCharacteristic.value();
        switch(was)
        {
          case '0':
            schreibeLED("abcdef");
            break;
          case '1':
            schreibeLED("bc");
            break;
        }
```

- BLEFloatCharacteristic (UUID, properties),
- BLEDoubleCharacteristic (UUID, properties).

Das *Properties*-Feld beschreibt die Art der anzulegenden Charakteristik. Der Arduino 101 kennt hier die Werte *BLERead, BLEWrite* und *BLENotify*. In der Dokumentation findet sich manchmal auch ein Verweis auf einen Parameter *maxLen*, der von typisierten Instanzen der *BLECharacteristic*-Klasse nicht entgegengenommen wird. Untypisierte Charakteristika stellen einen Sonderfall dar, den wir am Ende des Artikels näher betrachten.

Datenaustausch in loop()

Die Kommunikation mit angeschlossener Hardware erfolgt in der *Loop*-Methode. Am Beginn jedes Durchlaufs erfolgt ein Aufruf von *central()*.

Wenn der Arduino mit einem Central verbunden ist, so wird eine darauf verweisende Instanz der *BLECentral*-Klasse zurückgeliefert. Wenn diese mit dem Arduino verbunden ist, treten wir in eine innere Schleife ein. Diese Schleife prüft, ob die Schaltcharakteristik mit einem neuen Wert beschrieben wurde. Ist dies der Fall, so wird das angelieferte Zeichen weiterverarbeitet (Listing 1).

Beim Schreiben von Siebensegmentanzeigen empfiehlt es sich, die Arbeit in mehrere Schritte aufzuteilen. Unser Programm überprüft im ersten Schritt mittels einer Selektion, welche LEDs einzuschalten sind.

Diese Informationen wandern sodann in einen String, der an eine dafür spezialisierte Methode weitergereicht wird. Der Aufbau ebendieser Funktion

ist vergleichsweise einfach. Wir schalten im ersten Schritt alle Dioden aus:

```
#define AN LOW
#define AUS HIGH
void schreibeLED(char* _x)
{
  for(int i=2;i<=8;i++)
  {
    digitalWrite(i, AUS);
  }
```

Im nächsten Schritt durchlaufen wir den String Zeichen für Zeichen und aktivieren die jeweilige LED. Der in Listing 2 abgedruckte Code zeigt aus Platzgründen nur die Vorgehensweise für a bis c – die restlichen Chars verhalten sich analog.

Puristen wundern sich an dieser Stelle, warum An und Aus vom Autor durch Makros realisiert wurden. Dies liegt daran, dass es Siebensegmentanzeigen in zwei Varianten gibt: Neben der hier verwendeten Variante mit gemeinsamer Anode gibt es auch Siebensegmentanzeigen mit gemeinsamen Kathoden.

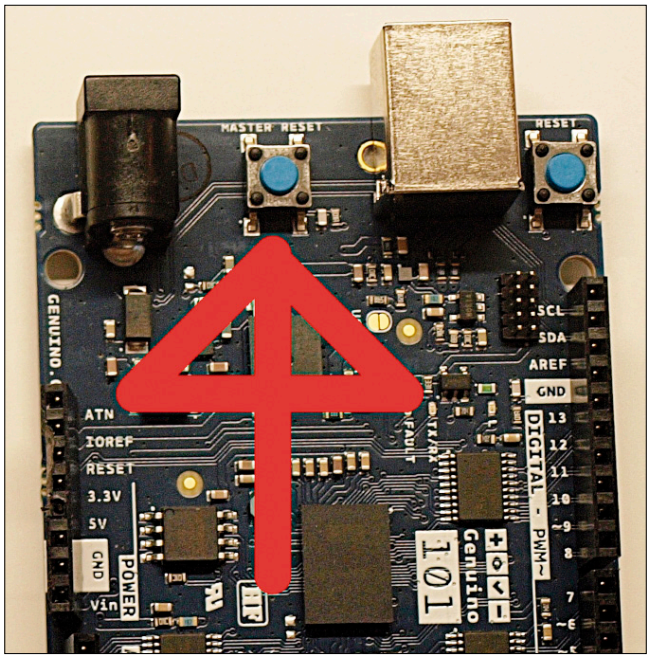
Durch die Nutzung einer Compilerkonstante kann man im Fall von Schwierigkeiten bei der Beschaffung eines bestimmten Bauteiltyps leicht zwischen Common Anode und Common Cathode hin- und herwechseln: Flexibilität, die sich im Zweifelsfall mehr als auszahlt.

Test des Arduino-Teils

Damit sind wir zum Test des Arduino-Teils bereit. Die praktische Erfahrung lehrt, dass es vernünftig ist, wenn man mehrteilige Kommunikationssysteme Schritt für Schritt überprüft. Für einfache BTLE-Kommunikationsszenarien bietet sich die unter https://play.google.com/store/apps/details?id=no.

Listing 2: Aktivieren der LED

```
while((*_x)!=NULL)
  {
    switch(*_x)
    {
      case 'a':
        digitalWrite(2, AN);
        break;
      case 'b':
        digitalWrite(3, AN);
        break;
      case 'c':
        digitalWrite(4, AN);
        break;
      ...
    }
    _x++;
  }
}
```



Durch Drücken der Master Reset-Taste wird der Prozessrechner für Kommandos aufnahmefähig (Bild 7)

nordicsemi.android.mcp&hl=en zum Download bereitstehende App *nRF Connect for Mobile* an, die das Vorhandensein des Sketches am Prozessrechner voraussetzt.

Aufgrund der vergleichsweise komplexen Architektur braucht der Arduino 101 beim Installieren neuer Programme etwas Hilfe. Drücken Sie die in Bild 7 gezeigte Taste *Master Reset*, um den Prozessrechner zum Annehmen neuer Software zu animieren.

Bei der Arbeit mit Ubuntu kommt es beim Deployment gern zu Fehlern der Bauart *dfu-util: Cannot open DFU device 8087:0aba*. Diese lassen sich durch Ausführung des *udev*-Generatorskripts beheben. Auf der Workstation des Autors sieht die dazu notwendige Befehlsfolge so aus:

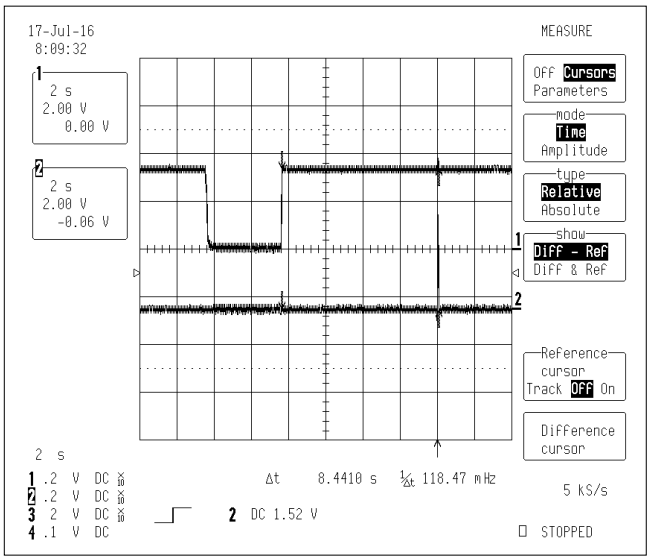
```
tamhan@tamhan-thinkpad:~/.arduino15/packages/Intel/
tools/arduino101load/1.6.9+1.24/scripts$ sudo ./
create_dfu_udev_rule

[sudo] password for tamhan:
tamhan@tamhan-thinkpad:~/.arduino15/packages/Intel/
tools/arduino101load/1.6.9+1.24/scripts$
```

Starten Sie *nRF Connect* nach dem erfolgreichen Deployment auf Ihrem Smartphone. Im Rahmen des sehr schnell erfolgenden Scanprozesses von Bluetooth LE sollte die App Ihren Prozessrechner unter dem per *setLocalName* festgelegten Namen erkennen.

Nach dem Anklicken des *Connect*-Buttons listet das Programm die vom Arduino 101 bereitgestellten Dienste auf. Ein Tap auf einen Eintrag öffnet die jeweilige Characteristic, die sich über den nach oben zeigenden Eintrag einen beliebigen Wert einschreiben lässt. Die zum Testen ideal geeignete Befehlsfolge ist in Bild 8 gezeigt. ▶

Die Ermittlung der Start-up-Zeit erfolgt mit einem Zweikanal-Oszillografen. Verbinden Sie den Triggereingang mit der Spannungsversorgung, während Pin 13 an einen Nicht-Triggerkanal kommt.

Stecken Sie den Prozessrechner sodann bei einer sehr langsamen Zeitbasis an, aktivieren Sie *Peak Detect* und warten Sie, bis der Oszillograf das Ansteigen der Spannung am Pin 13 erfasst. Im nächsten Schritt können Sie die Startzeit dann wie in Bild 9 gezeigt mit Cursoren messen.



Der Arduino 101 lässt sich beim Kaltstart rund acht Sekunden Zeit (Bild 9)

Kommunikatives Android

Nach diesem grundlegenden Funktionstest wollen wir uns nun der Realisierung eines nativen Clients zuwenden. Erzeugen Sie ein neues Projektskelett auf Basis einer halbwegs aktuellen Android-Version und passen Sie die Manifestdatei an:

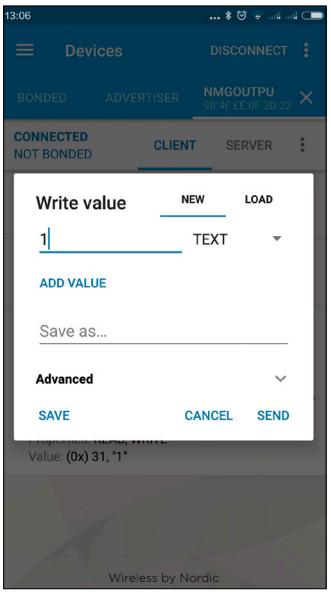
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
"http://schemas.android.com/apk/res/android"
package="com.tamoggemon.nmgsevensegmentdriver">

  <uses-permission android:name=
  "android.permission.BLUETOOTH"/>
  <uses-permission android:name=
  "android.permission.BLUETOOTH_ADMIN"/>
  <uses-feature android:name=
  "android.hardware.bluetooth_le"
  android:required="true"/>
```

Neben der Deklaration der beiden für Bluetooth erforderlichen Capibilities setzen wir hier auch das Vorhandensein von Bluetooth-LE-Hardware voraus. Es ist auch heute noch nicht gegeben, dass jedes im Play Store befindliche Telefon mit Bluetooth LE umgehen kann. Im nächsten Schritt müssen wir die *MainActivity* mit den zum Suchen notwendigen Objekten ausstatten. Als Erstes beschaffen wir einen Verweis auf den Bluetooth-Manager, der sodann um einen Bluetooth-Adapter erleichtert wird:

```
public class MainActivity extends
AppCompatActivity  implements
BluetoothAdapter.LeScanCallback{
  BluetoothManager myBluetoothManager;
  BluetoothAdapter myBluetoothAdapter;
  @Override
  protected void onCreate(Bundle
  savedInstanceState) {
    ...
    myBluetoothManager =
    (BluetoothManager) getSystemService
    (Context.BLUETOOTH_SERVICE);
    myBluetoothAdapter =
    myBluetoothManager.getAdapter();
```

An dieser Stelle können wir prüfen, ob die Funkverbindung aktiviert ist. Ist dies der Fall, so weisen wir den Bluetooth-Adapter dazu an, nach Bluetooth-LE-Geräten in der Umgebung zu suchen:

```
public class MainActivity ...{
  ...
  if (myBluetoothAdapter != null &&
  myBluetoothAdapter.isEnabled()) {
    UUID[] myUUID=new UUID[1];
    myUUID[0]=UUID.fromString
    ("19B10000-E8F2-537E-4F6C-D104768A1214");
    myBluetoothAdapter.startLeScan(myUUID,this);
  }
}
```



Mit diesen Parametern bringen Sie eine 1 auf den Bildschirm (Bild 8)

Hier sind zwei Dinge interessant: Erstens ermöglicht uns das Übergeben des UUID eines Services das Einschränken der Suchergebnisse. Android liefert in diesem Fall nur jene Geräte zurück, die den passenden Service auch implementieren. Zweitens könnten wir beim Nicht-Eingeschaltetsein des Radios einen Dialog aufpoppen, der Nutzer zur Aktivierung animiert. Der dazu notwendige Code würde einen Intent ans System schicken, der folgendermaßen aufgebaut ist:

```
if (!mBluetoothAdapter.isEnabled()) {
  Intent enableBtIntent = new Intent
  (BluetoothAdapter.
  ACTION_REQUEST_ENABLE);
  startActivityForResult
  (enableBtIntent, REQUEST_ENABLE_BT);
}
```

Per *StartLeScan* aufgerufene Suchprozesse setzen die Implementierung eines LE-Scan-

Callbacks voraus. Wir platzieren ihn aus Bequemlichkeitsgründen direkt in der *MainActivity*, was die Implementierung einer weiteren Funktion voraussetzt:

```
public class MainActivity ...
implements  BluetoothAdapter.LeScanCallback{
  ...
  @Override
  public void onLeScan(BluetoothDevice device,
  int rssi, byte[] scanRecord) {
  ...
  }
}
```

Eine vollständige Implementierung der Funktion, die beim Finden eines Geräts aufgerufen wird, sieht folgendermaßen aus:

```
@Override
public void onLeScan(BluetoothDevice device, int rssi,
byte[] scanRecord) {
  myBluetoothAdapter.stopLeScan(this);
  myCallback=new NMGGattCallback();
  myGatt = device.connectGatt(this, false,
  myCallback);
  myCallback.myGatt=myGatt;
  myGatt.connect();
}
```

Bluetooth LE unterscheidet sich vom klassischen Bluetooth durch den extrem schnellen Scanprozess. Der für seine defensive Programmierweise bekannte Bluetooth SIG-Manager Martin Wooley zeigt auf Kongressen gerne Applikationen vor, die nur eine Sekunde lang scannen. Im Fall von Android ist dies insofern ärgerlich, als ein weiterlaufender Scan ein und dasselbe Gerät immer wieder findet – ein Problem, dem sich durch Aufrufen von *stopLeScan* abhelfen lässt.

Listing 3: Einfügen der Buttons

```
@Override
public void onClick(View v) {
  byte c[]=new byte[1];
  if(v==findViewById(R.id.button1)){
    c[0]='1';
  }
  else if(v==findViewById(R.id.button2)){
    c[0]='2';
  }
  else if(v==findViewById(R.id.button3)){
    c[0]='3';
  }
  myCallback.myChara.setValue(c);
  myGatt.writeCharacteristic(myCallback.myChara);
}
```

Im nächsten Schritt erzeugen wir eine neue Instanz einer *Callback*-Klasse und weisen das Betriebssystem dann zum Aufbau einer Verbindung zwischen Arduino 101 und Smartphone an. Die einfachste Version des NMGGattCallbacks präsentiert sich folgendermaßen:

```
public class NMGGattCallback extends
BluetoothGattCallback {
  public BluetoothGatt myGatt;
  @Override
  public void onConnectionStateChange
  (BluetoothGatt gatt, int status, int newState) {
    super.onConnectionStateChange(gatt, status,
    newState);
    if (newState ==
    BluetoothProfile.STATE_CONNECTED) {
      myGatt.discoverServices();
    }
  }
```

Neben einer öffentlichen Variablen, die eine Instanz auf Bluetooth-Gatt entgegennimmt, findet sich hier auch die Methode *OnConnectionStateChanged*, die beim Auf- oder Abbau einer Verbindung aufgerufen wird. Wir rufen an dieser Stelle den Befehl *DiscoverServices* auf, der Android zum Finden aller an Arduino101 angebotenen Dienste animiert.

Beim erfolgreichen Abschluss der *ServiceDiscovered*-Routine wird der Callback *onServicesDiscovered* aufgerufen. Wir beschaffen in ihm einen Verweis auf die *BluetoothGattService*-Klasse unseres LED-Dienstes, die im nächsten Schritt um die betreffende Charakteristik erleichtert wird:

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt, int
status) {
  super.onServicesDiscovered(gatt, status);
  BluetoothGattService aService=gatt.getService(UUID.
  fromString("19B10000-E8F2-537E-4F6C-D104768A1214"));
  myChara=aService.getCharacteristic
  (UUID.fromString("19B10001-E8F2-537E-4F6C-
  D104768A1214"));
}
```

An dieser Stelle müssen Sie mehr oder weniger Buttons in die *MainActivity* einfügen. Unser Beispiel beschränkt sich auf die Implementierung der Knöpfe eins bis drei, die wie in Listing 3 gezeigt mit dem Event Handler zu verdrahten sind.

OnClick beginnt damit, die zu übertragende Payload zu übermitteln. Im nächsten Schritt wird die Charakteristik geschrieben. Android setzt dazu einen zweistufigen Arbeitsprozess voraus: Im ersten Schritt erfolgt ein Aufruf von *SetValue*, der den Lokalwert der Charakteristik aktualisiert. Die eigentliche Übertragung erfolgt durch einen Aufruf der in *myGatt* befindlichen Methode *WriteCharacteristic*, die zudem in einem Callback über Erfolg oder Misserfolg informiert. Damit ist die erste Version unsesres nativen Clients einsatzbereit. Bild 10 zeigt, wie das Programm in Aktion aussieht. ▶

Bluetooth LE unterscheidet sich insofern vom klassischen Bluetooth, als das Protokoll eine klare Rollenverteilung vorsieht: Der Server (Peripheral) stellt den KV-Speicher bereit, der vom Client (Center) ausgelesen wird.

Da eines der Ziele von Bluetooth LE die konsequente Reduktion des Stromverbrauchs ist, gilt permanentes Polling als ernergieineffizient. Da es allerdings immer wieder Situationen gibt, in denen ein Peripheral Informationen zum Center schicken muss, wurde von der Bluetooth-SIG ein Notification-Konzept implementiert. Dieses ermöglicht dem Center das Anmelden von Interesse an betreffenden Informationen, die im nächsten Schritt vom Peripheral automatisch angeliefert werden.

Zur Implementierung eines Notification Services müssen wir das bisher errichtete Prozessrechenprogramm anpassen. Der erste Unterschied ist die andere Deklaration der Characteristic, die nun mit dem Parameterpaar *BLERead* und *BLENotify* auszustatten ist:

```
BLEService dataStreamService
("19B10000-D1F2-537E-4F6C-D104768A1214");
BLEUnsignedCharCharacteristic streamCharacteristic
("19B10001-D1F2-537E-4F6C-D104768A1214",
BLERead | BLENotify);
```

Als erster Test bietet sich die Ermittlung der maximalen Übertragungsgeschwindigkeit an. Dazu erstellen wir zwei globale Variablen: Die eine realisiert einen von 0 bis 255 weiterlaufenden Zähler, während die zweite für das periodische Ein- und Ausschalten des mit einem Oszillografen zu verbindenden Pin 13 dient:

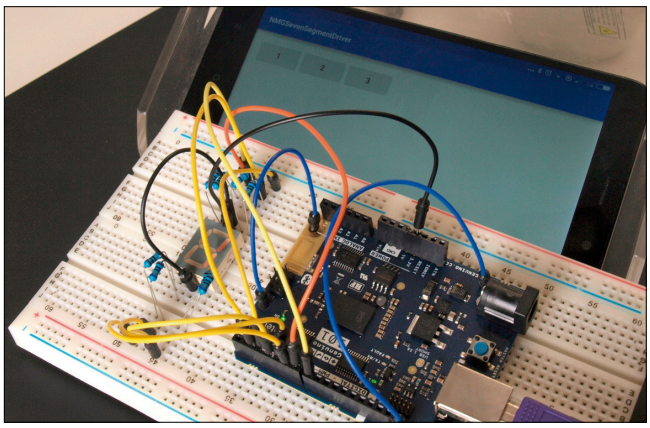
```
char globCounter;
bool toggler;
```

Der eigentliche Methodenkorpus von *loop()* ist dann primitiv: Der Wert des Pins wird im ersten Schritt geändert. Danach schreiben wir den aktuellen Wert in die Characteristic, um ihn in Richtung des Centers zu senden:

```
void loop() {
  globCounter++;
  toggler=!toggler;
  digitalWrite(13, toggler);
  streamCharacteristic.setValue(globCounter);
}
```

Zum Test ist nun eine Gegenstelle erforderlich, die die eingehenden Notifications verarbeitet und auf Periodizität der Werte prüft. Sind die entgegengenommenen Informationen keine klare Sequenz, so wurden nicht alle Änderungen übertragen. Die erste Änderung zum vorigen Beispiel ist die Anpassung der GUID. Wer die alte GUID am Arduino weiterverwendet, muss den folgenden Block nicht übernehmen:

```
protected void onCreate(Bundle savedInstanceState) {
  ...
```



Wer den Knopf 1 drückt, sieht die entsprechende Ziffer auf der Siebensegmentanzeige (Bild 10)

```
if (myBluetoothAdapter != null &&
myBluetoothAdapter.isEnabled()) {

  UUID[] myUUID=new UUID[1];
  myUUID[0]=UUID.fromString
  ("19B10000-D1F2-537E-4F6C-D104768A1214");
  ...
```

Zudem sind die gleichen Anpassungen auch an anderen Teilen des Programms notwendig. Für den kommerziellen Einsatz vorgesehener Code würde an dieser Stelle ein globales GUID-Halterobjekt verwenden.

Im nächsten Schritt muss die Charakteristik als *für Notifications* interessant erklärt werden. Dies erfolgt abermals in *NMGGattCallback*, wo die Methode *onServiceDiscovered* folgendermaßen angepasst wird:

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt,
int status) {
  ...
  myChara=aService.getCharacteristic
  (UUID.fromString("19B10001-D1F2-537E-4F6C-
  D104768A1214"));
  myGatt.setCharacteristicNotification(myChara,true);
  UUID uuid = UUID.fromString("00002902-0000-1000-
  8000-00805f9b34fb");
  BluetoothGattDescriptor descriptor =
  myChara.getDescriptor(uuid);
  descriptor.setValue
  (BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
  gatt.writeDescriptor(descriptor);
}
```

Googles API enthält eine böse Falle: Das Aufrufen von *setCharacteristicNotification* reicht nicht aus, um Ereignisse auch wirklich entgegenzunehmen. Stattdessen ist zusätzlich das Beschreiben einer von der Bluetooth SIG spezifizierten Characteristic erforderlich, die den Bluetooth-Stack des Zielgeräts zum Anliefern der Daten animiert.

Damit können wir uns mit dem Entgegennehmen der Informationen befassen. Android ruft beim Eingehen von Daten die Methode *onCharacteristicChanged* auf, die bequemerweise gleich mit einer Instanz des für den Aufruf verantwortlichen *GattCharacteristic*-Objekts ausgestattet wird:

```
@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic) {
  byte[] myBytes = characteristic.getValue();
  myDataStore[myCounter] = myBytes[0];
  myCounter++;
  if (myCounter == 512)
  {
    myCounter=0;
  }
}
```

Die erste Version unserer Routine sammelt 512 Werte in einem Byte-Array, um die Daten nach dem erfolgreichen Sampling bereitzustellen. Zum Abernten reicht es aus, einen Breakpoint in die Selektion zu setzen. Die Resultate lassen sich dann wie in Bild 11 gezeigt im Debugger ablesen.

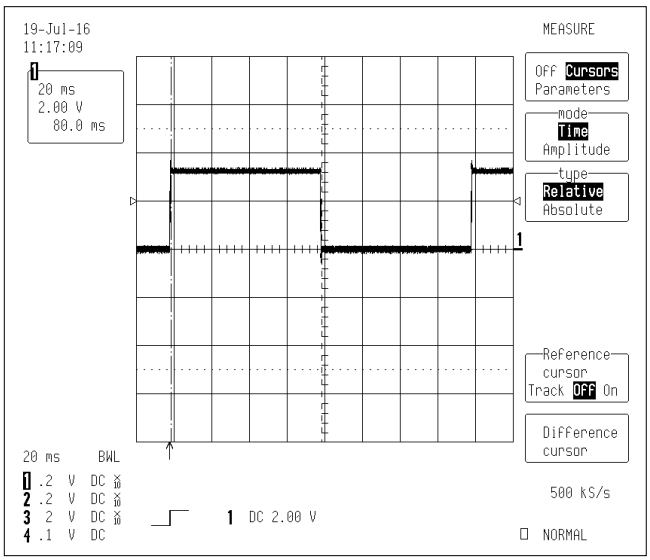
Damit stellt sich die Frage nach der Kommunikationsgeschwindigkeit. Dank des Togglings von Pin 13 können wir dies durch Anschließen eines Oszilloskops beantworten. Bild 12 zeigt, dass unser Arduino pro Sekunde rund 12 Datenpakete in Richtung des Telefons senden kann.

Der Beweis für das synchrone Arbeiten des Bluetooth-LE-Stacks lässt sich durch das Neustarten des Prozessrechners erbringen. Ist kein anderes Endgerät mit dem Bluetooth-Transmitter verbunden, so sinkt die Umschaltrate auf rund 5 Millisekunden. Der Arduino arbeitet schneller, weil er ja nun keine Daten übertragen muss.

Cambridge Silicon Radio spricht in Präsentationen zum Standard (siehe zum Beispiel https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=227336) gern von Datenraten im Bereich von 260 kBit/s. Es handelt sich dabei um eine sehr optimistische Einschätzung, die in der Praxis so gut wie nie erreicht wird. Entwickler berichten an verschiedenen Stellen von Paketraten im Bereich von zwischen 20 und 100 Millisekunden. Zum Erreichen ersterer Werte ist meist optimierte Hardware erforderlich.

Eventgetriebene Kommunikation

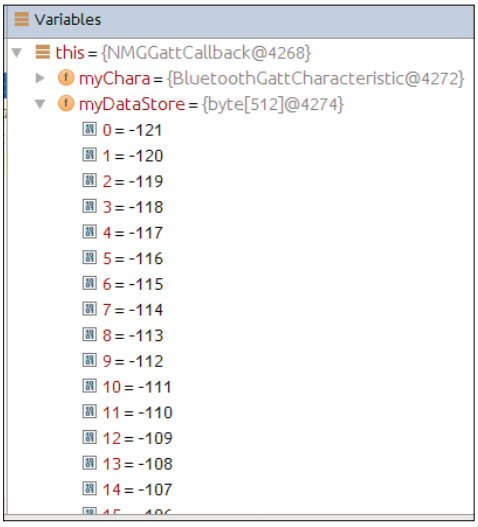
Endlosschleifen sind nicht der Weisheit letzter Schluss. Während das Belegen von *Main* für die hier realisierten Aufgaben kein Problem darstellt, ist es immer wieder wünschenswert, stattdessen auf eine eventgetriebene Kommunikation zwischen Bluetooth-



Die Datenübertragung per Bluetooth LE ist nicht sonderlich schnell (Bild 12)

Stack und Anwenderprogramm umzustellen. Auch diese Vorgehensweise wird vom Arduino 101 unterstützt. Zur Demonstration wollen wir hier ein einfaches Programm realisieren, das die mit Pin 13 verbundene LED je nach Bedarf ein- und ausschaltet. Wie im Fall der vorhergehenden Beispiele ist auch hier die Initialisierung einer *Peripheral*-, einer *Service*- und einer *Characteristic*-Klasse erforderlich:

```
#include <CurieBLE.h>
BLEPeripheral blePeripheral;
BLEService ledService
("19B10000-E8F2-537E-4F6C-D104768A1214");
BLECharCharacteristic switchChar
("19B10001-E8F2-537E-4F6C-D104768A1214",
BLERead | BLEWrite);
```



Die Zahlensequenz sieht im Großen und Ganzen korrekt aus (Bild 11)

Setup beginnt mit der hier nicht abgedruckten Initialisierung von Ausgabeports und BLE-Peripheral. Als nächste Aufgabe folgt das Aufrufen der *setEventHandler*-Methode, die das Einschreiben von Ereignishandhabungsroutinen erlaubt:

```
void setup() {
  ...
  blePeripheral.setEventHandler
  (BLEConnected,
  blePeripheralConnectHandler);
  blePeripheral.setEventHandler
  (BLEDisconnected,
  blePeripheralDisconnectHandler);
```

Die Anmeldung von Event Handlern auf der Charakteristik-Ebene erfolgt nach demselben Schema: ▶

```
  switchChar.setEventHandler(BLEWritten,
  switchCharacteristicWritten);
  switchChar.setValue(0);
  blePeripheral.begin();
}
```

Ein wichtiger Unterschied zum vorhergehenden Beispiel ist die *loop()*-Methode, die nun wesentlich kürzer ausfällt. Die regelmäßigen Aufrufe der *poll()*-Funktion dienen dazu, die *Peripheral*-Klasse mit Rechenleistung zu versorgen:

```
void loop() {
  blePeripheral.poll();
}
```

Damit fehlt uns nur noch der Code für die Event Handler, die folgendermaßen aussehen:

```
void blePeripheralConnectHandler(BLECentral& central) {}
void blePeripheralDisconnectHandler(BLECentral& central)
{}
void switchCharacteristicWritten(BLECentral& central,
BLECharacteristic& characteristic) {
  if (switchChar.value()) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  } ...
```

Für die Ansteuerung dieser Fingerübung bietet sich das weiter oben erwähnte Control Panel aus dem Hause Nordic an. Alternativ dazu können Sie natürlich auch einen unserer nativen Clients anpassen.

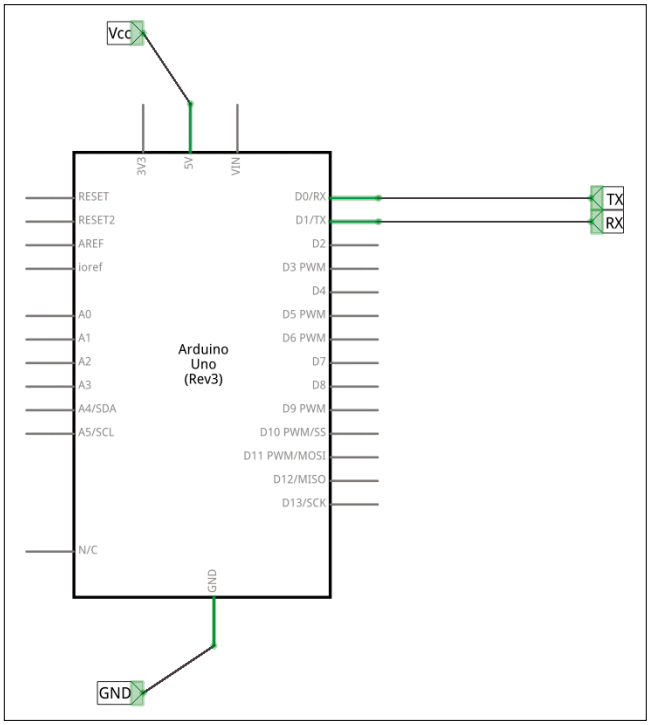
Flusskontrolle

Das Übertragen einzelner Bytes mag für Übungen interessant sein, in realen Anwendungen sendet man normalerweise größere Datenmengen. Bluetooth LE zeigt sich an dieser Stelle insofern unkooperativ, als die maximale Länge eines GATT-Pakets auf 20 Bytes beschränkt ist.

Bei der Kommunikation zwischen zwei Android-Geräten lässt sich die maximale Paketlänge durch die Methode *BluetoothGatt.requestMtu (int mtu)* auf bis zu 512 Bytes erhöhen. In der Praxis ist diese Vorgehensweise allerdings nur wenig sinnvoll, da sie von den meisten Gegenstellen nicht verstanden wird.

Als Alternative dazu bietet sich die Übertragung von Paketen an. Als Beispiel dafür wollen wir eines der weit verbreiteten 128x96-OLEDs einspannen. Die bei AliExpress zum Preis von zwischen drei und fünf US-Dollar erhältlichen Displays stellen nur wenig Ansprüche an die Host-Hardware und sollten in keinem Labor fehlen.

Die Ansteuerung erfolgt normalerweise über SPI – wir nutzen hier die Software-Implementierung. Aus schaltungstechnischer Sicht ist nicht sonderlich viel Aufwand erforderlich. Bild 13 zeigt, wie das Display mit dem Arduino 101 verbunden wird. Das Kommunikationsformat ist vergleichsweise einfach. Wir senden stets vier 20 Byte lange Pakete: Byte 0 ist die Sequenznummer des jeweiligen Pakets, während Byte 1 die Gesamtmenge (im vorliegenden Fall 3) ist. Die restlichen achtzehn Bytes sind normale Nutzdaten, die vom Arduino weiterverarbeitet werden.



Sieben Leitungen sorgen für die Verbindung zwischen Prozessrechner und Display (Bild 13)

Für die Ansteuerung des Bildschirms greifen wir auf die von Adafruit bereitgestellte Bibliothek *SSD_1306* zurück, die unter https://github.com/adafruit/Adafruit_SSD1306 zum Download bereitsteht. Fordern Sie ein ZIP-Archiv des gesamten Repositorys an, das Sie im nächsten Schritt in den Bibliotheksmanager laden. Selbiges geschieht mit der unter https://github.com/adafruit/Adafruit-GFX-Library bereitstehenden Adafruit Graphics Library, die die eigentliche Darstellungslogik enthält.

Handarbeit erforderlich

Da Adafruit die Bibliothek zum Zeitpunkt der Drucklegung noch nicht an Nicht-AVR-Arduinos angepasst hat, ist an dieser Stelle etwas Handarbeit erforderlich. Öffnen Sie die Datei *Adafruit_SSD1306.h* und kommentieren Sie die folgende Passage aus:

```
/*
#if !defined(__ARM_ARCH) && !defined(ENERGIA) &&
!defined(ESP8266)
  #include <util/delay.h>
#endif
*/
```

Zudem muss das Attribut *HAVE_PORTREG* auskommentiert werden, um die Kompilation zu ermöglichen:

```
#if defined(__SAM3X8E__)
  ...
#else
  typedef volatile uint8_t PortReg;
  typedef uint8_t PortMask;
  //#define HAVE_PORTREG
#endif
```

Wegen der Unförmigkeit der Daten nutzen wir hier keine typisierte Charakteristik. Die normale Version von BLECharacteristic nimmt einen Parameter namens *maxLen* entgegen, der die Datenmenge beschreibt:

```
BLEPeripheral blePeripheral;
BLEService dataStreamService
("19B10000-D1F2-537E-4F6C-D104768A1214");
BLECharacteristic streamCharacteristic
("19B10001-D1F2-537E-4F6C-D104768A1214",
BLERead | BLEWrite, 20);
```

Für die Kommunikation mit dem Display ist eine Instanz der *Display*-Klasse erforderlich, die folgendermaßen angelegt wird:

```
#include <gfxfont.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_MOSI   9
#define OLED_CLK   10
#define OLED_DC    11
#define OLED_CS    12
#define OLED_RESET 13
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC,
OLED_RESET, OLED_CS);
```

Für ein einfacheres Handling zeigt unser Programmbeispiel sofort nach dem Aufruf von *setup()* eine Meldung am Bildschirm an, die über die Bereitschaft zur Datenentgegennahme informiert:

```
void setup() {
  display.begin(SSD1306_SWITCHCAPVCC);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("Display bereit!");
  display.display();
```

Die Interaktion zwischen Bluetooth-Stack und Nutzerapplikation erfolgt aus Bequemlichkeitsgründen durch die im vorigen Abschnitt besprochenen Event Handler, weshalb *setup()* die Initialisierung folgendermaßen erledigt:

```
  blePeripheral.addAttribute(streamCharacteristic);
  streamCharacteristic.setEventHandler(BLEWritten,
  characteristicWritten);
  blePeripheral.begin();
}
```

Loop ist – wie immer – für das Zuweisen von Ressourcen an das BLEPeripheralDevice zuständig. Das eigentliche Entgegennehmen der Daten erfolgt in *characteristicWritten*:

```
void characteristicWritten(BLECentral& central,
BLECharacteristic& characteristic) {
  const unsigned char* myChara=characteristic.value();
  if(myChara[0]==0){
    packetID=0;
    packetIDMax=myChara[1];
    myInCopyLen=0;
  }
  else{
    packetID++;
  }
```

Die erste Aufgabe der Funktion ist das Ermitteln der Art des Pakets. Ist das Sequenzbyte *0*, so haben wir es mit dem Beginn einer neuen Datenübertragung zu tun. In diesem Fall müssen wir den Zielwert in *packetIDMax* schreiben und die Länge der eingelesenen Daten auf null setzen. Ist dies nicht der Fall, so inkrementieren wir einfach die ID des erwarteten Pakets.

Die so ermittelte ID erlaubt uns im nächsten Schritt das Prüfen der Korrektheit der Datensequenz. Wenn Sie mit Übertragungsfehlern rechnen, so können Sie an dieser Stelle im Fehlerfall eine Exception werfen:

```
  if(packetID!=myChara[1])
  {
    //Fehler
  }
```

Im nächsten Schritt kopieren wir die Informationen in das Sammelarray:

```
  for(int i=0;i<18;i++) {
    myInCopy[myInCopyLen]=myChara[i+2];
    myInCopyLen++;
  }
```

Abschließend prüfen wir, ob die Sequenz erfolgreich übertragen wurde: ▶

Wieso ist es langsam?

Ein auf Bluetooth LE spezialisiertes Beratungsunternehmen bietet unter der Adresse https://punchthrough.com/blog/posts/maximizing-ble-throughput-on-ios-and-android eine für elektronisch interessierte Personen lesenswerte Zusammenfassung der Gründe für die Langsamkeit von Bluetooth LE.

```
  if(packetID==packetIDMax){
    //Paket raus
    updateDisplayContent();
  }
}
```

Das eigentliche Ausgeben der Daten erfolgt nach dem in *setup()* vorgestellten Schema. Da die Adafruit-Bibliothek überlange Texte automatisch umbricht, brauchen wir uns an dieser Stelle keine Gedanken um die Realisierung einer Bannerfunktion machen (Listing 4).

Damit können wir uns wieder dem nativen Teil der Applikation zuwenden. Das Programm verhält sich im Großen und Ganzen analog zu den vorhergehenden Beispielen. Der einzige Unterschied findet sich im *NMGGattCallback*, der nun mit der Initialisierung eines Zählers beginnt:

Listing 4: Ausgeben der Daten

```
void updateDisplayContent()
{
  for(int i=0;i<100;i++)
  {
    myInChar[i]=myInCopy[i];
    myInLen=myInCopyLen;
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println(myInChar);
  display.display();
}
```

Listing 5: Funktion onCharacteristicWrite

```
@Override
public void onCharacteristicWrite(BluetoothGatt
gatt, BluetoothGattCharacteristic characteristic,
int status) {
  if(myCounter==4)return;
  byte[]  myByte1 = {myCounter, 3, 'A', 'B', 'C',
  'X', 'x', 'X', 'x', 'X', 'X', 'x', 'X', 'x',
  'X', 'X', 'x', 'X', 'x', 'x'};
  if(myCounter==3) {
    myByte1[19]=0;
  }
  myChara.setValue(myByte1);
  myGatt.writeCharacteristic(myChara);

  myCounter++;
  }
}
```

```
public class NMGGattCallback extends
BluetoothGattCallback {
  ...
  public byte myCounter=0;
```

Die Suche nach Geräten wird – wie gewohnt – von der Activity angestoßen. Der für uns interessante Code sitzt in *onServicesDiscovered*, wo wir im ersten Schritt ein zu übertragendes Paket anlegen:

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt, int
status) {
  ...

  byte[] myByte1={0, 3, 'A', 'B', 'C', 'X', 'x', 'X',
  'x', 'X', 'X', 'x', 'X', 'x', 'X', 'X', 'x', 'X', 'x',
  'X'};
```

Die hier genutzte Vorgehensweise ist insbesondere beim Testen von Programmen hilfreich: Ein von Hand angelegtes Array eliminiert Bit- und Encodierungsfehler. Android stellt Entwicklern an dieser Stelle eine kleine, aber bösartige Falle: Ist das Array zu lang, so wird es kommentarlos verworfen.

Anschließend senden wir die Informationen – wie gewohnt – in Richtung des Prozessrechners:

```
  myCounter++;
  myChara.setValue(myByte1);
  myGatt.writeCharacteristic(myChara);
}
```

Android ruft nach jeder erfolgreichen Übertragung von Charakteristikdaten die Funktion *onCharacteristicWrite* auf. Sie sendet bei jedem Durchlauf ein neues Paket, bis der Ziel-Zählerstand erreicht ist (Listing 5).

Fazit

Auch wenn andere Arduinos und Selbstlaborate mit Bluetooth-Modulen etwas billiger sind: Der Arduino 101 spart insbesondere bei Kleinserien wertvolle Zeit. Wenn die von Ihnen zu lösende Aufgabe in das von Bluetooth LE vorgegebene Kommunikationsschema passt, sollten Sie dem 101 eine Chance geben. Es erweist sich in vielen Fällen als unbezahlbar, wenn man sich um die Funkhardware keine Gedanken machen muss. ■

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Er lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter:
www.tamoggemon.com

.NET Developer Conference 2016

Trends, Lösungen und Know-how für Profi-Entwickler

05. – 07.12.2016, Köln

web & mobile DEVELOPER Leser erhalten 15 % Rabatt mit Code DDC16wmd

DevSessions

05.12.2016

8 halbtägige Workshops

Konferenz

06.12.2016

1 Keynote, 12 Vorträge

Workshops

07.12.2016

Ganztägiges Praxistraining

Ihre Experten (u.a.):

Bernd Marquardt
Consultant und Autor
IT-Visions.de

Dr. Holger Schwichtenberg
Fachlicher Leiter
IT-Visions.de

Gernot Starke
Gründungsmitglied des iSAQB e.V.

David Tielke
Trainer & Berater
david-tielke.de

Ralf Westphal
Mitgründer der Clean Code Developer Initiative

dotnet-developer-conference.de **#netdc16** **DDConference**

Veranstalter: developer media — Neue Mediengesellschaft Ulm mbH

Präsentiert von: dnp dotnetpro

CSS3–LAYOUTMODUL FLEXBOX

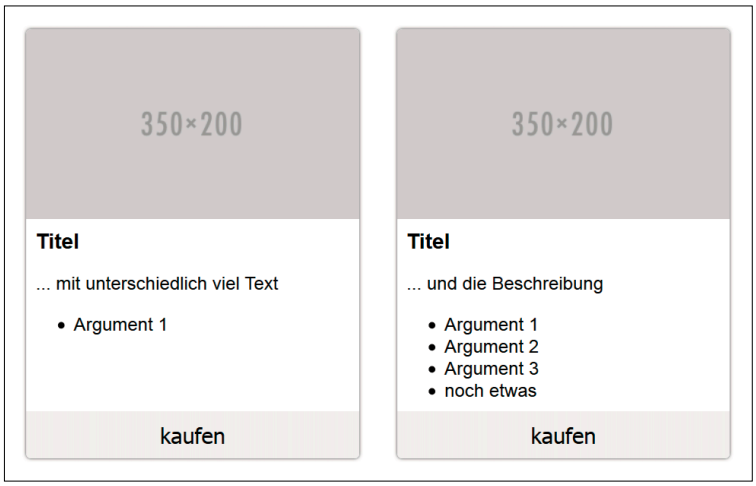
Reif für einen Wechsel

Jetzt ist der richtige Moment, für Layouts auf Flexbox umzusatteln.

Obwohl die Zeit eindeutig reif ist für den Einsatz des Flexbox-Layoutmoduls, setzen viele Frontend-Entwickler weiterhin auf die klassischen Layouttechniken wie *float, display: inline-block* oder *display: table*. Dabei gibt es gute Gründe, jetzt umzusteigen – und alles, was Sie dafür brauchen, erfahren Sie hier.

Was sind die entscheidenden Vorteile von Flexbox? Zuerst einmal löst Flexbox viele klassische Layoutprobleme, die man mit Layouteigenschaften aus CSS 2.1 nur durch Tricksereien hinbekommt, beispielsweise gleich lange Spalten oder das vertikale Zentrieren bei unbekannter Höhe.

Wenn man sich aber auf die Möglichkeiten beschränkt, wie sich klassische Aufteilungen mit Flexbox eleganter und einfacher lösen lassen, verpasst man das Neue an Flexbox.



Solche vertikalen Anordnungen ohne Definition einer Mindesthöhe oder Ähnliches lassen sich mit Flexbox einfach umsetzen (Bild 2)

Das Wesentliche ist: Beim Einsatz von Flexbox geben wir dem Browser nur Richtlinien vor und überlassen es ihm, das Richtige daraus zu machen. Statt etwa die Breiten für Bereiche exakt zu definieren, bestimmen wir beim Einsatz von Flexbox nur, dass ein bestimmte Bereich mindestens x Platz benötigt und wie überschüssiger Platz – sofern vorhanden – aufgeteilt werden soll.

Die Hauptarbeit erledigt der Browser

Das Entscheidende ist, dass man dem Browser die Hauptarbeit überlässt. Das ist auch die beste Antwort auf die heutigen Ungewissheiten beim Layouten: Wir wissen nicht, auf welchen Geräten eine Webseite betrachtet wird und wie viel Platz dort zur Verfügung steht.

Oft haben wir es auch mit unterschiedlichen Inhaltsmengen zu tun. Das ist beispielsweise dann der Fall, wenn es sich um mehrsprachige Webseiten handelt oder auch, wenn die Benutzer die Inhalte beisteuern. Um damit zurechtzukommen, dürfen die Vorgaben nicht zu starr sein – und genau das ist die Kernkompetenz von Flexbox.

Kommen wir nun aber konkreter zu den Vorteilen von Flexbox. Prinzipiell tut man sich in CSS 2.1 immer dann schwer, wenn man mehrere Elemente anordnen, dabei den Leerraum geschickt verteilen und auch noch die Anzeigereihenfolge genau bestimmen möchte.

Flexbox hat in dieser Hinsicht einige Vorteile zu bieten: Nebeneinander angeordnete Elemente sind automatisch gleich hoch. Bei Bedarf können Sie dieses Verhalten aber auch durch *align-items* abschalten. Wenn Sie etwa *align-items: center* auswählen, behalten die Elemente ihre verschiedenen Höhen und sind zentriert.

Sie können den Leerraum geschickt verteilen: Soll er gleichmäßig zwischen den Elementen aufgeteilt werden? Und auch am Anfang und Ende? Oder soll da kein Leerraum sein? Für all das ist die Eigenschaft *justify-content* zuständig. Mit *justify-content: space-around* befindet sich der Leerraum überall, mit *justify-content: space-between* nicht am Anfang und Ende, sondern nur zwischen den Elementen.

Die flexible Platzaufteilung funktioniert auch dann, wenn ein Element breiter als das Suchfeld wird (Bild 1)

Die Anzeigereihenfolge lässt sich genau festlegen. Standardmäßig werden die Elemente so dargestellt, wie sie im Quellcode stehen. Durch *flex-direction: row-reverse* (Anzeige nebeneinander) oder *flex-direction: column-reverse* (Anzeige untereinander) drehen Sie die Reihenfolge um. Außerdem können Sie auch einzelne Elemente direkt platzieren – über die Ei-

genschaft *order*. Ohne weitere Angaben schiebt *order: -1* das entsprechende Element ganz an den Anfang und *order: 1* platziert es ganz ans Ende. Aber Sie können mit *order* die Position auch für jedes Element einzeln bestimmen.

Die Breiten lassen sich flexibel definieren. So können Sie eine bevorzugte Breite angeben und genau bestimmen, was passieren soll, falls mehr Platz zur Verfügung steht. Für solche Kniffe ist die Eigenschaft *flex-grow* zuständig, oder besser noch die verkürzte Schreibweise *flex*, die die restlichen Angaben intelligent setzt. Wenn bei drei Flexitems *flex: 1* angegeben ist, erhält jedes einen Teil des verfügbaren Platzes. Bei drei Elementen ein Drittel, bei vier Elementen ein Viertel. Das ist wesentlich flexibler als Prozentangaben, die nur funktionieren, solange Sie wissen, wie viele Elemente es gibt. Zudem passen sich die Breiten bei dynamischen Veränderungen an – praktisch, wenn Sie beispielsweise ein Suchfeld bei Klick vergrößern wollen (Bild 1).

Überzeugend sind die Aufteilungsmöglichkeiten von Flexbox gerade auch bei Anordnungen in der Vertikalen. Gehen wir davon aus, dass wir Boxen mit Produktbeschreibungen haben, die unterschiedlich viele Inhalte haben. Trotzdem sollen sie gleich hoch sein und der Bestellbutton immer unten angeordnet sein (Bild 2). Mit Flexbox sind solche Aufteilungen einfach und schnell umgesetzt.

Ein weiterer Vorteil von Flexbox ist die bessere Performance. Das betrifft besonders dynamische Veränderungen am Layout: Hier zeigt sich eine bessere Performance bei mit Flexbox realisierten Layouts als etwa bei den klassischen *float*-Layouts.

Wenn das alles so positiv ist – warum wird Flexbox dann nicht häufiger und umfassender eingesetzt? Dafür gibt es folgende – scheinbare – Gründe:

- mangelnde Browserunterstützung,
- verwirrende Angaben durch mehrere Versionen der Spezifikation,
- Komplexität von Flexbox,
- unbekannte Bugs.

Nehmen wir diese Argumente im Folgenden einmal genauer unter die Lupe.

Browserunterstützung

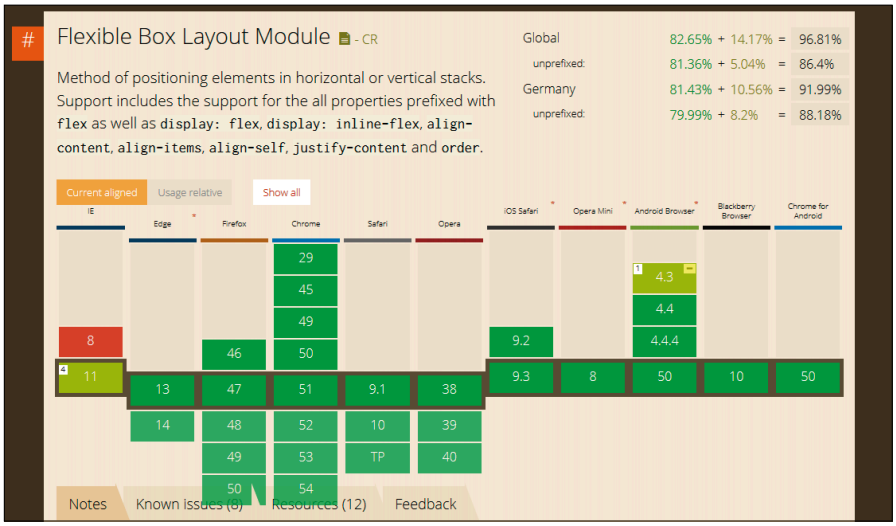
Als Argument gegen Flexbox wurde und wird oft die dürftige Browserunterstützung herangezogen. Dabei sieht es heute, im Jahr 2016, gar nicht mehr so schlecht aus. Alle aktuellen Browser unterstützen Flexbox. Der Internet Explorer ab Version 10 (Bild 3). Wie relevant sind Internet-Explorer-Versionen vor 10?

Bei Caniuse ist immer oben rechts eine kleine Statistik aufgeführt, die angibt, welcher Prozentsatz an Browsern ein Feature unterstützt. Bei Flexbox sind es global 96,81 Prozent der Browser, in Deutschland 97,36 Prozent. Das heißt aber nicht, dass die verbleibenden Prozente nichtunterstützende Browser sind, da 2,04 Prozent »untracked Browser« sind, also Browser, die sich nicht zu erkennen geben. Das heißt, dass an dieser Stelle nie 100 Prozent Unterstützung erscheinen, selbst die ehrwürdige Eigenschaft *border-radius* kommt nach denselben Statistiken auf deutlich weniger als 100 Prozent. Fazit: Die Browserunterstützung ist eindeutig gut genug.

Wichtiger als die globalen Statistiken sind aber natürlich Ihre eigenen projektspezifischen Statistiken. Und hier können Sie dann Ihren eigenen Schwellenwert bestimmen und festlegen, dass Browser mit einer Verbreitung unter x Prozent nicht mehr berücksichtigt werden.

Aber auch wenn Sie den Internet Explorer < 10 noch unterstützen müssen, ist das kein Argument gegen Flexbox – zu den Möglichkeiten kommen wir gleich noch.

Abschreckend wirkt bei einer ersten Begegnung mit Flexbox, dass es mehrere Versionen der Spezifikation gibt, die teilweise in den Browsern implementiert sind. Neben der



Browserunterstützung für Flexbox: Alles im grünen Bereich (Bild 3)

heutigen Standardversion gibt es die Vorversion von 2009 und eine Zwischenversion – auch liebevoll Tweener genannt – von 2011. Erschwerend kommt hinzu, dass in den einzelnen Spezifikationen die Eigenschaften oder Werte teilweise unterschiedlich heißen: So wird die Flexbox-Darstellung in der Standardversion mit *display: flex* aktiviert, die Zwischenversion verlangte *display: flexbox* und die ursprüngliche *display: box*.

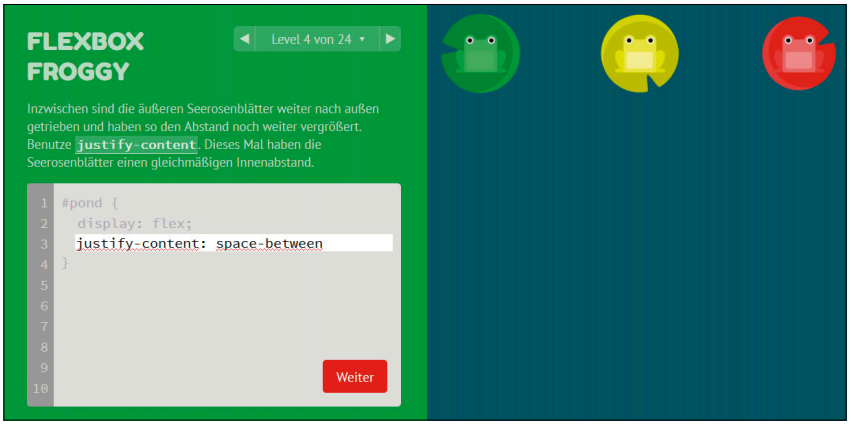
Das ist allerdings halb so schlimm, wie es auf den ersten Blick aussieht. Zuerst einmal unterstützen über 86 Prozent der Browser in Deutschland – so sagen es die Daten von Caniuse – die aktuelle Spezifikation. Nur 8,2 Prozent haben Vorversionen implementiert – und dabei verschieben sich die Werte bei fortschreitender Zeit automatisch zugunsten der Standardversion.

Trotzdem sollten Sie neben den Standardangaben auch noch die Angaben für die Vorversionen machen, oder bes- ▶

ser: machen lassen. Hier schlägt die Stunde von Autoprefixer, das Sie in Ihren Build-Prozess integrieren oder alternativ in der Online-Version nutzen können. Autoprefixer ist so intelligent, dass es nicht einfach nur Varianten mit Präfix ergänzt, sondern wirklich die benötigten Eigenschaften erzeugt.

Auf den ersten Blick kompliziert

Auf den ersten Blick wirkt Flexbox kompliziert. Es gibt zwölf neue Eigenschaften und außerdem eine Reihe von neuen Begriffen wie Flexcontainer, Flexitems, Hauptachse und Nebenachse et cetera. Die Begriffe Flexcontainer und Flexitems sind relativ selbsterklärend: Das Elternelement der anzuordnenden Elemente wird durch *display: flex* zum Flexcontainer – die Kindelemente sind die Flexitems, die angeordnet werden sollen. Ein bisschen schwieriger sind die Begriffe primäre und sekundäre Achse. Der Grund für diese Begrifflichkeiten: Mit Flexbox können Sie Elemente ja sowohl horizontal als auch vertikal anordnen. Standardmäßig werden die Kindelemente nebeneinander platziert:



Frösche auf Seerosen platzieren und dabei Flexbox lernen (Bild 4)

```
.container {
  display: flex;
}
```

Für eine Anordnung untereinander schreiben Sie:

```
.container {
  display: flex;
  flex-direction: column;
}
```

Für das Feintuning gibt es eine Reihe von Eigenschaften, die sich aber entweder auf die Vertikale oder Horizontale beziehen – in Abhängigkeit davon, welche grundlegende Darstellung gewählt ist.

Die Angabe *justify-content: flex-end* zum Beispiel bewirkt bei einer Darstellung nebeneinander, dass die Elemente ganz rechts platziert werden – bei einer Darstellung untereinander sind die Elemente hingegen am unteren Rand des umfassenden Elements angeordnet. Entscheidend ist eben immer, wie die prinzipielle Anordnung gewählt ist. Um die Eigenschaften verallgemeinern zu können, wird der Begriffe Hauptachse eingeführt, während für andere Eigenschaften die Nebenachse gilt. Erst einmal ist das natürlich nur graue Theorie und man benötigt Praxis, um sich mit den Begriffen und Eigenschaften vertraut zu machen. Dabei helfen eine Reihe von Tools. Zwei sollen hier erwähnt werden.

Sehr nett gemacht und zudem äußerst hilfreich ist Flexbox Froggy. Bei diesem Spiel geht es darum, bunte Frösche auf den passenden Seerosenblättern zu platzieren – unter Einsatz von Flexbox, versteht sich (Bild 4).

Fast schon ein Klassiker ist FlexyBoxes (Bild 5): Hier können Sie alle Flexbox-Eigenschaften austesten und sehen direkt ihre Auswirkungen.

Mit diesen beiden Tools sollte Ihnen der Einstieg in Flexbox leicht gelingen und zudem vielleicht sogar ein bisschen Spaß machen.

Ein weiteres Argument gegen Flexbox ist, dass es hier Browserbugs gibt. Das ist an sich nichts Ungewöhnliches: Wer mit CSS arbeitet, kennt Browserbugs oder auch CSS-Eigentümlichkeiten zur Genüge. Warum also ist das ein Argument gegen Flexbox?

Weil man die anderen Bugs und Ungereimtheiten seit Jahren kennt und die von Flexbox nicht. Getreu der Devise, dass das bekannte Unglück dem unbekannten vorzuziehen ist, wirkt es abschreckend, wenn es bei dem schönen Neuen eben auch Bugs gibt …

Überschaubare Anzahl von Bugs

Die gute Nachricht ist zum einen, dass die Anzahl der Bugs überschaubar ist, und zum anderen, dass sie gut dokumentiert sind. Die richtige Anlaufstelle dafür ist Flexbugs. Hier finden Sie die Auflistung der Bugs samt der Angabe, welche Browser betroffen sind, und welchen Workaround Sie wählen können.

Prinzipiell gilt: Von den 14 aufgeführten Bugs betrifft die überwältigende Mehrheit den IE 10/11 – nämlich sieben. Zwei gelten nur für den Internet Explorer 10, zwei betreffen den Safari, einer den Firefox, einer ältere Chrome/Opera-Versionen und zwei gelten auch in aktuellen Browsern. Das heißt, dass die größten Probleme im IE 10 und IE 11 auftauchen, ansonsten sind es eher Kleinigkeiten, und je aktueller der Browser, desto weniger Bugs.

Um dem Internet-Explorer-Bashing zuvorzukommen: Dass im Internet Explorer so viele Fehler auftauchen, liegt nicht daran, dass Microsoft Spezifikationen ignoriert, sondern dass der Internet Explorer 10 der älteste der Flexbox unterstützenden Browser ist. Und die Spezifikation, auf die er sich bezieht, ist eben nicht mehr die aktuelle.

So hieß es in einer älteren Version der Spezifikation, dass bei der bevorzugten Größe bei der Kurzschreibweise *flex* eine Einheit erforderlich ist. Das gilt heute nicht mehr, aber im

Code-Generator und Experimentierkasten für Flexbox (Bild 5)

Internet Explorer 10 wird deswegen die folgende Angabe als ungültig erachtet:

```
flex: 1 0 0;
```

Abhilfe schaffen Sie, indem Sie eine Einheit bei der letzten Angabe schreiben.

```
flex: 1 0 0%;
```

Anstelle von Prozent wäre eigentlich auch jede andere Einheit möglich, aber das kann Probleme beim Minimieren der CSS-Datei bereiten – gängige Tools entfernen oft die Einheit bei *0*, wenn es sich um Pixel handelt, lassen Prozentangaben aber unberührt.

Aber es wird noch besser: Da die Bugs bekannt sind, wäre es doch schön, wenn es ein Tool gäbe, das einen davor warnt und sogar, wo möglich, automatisch für Abhilfe sorgt. Das gibt es: in Form eines Post-CSS-Plug-ins mit Namen postcss-flexbugs-fixes. Wenn Sie mit PostCSS arbeiten (zum Beispiel, um Autoprefixer einzusetzen), lässt es sich gut in den Build-Prozess integrieren und es behebt dann mögliche Probleme ganz von selbst.

Flexbox ohne Flexbox

Doch was tun, wenn man noch den Internet Explorer 9 unterstützen möchte? Auch hier gibt es mehrere Alternativen. Die beste Möglichkeit besteht darin, Flexbox für kleine Layoutkomponenten einsetzen, bei denen es akzeptabel ist, wenn die Darstellung im IE 9 weniger optimiert ist, das heißt, ganz im klassischen Sinne des Progressive Enhancements vorzugehen.

Hilfreich bei der Suche nach der richtigen Fallbacklösung ist die Webseite »Almost complete guide to flexbox (without flexbox)« von Kenan Yusuf (Bild 6). Die Beispielcodes sind nach Flexbox-Eigenschaften angeordnet. Bei jedem Flexbox-Feature wird gezeigt, wie man dasselbe auch mit klassischem oder auch modernerem – zumindest aber IE 9-kompatiblem – Code erreichen kann. Praktischerweise lernt man dabei auch einige Tricks klassischer CSS-Methoden kennen.

Abwärtskompatibilität in der Praxis

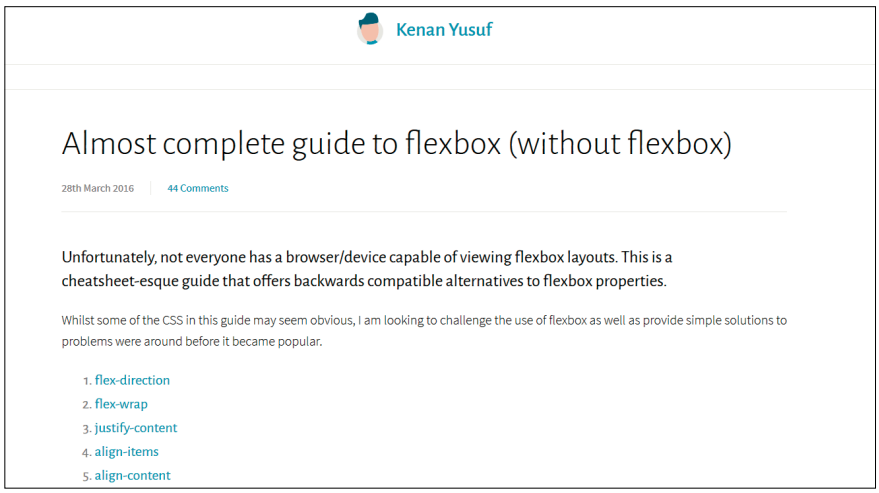
Sehen wir uns ein Beispiel für die Implementierung einer Fallbacklösung an. In vielen Fällen können Sie die Flex-Angaben zusätzlich zum normalen Code machen, da bestimmte klassische CSS-Eigenschaften beim Einsatz von Flexbox keine Auswirkung haben. Ignoriert werden etwa Breitenangaben, sofern Sie *flex* einsetzen, und *float* hat bei Flexitems ohnehin keine Wirkung.

Nehmen wir an, Sie möchten mit Flexbox zwei Elemente nebeneinander platzieren:

```
<section class="container">
  <div class="eins">
    Text 1
  </div>

  <div class="zwei">
    Text 2
  </div>
</section>
```

Dafür machen Sie das umfassende Element zum Flexcontainer und bestimmen die Ausrichtung:

```
.container {
  display: flex;
```



Fallbacklösungen für Flexbox mit klassischem CSS (Bild 6)

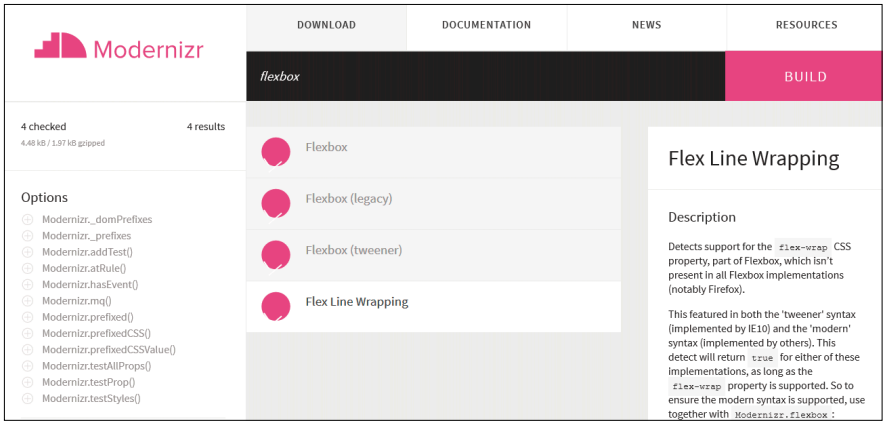
```
  flex-flow: row wrap;
  justify-content: space-between;
  align-items: baseline;
}
```

Falls nicht genügend Platz vorhanden ist, sorgt die Angabe *flex-flow: row wrap* dafür, dass die Elemente automatisch untereinander angeordnet werden. Dies ist eine sehr praktische Angabe, die Sie für Layouts nutzen können, die sich von selbst an unterschiedliche Gegebenheiten anpassen – ganz ohne Media Queries.

Außerdem erhalten *.eins* und *.zwei* die benötigte Breite:

```
.eins, .zwei {
  width: 40%;
}
```

Nun fehlt lediglich noch der Fallback für die Nicht-Flexbox-Browser. In diesen können wir die Elemente links und rechts floaten:



Bei Modernizr kann man bei Flexbox vier verschiedene Features testen (Bild 7)

```
.eins {
  float: left;
}
.zwei {
  float: right;
}
```

Praktischerweise hat *float* bei den Flexitems keinerlei Auswirkung, sodass wir nicht tricksen müssen, um unterschiedliche Angaben an verschiedene Browser auszuliefern.

Wo es geht, ist das die beste Lösung: Damit haben wir eine verbesserte oder sogar optimale Darstellung in Flexbox unterstützenden Browsern und eine akzeptable Darstellung mit kleinen Abweichungen im Internet Explorer 9 – und das ohne irgendwelche Hacks.

Mit Modernizr testen

In manchen Fällen kann es allerdings notwendig sein, unterschiedliche Angaben für Flexbox- und Nicht-Flexbox-Browser zu machen. In diesen Fällen können Sie auf Modernizr zurückgreifen.

Modernizr ist bei Flexbox großzügig und bietet gleich vier verschiedene Testmöglichkeiten an (Bild 7). Es können die folgenden Klassen vergeben werden:

- **flexbox (no-flexbox)** für die Unterstützung der Standardeigenschaften des Flexbox-Layoutmoduls, wie sie etwa im aktuellen Firefox, Chrome und Edge implementiert sind.
- **flexboxlegacy (no-flexboxlegacy)** dient der Überprüfung auf die Unterstützung der 2009er Version von Flexbox, wie sie etwa im Safari 6 oder Android 4.3 implementiert ist.
- **flewboxtweener (no-flexboxtweener):** Damit können Sie Browser auswählen, die die Zwischenversion von 2011 unterstützen. Konkret ist das nur ein Browser, nämlich der Internet Explorer 10.
- **flexwrap (no-flexwrap):** Damit kontrollieren Sie, ob die Eigenschaft *flex-wrap* implementiert ist.

Drei der Tests beziehen sich auf bestimmte Versionen von Flexbox – *flexbox, flexboxlegacy* und *flexboxtweener*. Der vierte Test – *flexwrap* – hingegen zielt nur auf eine bestimmte Eigenschaft ab, nämlich auf *flex-wrap*.

Durch den folgenden Code bewirken Sie, dass die Kindelemente des *.container*-Elements zu Flexitems werden und bei Bedarf auch mehrzeilig angeordnet werden:

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

In der Dokumentation von Modernizr finden Sie den Hinweis, dass Firefox *flex-wrap* nicht unterstützt. Das gilt allerdings nur für den Firefox bis zur Version 27. Ab 28 wird die Eigenschaft korrekt interpretiert. Da wir uns beim Firefox mit raschen Schritten der 50er-Version nähern, scheinen diese Uraltversionen nicht unbedingt berücksichtigenswert. Übrigens: Auch der Internet Explorer 10 unterstützt *flex-wrap*.

Es gibt allerdings einen Fall, in dem es sinnvoll sein könnte, auf die Unterstützung für *flex-wrap* zu prüfen. Angenommen, Sie haben Elemente, die Sie über Flexbox mehrzeilig anordnen wollen. Wenn ein Browser jetzt *flex-wrap* nicht interpretiert, aber die Flexbox-Eigenschaften an sich, so würden die Elemente gnadenlos nebeneinander angeordnet, ohne in die neue Zeile zu wandern – und das könnte dazu führen, dass sie herausragen oder abgeschnitten werden. Durch den folgenden Code – sofern Modernizr eingebunden ist und auf die verschiedenen Eigenschaften testet – sorgen Sie dafür, dass das nicht passiert:

```
.flexbox.flexwrap .container {
  display: flex;
```

```
  flex-wrap: wrap;
}
```

Hilfreich, wenn Sie Modernizr zur Feature-Detection einsetzen, ist ein Bookmarklet von Chris Wright, das die Klasse beim HTML-Start-Tag automatisch umschaltet. Aber Sie benötigen, wie gesagt, nicht immer Modernizr und können in vielen Fällen den Code für Nicht-Flexbox-Browser mit dem Code für Flexbox-Browser mischen.

JavaScript zur Hilfe

Wenn Sie nicht nur eine ähnliche Darstellung wünschen, sondern die Flexbox-Funktionalität in Nicht-Flexbox-Browsern nachbilden wollen, brauchen Sie Javascript – genau genommen das Skript Flexibility. Sie müssen es zuerst einbinden:

```
<script src="flexibility.js"></script>
```

Dort, wo Sie ein Nachbessern für den IE 9 und früher wünschen, ergänzen Sie die Flexbox-Angabe mit dem *-js*-Präfix:

```
.container {
  -js-display: flex;
  display: flex;
}
```

Links zum Thema

- Praktische Beispiele für Flexbox
  http://callmenick.com/post/flexbox-examples
- Flexbox-Browser-Bugs
  https://github.com/philipwalton/flexbugs
- Flexbox-Browser-Bugs mit PostCSS-Plug-in beheben
  https://github.com/luisrudge/postcss-flexbugs-fixes
- Flexbox lernen
  http://the-echoplex.net/flexyboxes
  http://flexboxfroggy.com
- Mit Modernizr auf Flexbox testen
  http://zomigi.com/blog/using-modernizr-with-flexbox
- Bookmarklet zum Umschalten der Flexbox-Klasse
  http://chriswrightdesign.github.io/tooflexy
- Cross-Browser-Code erzeugen mit Autoprefixer
  https://autoprefixer.github.io
  https://github.com/autoprefixer/autoprefixer.github.io
- Flexbox-Features mit klassischem CSS erreichen
  https://kyusuf.com/post/almost-complete-guide-to-flexbox-without-flexbox
- JavaScript-Fallback
  https://github.com/jonathantneal/flexibility
- Flexbox am besten für kleine Komponenten
  https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout

Übrigens gibt es ein PostCSS-Plug-in, das dies für Sie automatisch erledigt.

Zum Schluss aktivieren Sie das Polyfill für den äußersten Flexcontainer:

```
flexibility(document.documentElement);
```

Dass die Zeit reif ist für den Einsatz von Flexbox, heißt aber natürlich nicht, dass Flexbox in allen Fällen die richtige Wahl ist. Jake Archibald erklärt in einem Artikel, man solle Flexbox nicht für vollständige Layouts nutzen. Er begründet das damit, dass es beim progressiven Laden beim Einsatz von Flexbox zu unschönen Neuanordnungen kommen kann, weil die Elemente sich nach und nach aneinander orientieren, was er in einem Film veranschaulicht. Zum Vergleich zeigt er dasselbe Layout beim progressiven Laden beim Einsatz des Gridlayout-Moduls.

Und tatsächlich gibt es eine ganz klare Aufgabenteilung zwischen Gridlayout und Flexbox-Layout: Gridlayout ist für komplexe Layouts geeignet, eben für vollständige, komplexe Rasterrealisierungen. Flexbox hingegen ist für die Anordnung von Boxen vorgesehen, und damit im Wesentlichen für eindimensionale Layouts. Allerdings können dank *flex-wrap* Elemente auch auf mehrere Zeilen beziehungsweise Spalten verteilt werden, womit man den rein eindimensionalen Bereich verlässt.

Das heißt, eine klassische Aufteilung würde so aussehen, dass Gridlayout für die große Einteilung benutzt wird, während mit Flexbox die kleineren Layoutkomponenten (Navigationspunkte, Anordnung innerhalb von Komponenten) platziert werden.

Das ist richtig so, allerdings ist derzeit Gridlayout noch nicht zum produktiven Einsatz geeignet, weil es im Chrome und Firefox nur hinter einem Flag funktioniert. Bis sich das ändert, gibt es also nur zwei Möglichkeiten: Entweder erstellen Sie trotzdem das umfassende Layout mit Flexbox und nehmen die Neuanordnungen beim Laden in Kauf, oder Sie setzen für das große Layout auf *float* oder Ähnliches und realisieren die Anordnung innerhalb der Bereiche – Navigationsleisten, Produktpräsentationen, Anordnungen von Social-Media-Icons et cetera – mit Flexbox.

So oder so sollte Flexbox in Ihrem CSS-Werkzeugkoffer nicht fehlen. Und dass der richtige Moment zum Einsatz gekommen ist, zeigt sich auch an den überzeugenden Tools, die es inzwischen gibt: von den hilfreichen Lernspielen über Bugsammlungen bis hin zu passenden PostCSS-Plug-ins. ■

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

WEBSOCKETS IN HTML5 UND JAVASCRIPT

Bidirektional

Das WebSockets-Protokoll ermöglicht bidirektionale Verbindungen zwischen einem Server und einer Webanwendung.

Die HTML5-WebSocket-Spezifikation ermöglicht den Aufbau einer persistenten bidirektionalen Verbindung zwischen Client und Server über das WebSocket-API, was den Anforderungen heutiger Webseiten entspricht. Das WebSocket-Protokoll ist Teil des HTML5-Standards und umgeht das HTTP-Protokoll, über das der Client den Server regelmäßig anfragen muss, um Daten zu erhalten.

Das WebSocket-Protokoll, dessen Ansteuerung mit JavaScript über das WebSocket-API erfolgt, ermöglicht das schnelle Senden und das sofortige Empfangen von Texten, binären Arrays oder Blobs.

Vom Standard zur Notlösung zum Standard

Während der Besucher einer Website früher noch bereit war, für das Bereitstellen von Informationen längere Wartezeiten in Kauf zu nehmen, sollen diese heute sofort verfügbar und zugänglich sein und visualisiert werden. Ein Zitat der deutschen Publizistin Anita Berres bringt es auf den Punkt: »Lesen Sie schnell, denn nichts ist beständiger als der Wandel im Internet!« (Quelle: »Marketing und Vertrieb mit dem Internet«, ISBN 978-3-642-60577-2)

Denn genau hier liegt die Krux der Geschichte: Der ständige Wandel des Internets und die damit verbundenen zunehmenden Anforderungen an Webapplikationen führen dazu, dass die Techniken, die in der Anfangszeit des Internets von Entwicklern genutzt wurden und durchaus ihre Daseinsberechtigung hatten, in der Gegenwart überholt sind.

Das Internet basiert im Grunde genommen auf dem HTTP-Protokoll, also einem Frage- und Antwortmuster. Das heißt, ein Client schickt eine Anfrage an den Server und wartet auf eine Antwort (Bild 1).

Um den heutigen Anforderungen zu entsprechen, nutzen Software-Entwickler gegenwärtig oft das sogenannte Long-Polling. Hierbei hält der Server eine gesendete Anfrage so lange geöffnet, bis neue Daten verfügbar sind und er diese dann als Antwort an den Client zurücksenden kann. Der Client stellt nach Erhalt der Antwort wiederum direkt eine Anfrage an den Server (Bild 2).

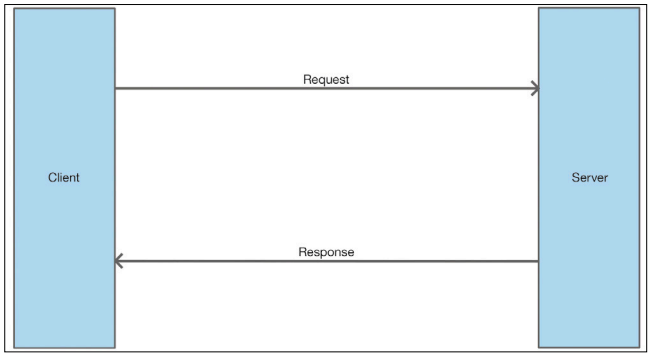
Long Polling ist jedoch im Grunde nur eine Notlösung zum Vermeiden kontinuierlich neuer Anfragen an den Server – zum Beispiel beim Aufruf von Ajax – und zum Beschränken auf eine Anfrage. Mit dem Erscheinen des WebSocket-Protokolls 2009 und dessen Standardisierung 2011 durch die IETF ist dieses Vorgehen hinfällig.

Über HTML5, JavaScript und das WebSocket-API verfügen Software-Entwickler nun über die Möglichkeit, eine bidirektionale Verbindung zwischen Client und Server aufzubauen. Diese Verbindung bleibt so lange geöffnet, bis eine der beiden Seiten diese schließt. Im Gegensatz zu Long Polling muss der Client die Verbindung zum Server nicht erneut öffnen. Ebenfalls unnötig ist, dass der Server periodisch Daten abfragt. Dies dient dazu, unnötige Last zu vermeiden (Bild 3).

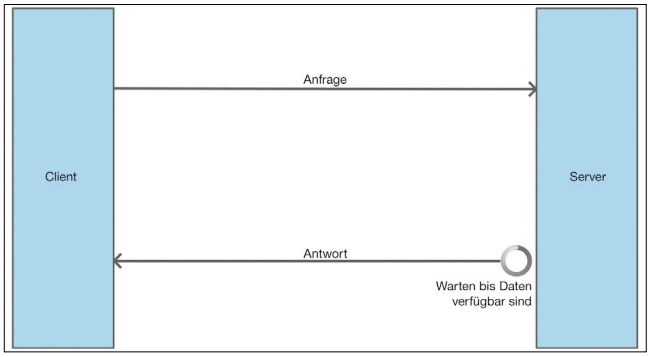
Handshake

Beim Verbindungsaufbau führen der Server und der Client einen sogenannten Handshake durch. Das bedeutet, der Client sendet initial eine HTTP-Anfrage an den Server und teilt ihm mit, dass die zukünftige Kommunikation auf dem WebSocket-Protokoll basieren soll:

```
GET http://localhost:1337/ HTTP/1.1
Host: localhost:1337
```



HTTP-Protokoll: Client schickt eine Anfrage an den Server und wartet auf eine Antwort (Bild 1)



Long-Polling: Verzögertes Antworten auf eine vom Client initiierte Anfrage (Bild 2)

WebSocket-API: Bidirektionale Verbindung zwischen Client und Server (Bild 3)

```
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: TAiaNr3IZeJ42dfYra2MwA==
Sec-WebSocket-Extensions: permessage-deflate;
client_max_window_bits
```

Optional können im Header noch spezifische Protokolle zur Kommunikation übergeben werden. Der Server antwortet bei einem erfolgreichen Verbindungsaufbau mit dem Statuscode *101* und teilt somit dem Client mit, dass das Upgrade auf das WebSocket-Protokoll erfolgreich war:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: VLwvKbvCM28kuvho5Re3ceQ0g8k=
Sec-WebSocket-Extensions: permessage-deflate
```

Tabelle 1 zeigt, welche Angaben der Header enthalten kann. Zur Verwendung des WebSockets-Protokolls in einer Webanwendung muss der Browser auf der Seite des Clients WebSockets unterstützen.

Browser überprüfen

Webentwickler können die Unterstützung des Client-Browsers einfach über die Code-Implementation im folgenden Listing überprüfen:

```
if (window.WebSocket){
  console.log("Browser unterstützt WebSockets!");
} else {
  console.log("Browser unterstützt keine WebSockets!");
}
```

In Hinblick auf ihre Webapplikation sollten Anwendungsentwickler im Fall von nicht unterstützten Browsern Ersatzstrategien einsetzen und den Einsatz von entsprechenden Techniken, wie zum Beispiel der Long-Polling-Technik, erwägen.

Der Server, mit dem eine bidirektionale Verbindung aufgebaut werden soll, muss das Protokoll unterstützen. In den folgenden Beispielen wird nativ mit Node.js ein Server implementiert, der das WebSocket-Protokoll unterstützt. In nahezu jeder Programmiersprache gibt es die Möglichkeit, WebSockets zu verwenden. Mittlerweile setzen die meisten Entwickler Frameworks wie zum Beispiel Socket.io für Node.js ein, um mit dem Server zu kommunizieren.

Die Beschreibung dieser Frameworks und ihrer Feinheiten würde jedoch den Rahmen des Artikels sprengen. Trotzdem sollten sich Entwickler mit diesen Frameworks beschäftigen, da die Behandlung dieser Kinderkrankheiten und die Behebung der Fehler bei der nativen Implementation Alternativen schaffen und die Anwendung erleichtern. Eine Erleichterung in diversen Frameworks ist zum Beispiel die Implementierung von Ersatzstrategien für nicht unterstützte Browser.

Um den Server per Node.js zu installieren, ist die Erstellung eines leeren Dateiordners durch den installierenden Nutzer zu empfehlen, in dem die Daten des zu installierenden Servers abgelegt werden. Der Benutzer führt per Konsolenbefehl innerhalb des Ordners den Befehl *npm init* aus und gibt die Angaben ähnlich wie im Listing 1 ein.

WebSocket-Funktionalität hinzufügen

In dem Verzeichnis, in dem von dem installierenden Benutzer der Konsolenbefehl *npm init* ausgeführt wurde, befindet sich die Datei *package.json*. Der Benutzer muss nun noch den Befehl *npm install -- save ws* durchführen, um dem Server die WebSocket-Funktionalität durch den Download des *ws*-Packets hinzuzufügen.

Anschließend ist im Serverordner die Erstellung der Datei *server.js* oder der vom Nutzer unter dem Punkt *entry point* selbst benannten Datei erforderlich. Der Programmiercode aus Listing 2 wird dieser Datei hinzugefügt. Als letzter Schritt wird der Server mit dem Konsolenbefehl *node server.js* gestartet. ▶

Tabelle 1: Header-Beschreibung

Sec-WebSocket-Key	Wird einmalig in der HTTP-Anfrage beim Handshake vom Client zum Server mitgesendet.
Sec-WebSocket-Accept	Wird einmalig vom Server in der HTTP-Antwort zurückgeliefert, um dem Client mitzuteilen, dass der Server die Anfrage akzeptiert und das WebSocket-Protokoll unterstützt.
Sec-WebSocket-Extensions	Kann mehrfach in der HTTP-Anfrage, aber nur einmal in der HTTP-Antwort des Clients enthalten sein. Das WebSocket-Protokoll unterstützt seine Erweiterung. Diese Angabe dient dazu, diese Erweiterungen zwischen Client und Server zu kommunizieren.
Sec-WebSocket-Protokoll	Wird im initialen Handshake mitgeliefert, um Unterprotokolle zwischen Client und Server auszuhandeln. Der Server liefert im gleichen Wert das bevorzugte Protokoll zurück.
Sec-WebSocket-Version	Gibt die Protokollversion an und wird im initialen Handshake an den Server geschickt.

Um eine Verbindung mit dem Server einzurichten, instanzieren Webentwickler das WebSocket-Objekt:

```
var webSocketConnection = new WebSocket
('ws: //localhost:1337');
```

Der Konstruktor nimmt die Parameter *url* und *protocols* entgegen. Der erste Parameter öffnet eine Verbindung zum Server. Durch den zweiten Parameter können Entwickler Unterprotokolle definieren, die vom Server bevorzugt akzeptiert werden, zum Beispiel XMPP (Extensible Messaging Presence Protocol), SOAP (Simple Object Access Protocol) oder ein benutzerdefiniertes Protokoll. Der Server akzeptiert beim Verbindungsaufbau eines oder keines der Protokolle und akzeptiert den Handshake des Clients beziehungsweise schließt die Verbindung wieder.

Es können somit zwei Ereignisse auftreten: Einerseits kann die Verbindung erfolgreich sein und der Client kann wie im folgenden Listing mit der Rückruffunktion *onopen* auf den erfolgreichen Verbindungsaufbau reagieren. Andererseits kann das *Error*-Ereignis eintreten, auf das später eingegangen wird:

```
webSocketConnection.onopen = function () {
  console.log("Verbindung zum Server hergestellt!");
};
```

Wenn das Ereignis *Open* eingetreten ist, war der Handshake erfolgreich und die Verbindung wurde auf das WebSocket-Protokoll hochgestuft. Sollte die Verbindung fehlschlagen oder ein Fehler in der Kommunikation entstehen, erhält der Client das *Error*-Ereignis. Hierfür existiert die Rückruffunktion *on error*, die ein Entwickler in der Clientprogrammierung verwenden kann:

```
webSocketConnection.onerror = function (error) {
  console.log(error.message);
};
```

Das *Error*-Ereignis tritt nicht nur beim Öffnen der Verbindung auf, sondern bei allen Fehlern in der Kommunikation mit dem Server. Anwendungsentwickler können jedoch davon ausgehen, dass die Verbindung geschlossen wird und dass das *Close*-Ereignis als nächstes Ereignis folgt, da das *Error*-Ereignis die Verbindung schließt.

Listing 1: Installation des Servers

```
npm init
name: websocket-server
version: 1.0.0
description: A simple node chat server
entry point: server.js
test command: [Enter]
git repository: [Enter]
keywords: [Enter]
author: Philippe Bénard
License: [Enter]
```

Listing 2: server.js–Ergänzung

```
var WebSocketServer = require('ws').Server;
var wss = new WebSocketServer({port: 1337});

var messages = [];
wss.on('connection', function (ws) {
  messages.forEach(function(message){
    ws.send(message);
  });
  ws.on('message', function (message) {
    messages.push(message);
    console.log('Message Received: %s', message);
    wss.clients.forEach(function (conn) {
      conn.send(message);
    });
  });
});
```

Senden einer Nachricht

Wenn die Verbindung erfolgreich ist, kann der Client die Kommunikation mit dem Server beginnen. Zum Senden einer Nachricht verwendet der Client die *send()*-Funktion des WebSockets-Objekts. Die Funktion nimmt die Typen String, Blob, ArrayBuffer und ArrayViews entgegen.

Um eine simple Textnachricht zu schicken, übergibt der Entwickler einen String als Parameter an die *send()*-Methode. Erhält die *send()*-Methode einen String, so wird dieser UTF-8-codiert an den Server übertragen:

```
function SendText(text) {
  webSocketConnection.send(text);
}
SendText("Hallo Server!");
```

Das folgende Listing beschreibt, wie man einen JavaScript-Wert über *JSON.stringify()* in einen JSON-String konvertiert, um diesen vom Client zum Server zu übertragen:

```
function SendJson(obj) {
  webSocketConnection.send(JSON.stringify(obj));
}
SendJson({ headline: "Json mit WebSockets",
description: "Json" });
```

Wie bereits beschrieben, kann der Client mit WebSockets ebenfalls binäre Daten verschicken. Binäre Daten können per ArrayBuffer, Blob oder als ArrayView über WebSockets ausgetauscht werden. Wenn die Anwendung kleine Datensätze zum Verschicken von Daten vorgibt, empfiehlt es sich, ArrayBuffer zu verwenden.

Listing 3: Status der Verbindung

```
function SendToServer(sendAs) {
  if (webSocketConnection.readyState ===
  WebSocket.OPEN) {
    switch (sendAs) {
    case "text":
      SendText("Hallo Welt!");
      break;
    case "json":
      SendJson({ headline: "Json mit
      WebSockets", description: "Json" });
      break;
    case "buffer":
      SendArrayBuffer();
      break;
    case "file":
      SendBlob();
      break;
    default:
      console.log("Unbekannte Versandart");
    }
  } else {
    console.log("Verbindung zum Server ist nicht
    möglich");
  }
}
```

Tabelle 2: Status der Verbindung

readyState	Beschreibung
OPEN	Die Verbindung ist geöffnet.
CONNECTING	Die Verbindung befindet sich gerade im Aufbau (Standard).
CLOSING	Die Verbindung ist dabei, sich zu schließen.
CLOSED	Die Verbindung zum Server ist zurzeit beendet.

Listing 4: Empfang von Nachrichten

```
// Maximaler Puffer von 10k
var treshold = 10240;

// Auf das Open-Ereignis warten
webSocketConnection.onopen = function () {
setInterval(function() {

// Überprüfen der bislang noch nicht versendeten und
// zwischengespeicherten Daten
if (webSocketConnection.bufferedAmount < treshold)
  // Update der Daten, wenn der Puffer nicht voll ist
  webSocketConnection.send(updateData());
}, 1000);
```

Wenn es sich um größere Datensätze wie zum Beispiel Bilder handelt, sollte der Client die Daten als Blob an den Server verschicken:

```
function SendBlob()
{
  var blobData = document.querySelector
  ("input[type='file']").files[0];
  webSocketConnection.send(blobData);
}
```

Wenn der Client eine Nachricht schickt, während die Verbindung zum Server geschlossen ist, wirft die *send()*-Methode einen Fehler. Um solche Fehler zu vermeiden, haben Entwickler die Wahl, initial beim Eintreten des Ereignisses *Open* mit der Rückruffunktion *onopen* vom Client an den Server zu verschicken oder aber vor dem Aufruf der *send()*-Methode über die *readyState*-Eigenschaft wie in Listing 3 den Status der Verbindung zu prüfen (Tabelle 2).

Senden von großen Datenmengen

Beim Senden von großen Datenmengen werden Daten im Puffer des Browsers zwischengespeichert. Um den Zwischenspeicher abzufragen, bietet das WebSocket-API das Attribut *bufferedAmount* an. Hiermit haben Entwickler die Möglichkeit, die Bytes abzufragen, die zwischengespeichert und noch nicht an den Server verschickt wurden.

Das Ereignis *Message* kennzeichnet den Eingang einer Nachricht. Mit der passenden Rückruffunktion *onmessage* kann der Client mit den vom Server erhaltenen Daten weiterarbeiten. Vor dem Empfang der Daten haben Entwickler die Möglichkeit, die Datenauswahl einzuschränken, indem die Eigenschaft *binaryType* des WebSocket-Objekts auf *blob* oder *arraybuffer* gesetzt wird.

Diese Zuweisung ist keine Pflicht, jedoch sollten Anwendungsentwickler wissen, dass der Standardwert *blob* lautet. Diese Einstellung bedeutet nicht, dass der Client die Daten auch in dem erwarteten Format erhält. Deshalb sollten das Datenformat vor der Verarbeitung nochmals einer Prüfung unterzogen werden, um Fehler zu vermeiden. Der Empfang von Nachrichten wird durch das *onmessage*-Ereignis des WebSocket-Objekts erreicht (Listing 4).

Verbindung beenden

Der Verbindungsabbau kann sowohl vom Client als auch vom Server instanziert werden. Der Client kann die Verbindung über die Methode *close()* trennen. Der Server teilt dem Client den Verbindungsabbruch durch das *Close*-Ereignis mit. Der Client kann mit dem Ereignis *onclose* auf die serverseitige, aber auch die clientseitige Trennung reagieren, um zum Beispiel nachfolgende Prozesse anzustoßen:

```
webSocketConnection.onclose = function () {
  console.log("Verbindung wurde getrennt")
```

Tabelle 3: Statuscodes

Status	Beschreibung	Erläuterung
1000	CLOSE_NORMAL	Normales Beenden der Verbindung.
1001	CLOSE_GOING_AWAY	Entweder ist auf dem Server ein Fehler aufgetreten, der die Verbindung geschlossen hat, oder der Browser navigiert von der Seite weg, welche die Verbindung bereitstellt.
1002	CLOSE_PROTOCOL_ERROR	Ein Protokollfehler ist aufgetreten.
1003	CLOSE_UNSUPPORTED	Der Endpunkt hat Datentypen empfangen, die er nicht verarbeiten kann.
1004		Reservierter Statuscode für spätere Verwendung.
1005	CLOSE_NO_STATUS	Gibt an, dass kein Statuscode angegeben wurde, obwohl einer erwartet wurde.
1006	CLOSE_ABNORMAL	Die Verbindung wurde unerwartet geschlossen, obwohl ein Statuscode erwartet wurde.
1007	Unsupported Data	Der Endpunkt hat Daten in einem nicht unterstützten Format empfangen.
1008	Policy Violation	Der Endpunkt hat die Verbindung geschlossen, weil die erhaltene Nachricht das Regelwerk verletzt hat. Dieser Statuscode wird verwendet, wenn 1003 und 1009 nicht passen.
1009	CLOSE_TOO_LARGE	Der Endpunkt beendet die Verbindung, weil die Daten zu groß sind.
1010	Missing Extension	Der Client beendet die Verbindung, weil er eine oder mehrere Extensions mit dem Server aushandeln wollte, die der Server nicht kennt.
1011	Internal Error	Der Server befindet sich in einem unerwarteten Zustand, der ihn daran hindert, die Nachricht zu verarbeiten, und beendet die Verbindung.
1012	Service Restart	Der Server startet neu.
1013	Try Again Later	Die Verbindung wird vom Server beendet, weil er zum Beispiel derzeit zu viele Clientanfragen erhält.
1014		Reserviert
1015	TLS Handshake	Die Verbindung wird beendet, weil es einen Fehler im TLS Handshake gab (zum Beispiel bei einem nicht verifiziertem Zertifikat).

```
};
webSocketConnection.close();
```

Entwickler können der *Close()*-Methode zwei Parameter geben, zum einen einen Statuscode (Tabelle 3) und zum anderen einen Grund, warum der Client geschlossen hat.

Sicherheitshinweise

Um sichere Verbindungen aufzubauen, verwenden Entwickler SSL/TLS. Das heißt, die Entwickler nutzen wss:// anstatt ws://, also das Pendant zu https://, um eine verschlüsselte Verbindung zum Server aufzubauen.

WebSockets unterstützen eigentlich keine Authentifizierung und Echtheitsprüfung, was grundsätzlich erst einmal bedeutet, dass die WebSocket-Kommunikation keine Authentifizierungsdaten kennt. Es ist jedoch trotzdem möglich, die Authentifizierung durchzuführen, da WebSockets im Handshake dem Server einen Standard-HTTP-Header sendet, in dem eine implizite Authentifizierung über JavaScript beim Handshake verwendet werden kann.

Der WebSocket-Standard enthält die Definition, dass Browser einen Origin-Header mitschicken müssen. Jedoch sagt der Standard auch, dass unabhängige Clients von der Pflicht ausgeschlossen sind. Das heißt, dass ein unabhängiger Client diesen Wert verfälschen kann und der Server genau davon ausgehen kann. Es hört sich zwar trivial und selbstverständlich an, jedoch werden die Verifizierung und Validierung von Eingabedaten heutzutage immer noch übersehen und sollten somit weiterhin Beachtung finden, um zum Beispiel eine SQL-Injection zu verhindern.

Validierung der Daten

Nicht nur die Daten, die vom Client zum Server geschickt werden, sondern auch die Daten, die der Client vom Server erhält, sollten validiert und gegengeprüft werden. Tunneln von TCP-Diensten ist strengstens untersagt, und es ist eine Kommunikation über vorgelagerte Schichten zu wählen.

WebSockets werden in Echtzeitanwendungen verwendet, die geringe Latenzzeiten benötigen, wie zum Beispiel Online-Spiele, Chats, Finanzanwendungen, soziale Netzwerke–Feeds, kollaboratives Arbeiten, Sportticker und Lokalisierungsanwendungen. ■

Philippe Bénard
ist Senior Software Engineer bei der Adesso AG. Er verfügt über langjährige Berufserfahrung im Bereich der Entwicklung sowie der Analyse von Software-Applikationen. Seine Schwerpunkte sind Webapplikationen, Content-Management-Systeme und E-Commerce mit .NET, Konzeption, Analyse und Beschreibung von Schnittstellen und die Erstellung von serviceorientierten Architekturen und Backend-Systemen.
www.adesso.de

animago 20 YEARS AWARD & CONFERENCE 2016

3D ANIMATION ◆ VISUAL EFFECTS ◆ VISUALIZATION

TICKETS ON SALE NOW!

Images: animago AWARD & CONFERENCE 2015

Join us for our 20th anniversary!

- **animago AWARD & CONFERENCE on 27/28 October 2016**
- **For the first time ever in Munich's Culture Center GASTEIG**
- **EARLY BIRD ticket sale already started!**
  What are you waiting for? Get your ticket now on www.animago.com

Presented by: DIGITAL PRODUCTION

animagoAWARD
www.animago.com

Funded by: Bayerisches Staatsministerium für Wirtschaft und Medien, Energie und Technologie; Landeshauptstadt München Kulturreferat

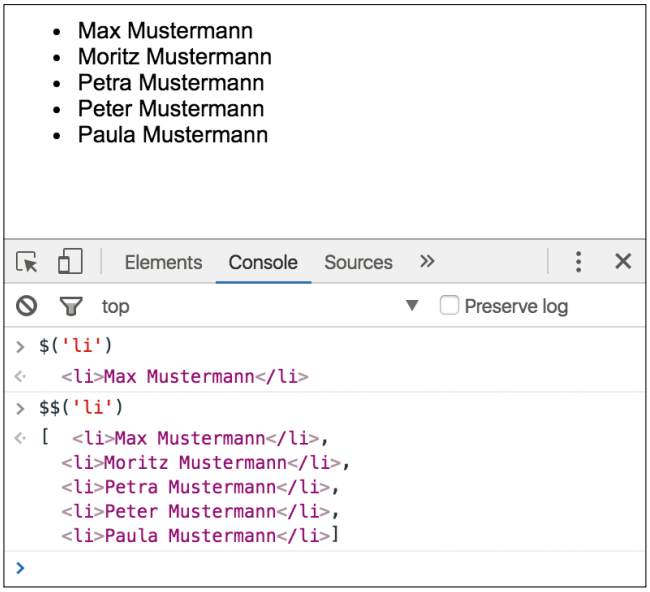
Sponsored by: AUTODESK, AVID, CHAOSGROUP, DELL, intel, Embassy of Canada, esri, fayteq technologies beyond imagination, MAXON A NEMETSCHEK COMPANY, NEC, PNY, VOGELSÄNGER, wacom

COMMAND LINE API

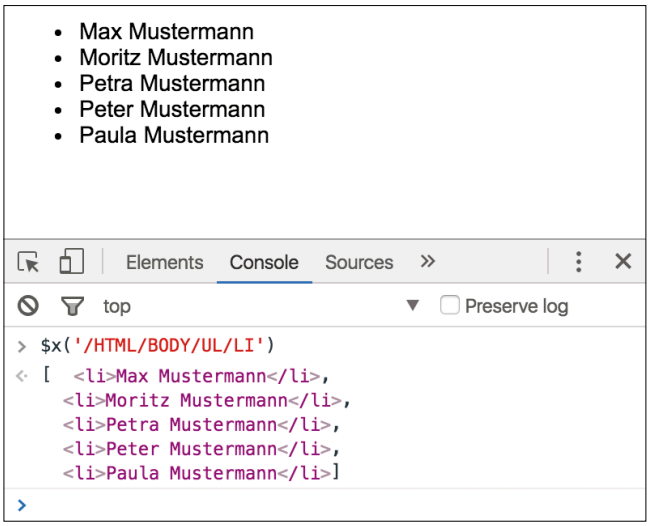
Entwickler-Tools

Das Command Line API stellt Funktionen von Browser-Tools per JavaScript zur Verfügung.

Browser-Tools wie die Chrome Developer Tools, Firebug oder die Entwickler-Tools von Safari gehören zu den Standardwerkzeugen eines jeden Webentwicklers. Sei es, um den DOM-Baum einer Webseite zu inspizieren, Ausgaben auf der Konsole zu erzeugen, Ausdrücke und Funktionsaufrufe auszuwerten oder um Breakpoints für das Debuggen zu definieren, den JavaScript-Code einer Webanwendung Schritt für Schritt durchzugehen, die Anwendung zu profilen und den Netzwerkverkehr zu beobachten.



Selektion mit den Funktionen *$()* und *$$()* (Bild 1)



Sogar die Selektion nach XPath-Ausdrücken ist möglich (Bild 2)

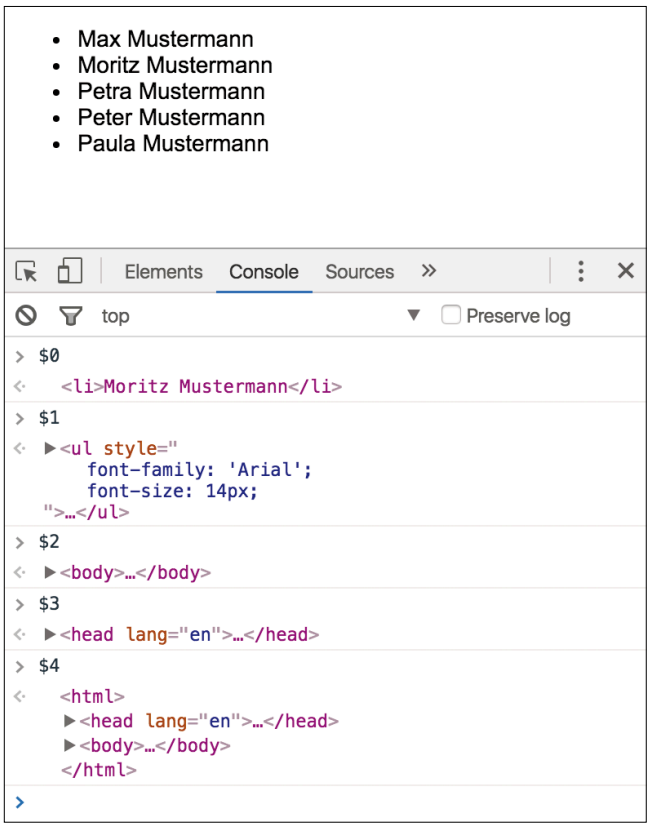
Dass diese Tools verschiedene Funktionalitäten auch per JavaScript zur Verfügung stellen (Tabelle 1), wissen allerdings nur die wenigsten Entwickler.

Eine gute Gelegenheit also, im Folgenden einmal einen kurzen Überblick über das relativ unbekannte Command Line API zu geben, das beispielsweise von Chrome (https://developer.chrome.com/devtools/docs/commandline-api), Safari (https://developer.apple.com/library/mac/documentation/AppleApplications/Conceptual/Safari_Developer_Guide/Console/Console.html) und Firefox (https://getfirebug.com/wiki/index.php/Command_Line_API) unterstützt wird.

Auswahl und Inspektion von DOM-Elementen

Für die meisten Webentwickler dürfte das *$*-Objekt ein alter Bekannter sein; schließlich wird standardmäßig – sofern eingebunden – die Funktionalität der jQuery-Bibliothek über diese Variable zur Verfügung gestellt.

Im Rahmen des Command Line API stellt *$* allerdings einen Alias für die Methode *document.querySelector()* dar (Bild 1). Der Aufruf *document.querySelector('li')* lässt sich also auch



Selektion der zuletzt selektierten DOM-Elemente (Bild 3)

ohne eingebundenes jQuery durch *$('li')* verkürzen und liefert das erste <*li*>-Element auf einer Webseite:

```
const firstListItem = $('li');
console.log(firstListItem);
```

Zu beachten ist dabei, dass für den Fall, dass eine Webseite jQuery einbindet, der Alias verständlicherweise nicht zur Verfügung steht (es sei denn, man hat jQuery einer anderen Variablen zugewiesen).

Analog zu *$()* stellt die Methode *$$()* einen Alias für die Methode *document.querySelectorAll()* dar. Der Aufruf *$$('li')* liefert beispielsweise alle Listenelemente auf der gesamten Webseite, der Aufruf *$('nav li')* nur diejenigen Listenelemente, die sich innerhalb eines <*nav*>-Elements befinden:

```
const allListItems = $$('li');
console.log(allListItems);
const allNavListItems = $$('nav li');
console.log(allNavListItems);
```

Besonders praktisch ist auch die Methode *$x()*, die Elemente für einen übergebenen XPath-Ausdruck liefert (Bild 2):

```
const listItems = $x('/HTML/BODY/UL/LI');
console.log(listItems);
```

▶

Tabelle 1: Übersicht über die bereitgestellten Methoden

Funktion	Beschreibung
$(selector)	Alias für *document.querySelector()*.
$$(selector)	Alias für *document.querySelectorAll()*.
$x(xpath)	Liefert die Elemente, die auf den übergebenen XPath zutreffen.
$0	Enthält den aktuell in der DOM-Baum-Ansicht selektierten Knoten.
$1..4	Enthält die zuletzt in der DOM-Baum-Ansicht selektierten Knoten.
$_	Enthält das Ergebnis des zuletzt ausgewerteten Ausdrucks.
clear()	Leert die Konsole.
copy(object)	Kopiert die String-Repräsentation eines Objekts in die Zwischenablage.
debug(function)	Veranlasst den Debugger, bei Aufruf der übergebenen Funktion anzuhalten.
dir(object)	Gibt alle Eigenschaften des übergebenen Objekts aus (Alias für *console.dir()*).
dirxml(object)	Gibt die XML-Repräsentation des übergebenen Objekts aus (Alias für *console.dirxml()*).
getEventListeners(object)	Liefert die für ein Objekt registrierten Event Listener zurück.
inspect(object/function)	Öffnet die DOM-Baum-Ansicht und selektiert dort das jeweilige Element, das übergeben wurde oder das von der übergebenen Funktion zurückgegeben wird.
keys(object)	Liefert ein Array zurück, das die Namen der Eigenschaften eines Objekts enthält.
monitor(function)	Sorgt dafür, dass bei Aufruf der übergebenen Funktion eine entsprechende Konsolenausgabe inklusive der jeweils übergebenen Argumente erzeugt wird.
monitorEvents(object[, events])	Startet das Monitoring von Events, die von einem Objekt ausgelöst werden.
profile([title])	Start den JavaScript-Profiler. Optional kann dabei ein Titel übergeben werden, der an entsprechender Stelle im Profiling-Report erscheinen soll.
profileEnd()	Stoppt den JavaScript-Profiler und erstellt einen entsprechenden Report.
table(data [, columns])	Gibt das übergebene Objekt in Tabellenform aus.
undebug(function)	Stoppt das Debuggen einer Funktion.
unmonitor(function)	Stoppt das Monitoring einer Funktion.
unmonitorEvents(object[, types])	Stoppt das Monitoring für die übergebenen Events am übergebenen Objekt.
values(object)	Liefert die Werte aller Eigenschaften eines Objekts als Array zurück.

Listing 1: Für ein Objekt registrierte Event Listener

```
window.addEventListener('load', function() {
  console.log('Dokument geladen');
});
window.addEventListener('resize', function() {
  console.log('Browserfenstergröße verändert');
});
console.log(getEventListeners(window));
```

Hat man über eine der Funktionen *$()*, *$$()* und *$x()* (oder über eine andere Selektionsmethode aus dem DOM-Standard) ein Element selektiert, lässt sich dieses über die Funktion *inspect()* des Command Line API in der DOM-Baum-Ansicht der jeweiligen Entwickler-Tools anzeigen:

```
const firstListItem = $('li');
inspect(firstListItem);
```

Über die Shortcuts *$0, $1, $2, $3* und *$4* lässt sich zudem auf die letzten fünf im DOM-Baum selektierten Elemente zugreifen (Bild 3).

Analyse von Events

Die Methoden *addEventListener()* und *removeListener()* dürfte jedem Webentwickler bekannt sein: Sie lassen sich bekanntermaßen dazu verwenden, um für einzelne Elemente Event Listener zu bestimmten Events zu registrieren beziehungsweise wieder zu entfernen. Es fehlt allerdings eine standardisierte Möglichkeit, um für ein Objekt an alle für ein bestimmtes Event registrierten Event Listener zu gelangen.

Das Command Line API definiert daher zusätzlich die Funktion *getEventListener()*, über die genau dies möglich ist (Listing 1). Übergibt man dieser Funktion ein Objekt, erhält man als Rückgabewert ein Objekt, das pro Event eine gleichnamige Eigenschaft enthält, in der wiederum jeweils ein Array mit den für das jeweilige Event registrierten Event Listener hinterlegt ist (Bild 4).

Debugging, Monitoring und Profiling

Breakpoints können bekanntermaßen direkt über das GUI des entsprechenden Entwickler-Tools definiert werden, oder alternativ auch über das Schlüsselwort *debugger*. Das Command Line API bietet darüber hinaus eine weitere Option (Listing 2): Die Methode *debug()* erwartet als Parameter eine Funktion oder Objektmethode und stoppt die Ausführung des jeweiligen Programms jedes Mal, wenn die übergebene Funktion oder Methode aufgerufen wurde (Bild 5).

Listing 2: Debugging einer Funktion

```
function createPerson(firstName, lastName, age) {
  return {
    firstName: firstName,
    lastName: lastName,
    age: age
  }
}
debug(createPerson);
let max = createPerson('Max', 'Mustermann', 55);
undebug(createPerson);
```

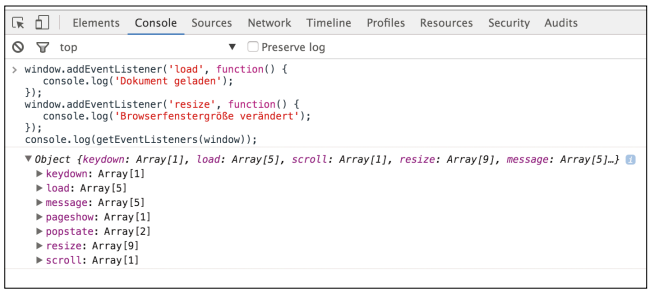
Soll dieses Debugging-Verhalten widerrufen werden, reicht ein einfacher Aufruf von *undebug()*, wobei auch hier die Funktion beziehungsweise Methode als Parameter übergeben werden muss.

Für den Fall, dass man lediglich wissen möchte, wann und mit welchen Argumenten eine Funktion oder Methode aufgerufen wird, stellt das Command Line API die Funktion *monitor()* zur Verfügung. Übergibt man dieser Funktion ein Funktionsobjekt als Argument, werden anschließend alle Aufrufe dieser Funktion entsprechend dokumentiert.

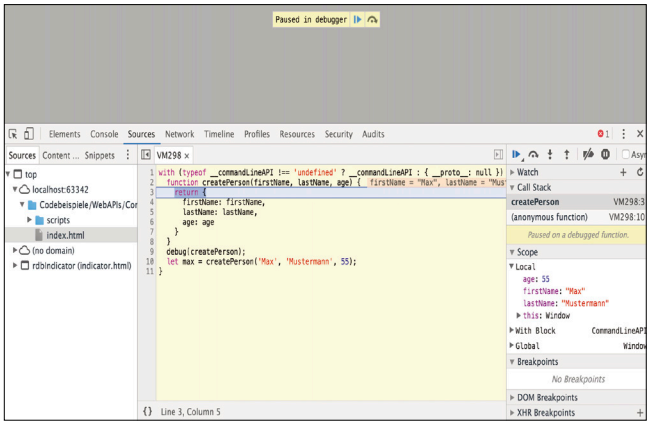
In Listing 3 beispielsweise wird über den Aufruf von *monitor(createPerson)* jeder Aufruf der Funktion *createPerson()* inklusive der jeweils übergebenen Argumente auf die Konsole ausgegeben (Bild 6).

Funktionen überprüfen

Dies ist besonders praktisch, um beispielsweise schnell zu überprüfen, ob eine Funktion nicht ungewollt mit falschen oder ungültigen Argumenten aufgerufen wird. Um das Monitoring zu einem späteren Zeitpunkt wieder zu stoppen, führt man analog die Funktion *unmonitor()* aus.



Zugriff auf die registrierten Event Listener eines Objekts (Bild 4)



Triggern der Debugging-Funktionalität (Bild 5)

Listing 3: Monitoring einer Funktion

```
function createPerson(firstName, lastName, age) {
  return {
    firstName: firstName,
    lastName: lastName,
    age: age
  }
}
monitor(createPerson);
let max = createPerson('Max', 'Mustermann', 55);
let moritz = createPerson('Moritz', 'Mustermann', 55);
// später:
unmonitor(createPerson);
```

Monitoring von Funktionsaufrufen (Bild 6)

Neben dem Monitoring von Funktionen und Methoden ist es auch möglich, das Auslösen von Events zu monitoren: Über die Funktion *monitorEvents()* lässt sich das Monitoring starten, über die Funktion *unmonitorEvents()* wieder stoppen. Als Argumente übergeben werden jeweils das Objekt, das die Events auslöst, sowie die Events, für die das Monitoring durchgeführt werden soll. Dabei lassen sich neben konkreten Event-Namen wie *mousedown*, *mouseup*, *keydown* und *keyup* auch die in Tabelle 2 gezeigten Event-Typen übergeben (Listing 4, Bild 7).

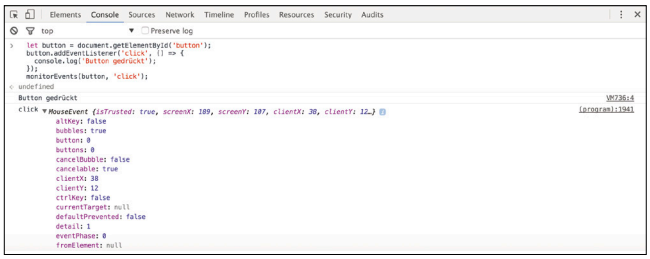
Läuft eine Webseite einmal nicht rund oder lässt die Performance zu wünschen übrig, bietet sich es sich an, über browserinterne Profiling-Tools entsprechende Messungen vorzunehmen. Auch diese Funktionalität wird durch das Command Line API zur Verfügung gestellt: So lässt sich über den Aufruf der Funktion *profile()* das Profiling starten und über den Aufruf von *profileEnd()* wieder stoppen, wobei als Argument jeweils ein Name für das zu erstellende Profil zu übergeben ist:

Tabelle 2: Ereignisse

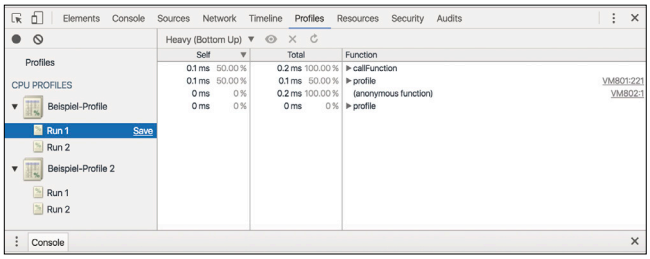
Event-Typ	Gemappte Events
mouse	*mousedown, mouseup, click, dblclick, mousemove, mouseover, mouseout, mousewheel*
key	*keydown, keyup, keypress, textInput*
touch	*touchstart, touchmove, touchend, touchcancel*
control	*resize, scroll, zoom, focus, blur, select, change, submit, reset*

Listing 4: Monitoring von Events

```
let button = document.getElementById('button');
button.addEventListener('click', () => {
  console.log('Button gedrückt');
});
monitorEvents(button, 'click');
// später:
unmonitorEvents(button, 'click')
```



Monitoring von Events (Bild 7)



Erstellen von Profilen (Bild 8)

```
profile('Beispiel-Profile 2');
profileEnd('Beispiel-Profile 2');
```

Anschließend findet sich das erstellte Profil im Reiter *Profile* der entsprechenden Entwickler-Tools (Bild 8).

Fazit

Das Command Line API ermöglicht es, auf Funktionalitäten von Browser-Entwickler-Tools zuzugreifen. Dabei stellt es zum einen diverse Shortcuts für DOM-Selektionsmethoden zur Verfügung, zum anderen verschiedene Helferfunktionen für das Debugging, das Monitoring und das Profiling. ■

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

FUNKTIONAL REAKTIVE PROGRAMMIERUNG IN JAVASCRIPT

Gut kombiniert

Die funktional reaktive Programmierung ist unter Webentwicklern gefragt.

Der Begriff der funktional reaktiven Programmierung (kurz FRP) dürfte den meisten Webentwicklern schon begegnet sein. Doch was verbirgt sich eigentlich hinter diesem Programmierparadigma? Dieser Artikel soll eine Einführung geben und anhand einiger Beispiele und der JavaScript-Bibliothek RxJS (https://github.com/Reactive-Extensions/RxJS) die Prinzipien erläutern.

Als Software- und Webentwickler kennt man mittlerweile eine ganze Reihe verschiedener Programmierparadigmen, sprich bestimmte Arten und Weisen der Programmierung, sowie bestimmte Arten und Weisen, wie Programme aufgebaut sind: sei es objektorientiert, funktional, aspektorientiert, imperativ oder ereignisorientiert.

In Ausgabe 7/2016 haben wir bereits einen Blick auf die objektorientierte Programmierung in JavaScript geworfen, in der vorigen Ausgabe 9/2016 einen Blick auf die funktionale Programmierung in JavaScript, und in der kommenden Ausgabe 11/2016 werden wir uns dem Thema der aspektorientierten Programmierung in JavaScript widmen.

In diesem Artikel dagegen geht es um die sogenannte funktional reaktive Programmierung, ein Programmierparadigma, das wiederum auf zwei anderen namensgebenden Paradigmen basiert: zum einen auf der funktionalen Programmierung, zum anderen auf der reaktiven Programmierung.

Die Prinzipien der funktionalen Programmierung dürften Ihnen noch aus der Ausgabe 9/2016 bekannt sein: Zum einen liegt hier der Fokus auf Funktionen und deren Wiederverwendung (etwa durch Komposition, partielle Auswertung oder Currying), zum anderen arbeiten Funktionen im besten Fall ohne Nebenwirkungen auf Datenstrukturen und liefern als Ergebnis neue Datenstrukturen zurück. Zum Einsatz kommen dabei Methoden wie *map(), filter()* oder *reduce()*.

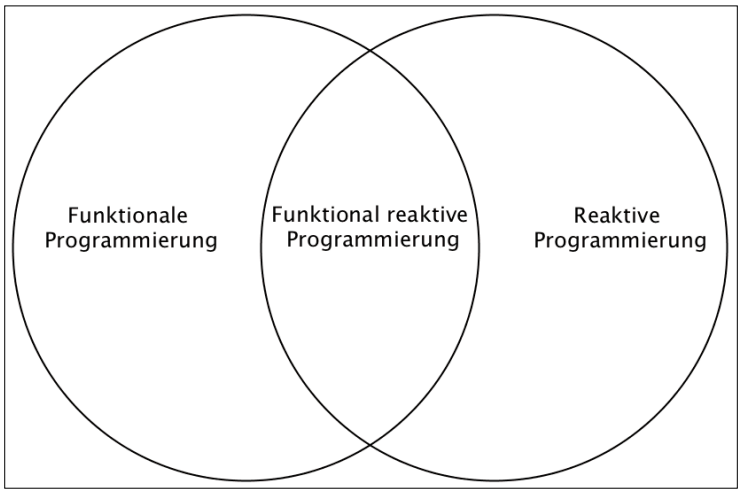
Listing 1: Funktionale Programmierung

```
'use strict';
const array = [
  '1', 'Max', '2', '3', '4', '5', 'IoT', '6', '7',
  '8', '9'
];
const result = array
  .map(x => parseInt(x))
  .filter(x => !isNaN(x))
  .reduce((x, y) => x + y);
console.log(result);
```

Bei der reaktiven Programmierung handelt es sich um ein Programmierparadigma, bei dem Zustandsänderungen innerhalb eines Softwaresystems über Datenflüsse beziehungsweise Datenströme an andere Komponenten des Systems propagiert werden. Dabei kommen in der Regel Entwurfsmuster wie das Observer-Pattern oder das Publish/Subscribe-Pattern zum Einsatz. Einzelne Komponenten registrieren sich als Observer an den Datenströmen (die auch als Observable bezeichnet werden), um über neue Daten informiert zu werden.

Funktionale Operationen

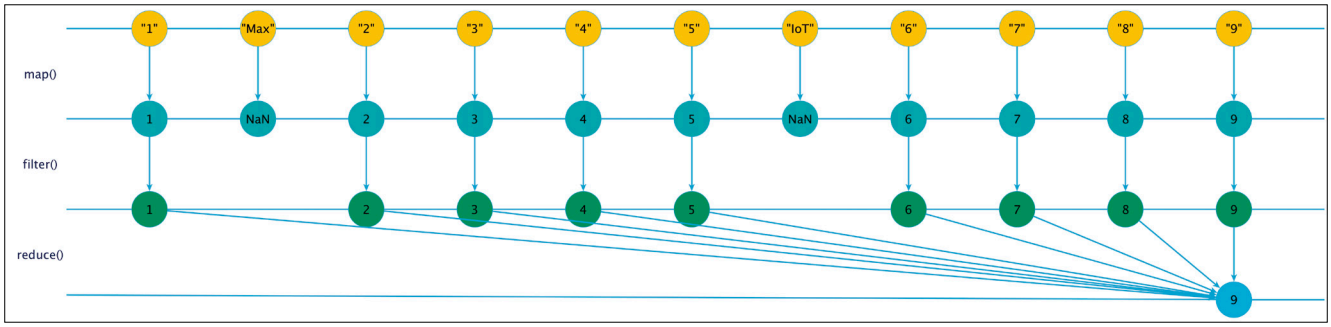
Die funktional reaktive Programmierung kombiniert nun die funktionale und die reaktive Programmierung (Bild 1), indem sich auf den genannten Datenströmen funktionale Operationen wie die genannten Methoden *map(), filter()* und *reduce()* anwenden lassen.



Zusammenhang zwischen den genannten Programmierparadigmen (Bild 1)

Ein API für die Programmierung mit Datenströmen existiert in Form von ReactiveX (http://reactivex.io). Implementierungen dieses API stehen für verschiedene Programmiersprachen wie Java, C#, Scala, Clojure, C++, Ruby, Python, Groovy, Swift und PHP zur Verfügung (siehe http://reactivex.io/languages.html).

Für JavaScript ist die Bibliothek RxJS eine der bekannteren Implementierungen von ReactiveX, die sowohl unter Node.js als auch im Browser verwendet werden kann. Bei Verwendung unter Node.js muss RxJS zunächst wie gewohnt über NPM mit Hilfe des Befehls *npm install rxjs* installiert werden. Möchte man RxJS auf Clientseite, also im Browser, verwenden, bietet sich alternativ die Installation mit Bower über den Befehl *bower install rxjs* an.



Die verschiedenen Datenströme für das Zahlenbeispiel (Bild 2)

Bevor wir uns ein erstes Beispiel für die funktional reaktive Programmierung mit RxJS ansehen, sei vorher noch ein Blick auf ein Beispiel der funktionalen Programmierung geworfen.

In Listing 1 sehen Sie dazu ein klassisches Beispiel für die Anwendung der Methoden *map()*, *filter()* und *reduce()*. Gegeben ist hier ein Array von Zeichenketten, von dem ausgehend über die Methode *map()* zunächst für jede Zeichenkette versucht wird, diese jeweils in eine Zahl umzuwandeln. Anschließend werden über *filter()* die Werte herausgefiltert, die keine Zahl sind (beziehungsweise für die die Funktion *parseInt()* zuvor zu dem Wert *NaN* führte), und schließlich über *reduce()* zu einem einzelnen Wert addiert.

In Listing 2 sehen Sie, wie man die gleiche Problemstellung mit Hilfe von RxJS und getreu dem funktional reaktiven Programmierparadigma implementiert.

Nachdem über *require()* zunächst das Node.js-Modul *rx* importiert und das Array mit Zeichenketten erstellt wurde, wird über den Aufruf *Rx.Observable.fromArray()* ausgehend von diesem Array ein Datenstrom erzeugt (Datenströme werden in RxJS beziehungsweise im API ReactiveX durch den Typ *Observable* repräsentiert).

Listing 2: Arrays

```
'use strict';
const Rx = require('rx');
const array = [
  '1', 'Max', '2', '3', '4', '5', 'IoT', '6', '7',
  '8', '9'
];
const stream = Rx.Observable.fromArray(array);
stream
  .map(x => parseInt(x))
  .filter(x => !isNaN(x))
  .reduce((x, y) => x + y)
  .subscribe(
    x => console.log(x),
    error => console.error(error),
    () => console.log('Fertig')
  )
;
```

Wie normale Arrays auch stellen Observables die Methoden *map()*, *filter()* und *reduce()* zur Verfügung, sodass der Code relativ ähnlich zu dem aus Listing 1 ist.

Über den Aufruf von *subscribe()* können dabei Observer (beziehungsweise Callback-Funktionen) für den Datenstrom registriert werden (Bild 2). Der erste Parameter ist ein Observer, der für den jeweils nächsten Wert im Datenstrom aufgerufen wird, der zweite Parameter ein Observer, der im Fehlerfall aufgerufen wird, und der dritte Parameter ein Observer, der aufgerufen wird, sobald die jeweilige Operation beendet wurde.

Im Beispiel wird der erste Observer nur einmal aufgerufen, da sich aufgrund der Anwendung von *reduce()* im resultierenden Datenstrom nur ein Wert befindet.

Unterschiede in der Funktionsweise

Auch wenn der Code der beiden Listings relativ ähnlich aussieht und auch in der funktional reaktiven Programmierung die Methoden *map()*, *filter()* und *reduce()* zur Verfügung stehen, gibt es einen wichtigen Unterschied bezüglich der Funktionsweise dieser Methoden: Bei der funktionalen Programmierung ist es so, dass die Methoden *map()*, *filter()* und *reduce()* synchron hintereinander ablaufen. Sprich, zunächst wird die Methode *map()* für das jeweilige Array ausgeführt und damit die entsprechende Callback-Funktion für jedes Element in diesem Array aufgerufen. Anschließend wird die Methode *filter()* für das Array ausgeführt (beziehungsweise die entsprechende Callback-Funktion für jedes Element im Array aufgerufen) und danach die Methode *filter()* (beziehungsweise wieder die entsprechende Callback-Funktion für jedes Element).

Bei dem Beispiel in Listing 2 (beziehungsweise der funktional reaktiven Programmierung) erfolgen die drei Schritte des Mappens, des Filterns und des Reduzierens jeweils pro Element. Mit anderen Worten: Zuerst wird das erste Element gemappt, gefiltert und reduziert (beziehungsweise auf einen Gesamtwert akkumuliert), dann das zweite, dann das dritte und so weiter. Dieser Ansatz skaliert wesentlich besser und spiegelt auch mehr die Tatsache wider, dass die Anzahl an Elementen in einem Datenstrom in der Regel nicht von vornherein feststeht.

Observables lassen sich dank entsprechender Helferfunktionen bequem auf Basis verschiedener Quellen erzeugen, sei dies nun wie gezeigt auf Basis von Arrays (Listing 2) oder ▶

Listing 3: Observable auf Basis einer Callback-Funktion

```
'use strict';
const Rx = require('rx');
function someAsyncFunction(callback) {
  callback(4711);
}
const someAsyncFunctionObservable =
Rx.Observable.fromCallback(someAsyncFunction);
const source = someAsyncFunctionObservable();
source.subscribe(x => console.log(x));
```

auf Basis von Callback-Funktionen (Listing 3), von Promises (Listing 4), von Ajax-Anfragen (Listing 5) oder auf Basis von Nutzerinteraktionen beziehungsweise Events (Listing 5). Eine vollständige Übersicht über die entsprechenden Helferfunktionen zeigt Tabelle 1.

Alle diese Datenquellen werden in der funktional reaktiven Programmierung als Datenströme gesehen, die zu unbestimmten Zeitpunkten Daten zur Verfügung stellen: Die Callback-Funktion oder das Promise-Objekt liefern ein Ergebnis, sobald die entsprechende asynchrone Funktion abgeschlossen wurde, Eingabefelder lösen durch Nutzereingaben Events aus, und bei Ajax-Anfragen werden die entsprechend registrierten Event Handler aufgerufen, sobald die Antwort vom Server bereitsteht.

Auf den Datenströmen, die durch die Observable-Objekte repräsentiert werden, können dann verschiedene Operationen beziehungsweise Methoden aufgerufen werden. Dazu zählen beispielsweise die schon im Eingangsbeispiel gezeigten Methoden *map(), filter()* und *reduce()*. Eine Reihe weiterer Operationen finden Sie in Tabelle 2 aufgelistet. Beispielsweise lassen sich zwei oder mehrere Datenströme zu einem Datenstrom zusammenfassen *(merge())*, Elemente innerhalb eines Datenstroms suchen *(find())*, Elemente überspringen *(skip())* und vieles mehr.

Praxisbeispiel: Drag and Drop

Zum besseren Verständnis sei im Folgenden gezeigt, wie sich die funktional reaktive Programmierung dazu einsetzen lässt, eine Drag-and-Drop-Funktionalität auf einer Webseite zu implementieren. Den entsprechenden Code sehen Sie in Listing 7. Was im Code passiert, ist der Übersicht halber in Bild 3 zusätzlich grafisch dargestellt.

Listing 4: Observable auf Basis eines Promises

```
'use strict';
const Rx = require('rx');
function someAsyncFunction() {
  return Promise.resolve(4711);
};
const source = Rx.Observable.
fromPromise(someAsyncFunction);
source.subscribe(x => console.log(x));
```

Listing 5: Observable auf Basis einer Ajax-Anfrage

```
'use strict';
const source = Rx.DOM.get('/users');
source.subscribe(
  (data) => {
    console.log(data.response);
  },
  (error) => {
    console.error(error);
  }
);

const source = Rx.DOM.post('/users', {
  firstName: 'Max',
  lastName: 'Mustermann'
});
source.subscribe(
  (data) => {
    console.log(data.response);
  },
  (error) => {
    console.error(error);
  }
);
```

Zunächst wird über *getElementById()* das Element selektiert, das durch die Drag-and-Drop-Operation verschoben werden soll (im Folgenden Drag-Element genannt). Anschließend werden über Aufrufe der Methode *Rx.Observab-*

Tabelle 1: Methoden zur Erstellung von Observables

Methode	Beschreibung
Rx.Observable. from(iterable)	Erstellt einen Datenstrom auf Basis eines iterierbaren Objekts.
Rx.Observable. fromArray([])	Erstellt einen Datenstrom auf Basis eines Arrays.
Rx.Observable. fromCallback(fn)	Erstellt einen Datenstrom auf Basis einer Callback-Funktion.
Rx.Observable. fromEvent(input, 'click')	Erstellt einen Datenstrom auf Basis eines UI-Events.
Rx.Observable. fromEvent(eventEmitter, 'event', fn)	Erstellt einen Datenstrom auf Basis eines Node.js-Events.
Rx.Observable. fromNodeCallback(fn)	Erstellt einen Datenstrom auf Basis einer Callback-Funktion im Node.js-Stil *(callback(error, result))*.
Rx.Observable. fromPromise(promise)	Erstellt einen Datenstrom auf Basis eines Promises.

Listing 6: Observable auf Basis eines Nutzer-Events

```
'use strict';
const input = document.querySelectorAll('input');
const source = Rx.DOM.fromEvent(input, 'click');
source.subscribe(
  (x) => {
    console.log('Button geklickt');
  },
  (error) => {
    console.log('Error: ' + err);
  },
  () => {
    console.log('Completed');
  }
);
```

Tabelle 2: Übersicht über verschiedene Operatoren

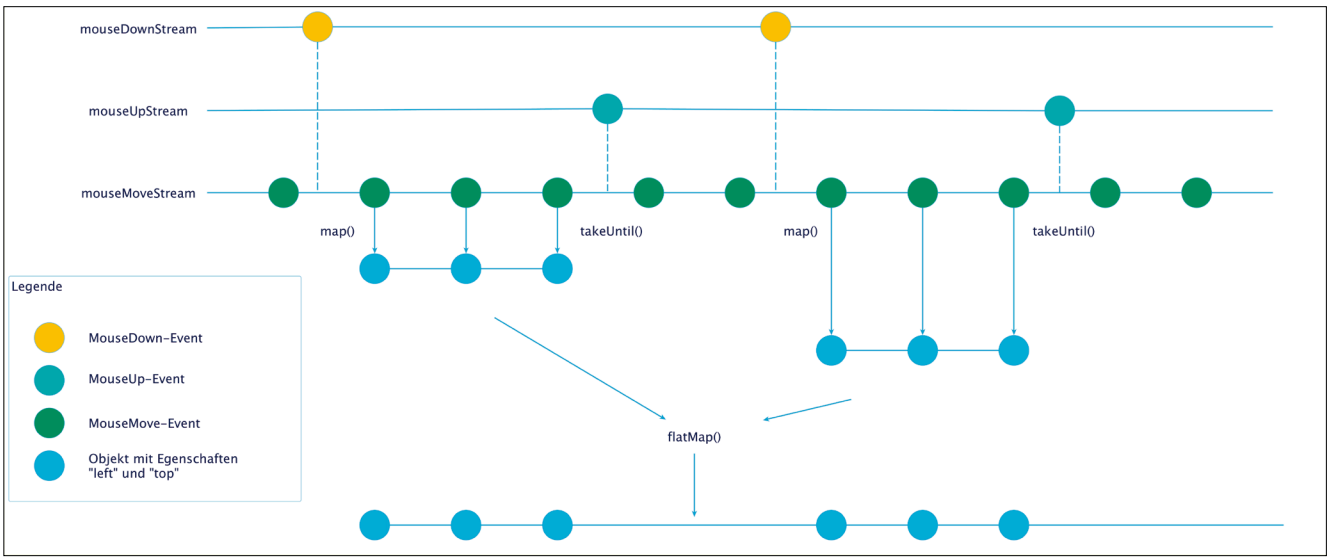
Operator	Beschreibung
concat()	Konkateniert eine Reihe von Datenströmen.
count()	Zählt die Elemente in einem Datenstrom, die ein bestimmtes Kriterium erfüllen.
delay()	Verzögert einen Datenstrom um eine bestimmte Zeit.
every()	Prüft, ob alle Elemente in einem Datenstrom ein bestimmtes Kriterium erfüllen.
filter()	Filtert die Elemente in einem Datenstrom anhand eines bestimmten Kriteriums.
find()	Sucht nach einem Element, das ein bestimmtes Kriterium erfüllt.
first()	Sucht nach dem ersten Vorkommen eines Elements, das ein bestimmtes Kriterium erfüllt.
flatMap()	Mappt jedes Element eines Datenstroms in einen neuen Datenstrom und fügt die resultierenden Datenströme zusammen.
last()	Sucht nach dem letzten Vorkommen eines Elements, das ein bestimmtes Kriterium erfüllt.
map()	Mappt jedes Element in einem Datenstrom auf ein neues Element.
reduce()	Akkumuliert die Elemente in einem Datenstrom zu einem einzelnen Wert.
skip()	Überspringt eine bestimmte Anzahl an Elementen in einem Datenstrom.
some()	Prüft, ob ein oder mehr Elemente in einem Datenstrom ein bestimmtes Kriterium erfüllen.
takeUntil()	Liefert die Elemente eines Datenstroms so lange, bis der übergebene Datenstrom ein Element erzeugt.
takeWhile()	Liefert die Elemente eines Datenstroms so lange, wie ein bestimmtes Kriterium erfüllt ist.

Listing 7: Drag and Drop

```
'use strict';
function init() {
  let dragTarget = document.getElementById
  ('drag-target');
  // Datenstrom für das mouseup-Event
  let mouseUpStream = Rx.Observable.fromEvent(
    document, 'mouseup'
  );
  // Datentrom für das mousemove-Event
  let mouseMoveStream = Rx.Observable.fromEvent(
    document, 'mousemove'
  );
  // Datenstrom für das mousedown-Event
  let mouseDownStream = Rx.Observable.fromEvent(
    dragTarget, 'mousedown'
  );
  mouseDownStream
    .flatMap(mouseDownEvent =>
      mouseMoveStream
        .map(mouseMoveEvent => {
          mouseMoveEvent.preventDefault();
            return {
              left: mouseMoveEvent.clientX -
                mouseDownEvent.offsetX,
              top: mouseMoveEvent.clientY -
                mouseDownEvent.offsetY,
            }
          })
        .takeUntil(mouseUpStream)
      )
    .subscribe(position => {
      console.log(
        'Relative Position:',
        'links:', position.left,
        'oben:', position.top
      );
    });
}

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

le.fromEvent() drei Datenströme erzeugt: einer für das *mousedown*-Event auf dem Drag-Element, einer für das *mousemove*-Event auf dem Document-Objekt und einer für das *mouseup*-Event auf dem Document-Objekt. Mit anderen Worten: Tritt ein *mousedown*-Event auf dem Drag-Element auf oder ein *mouseup*-Event oder ein *mousemove*-Event auf dem Document-Objekt, so wird dieses auf dem jeweiligen Datenstrom weitergeleitet.

Sobald nun auf dem Datenstrom für die *mousedown*-Events (auf dem Drag-Element) ein Ereignis auftritt, werden alle Ereignisse, die auf dem *mousemove*-Datenstrom ausgelöst werden, jeweils in ein Objekt gemappt, das die Pixelabstände ▶



Die verschiedenen Datenströme für das Drag-and-Drop-Beispiel (Bild 3)

der aktuellen Mauszeigerposition zur Ausgangsposition enthält. Dies wird so lange ausgeführt, bis auf dem *mouseup*-Datenstrom ein Ereignis auftritt, das wiederum über die Methode *takeUntil()* abgefangen werden kann.

Die einzelnen Datenströme, die hierdurch erzeugt werden (in Bild 3 exemplarisch zwei Datenströme), werden anschließend durch den Aufruf *flatMap()* zu einem einzelnen Datenstrom zusammengefasst.

Die Bibliothek RxJS kann durch verschiedene Module erweitert werden. Eines dieser Module ist RxJS-DOM (https://github.com/Reactive-Extensions/RxJS-DOM), die beispielsweise zusätzliche Methoden zur Verfügung stellt, um Datenströme auf Basis von Server-Sent-Events *(Rx.DOM.fromEventSource())*, von Web Workers *(Rx.DOM.fromWorker())*, von Mutation Observern *(Rx.DOM.fromMutationObserver())* oder auch von WebSockets *(Rx.DOM.fromWebSocket())* zu erzeugen.

Ein Beispiel für die Verwendung letzterer Methode ist im Folgenden gezeigt. Das Ziel ist, Daten per WebSocket-Verbindung von einem Socket-Server abzurufen und diese (mehr oder weniger in Echtzeit) mit Hilfe von D3.js (https://d3js.org)

Listing 7: Der Code für den WebSocket-Server

```
...
let wsServer = new WebSocketServer({
  httpServer: server,
  autoAcceptConnections: false
});

wsServer.on('request', (request) => {
  let connection = request.accept(
    'echo-protocol',
    request.origin
  );
  setInterval(() => {
    connection.sendUTF(Math.floor(Math.random() *
    100) + 1);
  }, 200);
  connection.on('close', (reasonCode, description)
=> {
    console.log((new Date()) + ' Peer ' +
      connection.remoteAddress + 'disconnected.'
    );
  });
});
```

Links zum Thema

- Das Reaktive-Manifest
  www.reactivemanifesto.org
- ReactiveX: API für asynchrone Programmierung mit Datenströmen
  http://reactivex.io
- RxJS, eine Implementierung des ReactiveX-API für JavaScript
  https://github.com/Reactive-Extensions/RxJS
- Operatoren in RxJS
  https://github.com/Reactive-Extensions/RxJS/blob/master/doc/api/core/observable.md#observable-instance-methods
- Interaktive Diagramme zur funktional reaktiven Programmierung
  http://rxmarbles.com
- Aufgaben zum Erlernen der funktional reaktiven Programmierung
  https://github.com/Reactive-Extensions/RxJSKoans
- Einführung in die reaktive Programmierung
  https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

Listing 8: Der Code für den WebSocket-Client

```
'use strict';
function init() {
  const MAX_DATAPOINTS = 150;
  let chart;
  let values = [];
  let chartData = [{
    values: values,
    key: 'Datenstrom',
    color: '#ff7f0e'
  }]
  const openObserver = Rx.Observer.create((event) => {
    console.info('Socket geöffnet');
  });
  const closingObserver = Rx.Observer.create(() => {
    console.log('Socket geschlossen');
  });
  const stream = Rx.DOM.fromWebSocket(
    'ws://localhost:3000',
    'echo-protocol',
    openObserver,
      closingObserver
  );
  stream.subscribe(
    (event) => {
      values.push({
        x: new Date(),
        y: parseFloat(event.data)
      });
      if (values.length > MAX_DATAPOINTS)
      {
        values.shift();
      }
      chart.update();
    },
    (error) => {
      console.error('error: %s', error);
    },
    () => {
      console.info('Socket geschlossen');
    }
  );
  const chartDataStream = Rx.Observable.from(values);
  chartDataStream.subscribe(
    (event) => {
      if(typeof chart !== 'undefined') {
        chart.update();
      }
    }
  );
  nv.addGraph(() => {
    chart = nv.models.lineChart()
      .interpolate('basis')
      .margin({left: 100})
      .showLegend(true)
      .showYAxis(true)
      .showXAxis(true);
    chart.xAxis
     .axisLabel('Time (ms)'
     .tickFormat(d3.format('.02f'));
    chart.yAxis
      .axisLabel('Voltage (v)')
      .tickFormat(d3.format('.02f'));
    d3.select('#chart svg')
      .datum(chartData)
      .call(chart);
    nv.utils.windowResize(() => {
      chart.update()
    });
    return chart;
  });
}
document.addEventListener('DOMContentLoaded', init);
```

und NVD3.js (http://nvd3.org) in einem Liniendiagramm darzustellen. Einen Ausschnitt des entsprechenden Codes für die Serverseite sehen Sie in Listing 7: Hier wird ein Socket-Server gestartet, der alle 200 Millisekunden eine zufällig generierte Zahl zwischen *1* und *100* an den jeweiligen Socket-Client sendet.

Der Code für die Clientseite ist in Listing 8 zu sehen. Über die Methode *Rx.DOM.fromWebSocket()* wird zunächst ein Datenstrom auf Basis eines WebSockets erzeugt.

Jedes Mal, wenn der Socket-Server nun eine Zufallszahl generiert und dadurch eine entsprechende Nachricht auf dem Datenstrom erzeugt wird, wird der registrierte Observer aufgerufen und damit ein entsprechender Datenpunkt (bestehend aus Zufallszahl und aktuellem Zeitpunkt) für die Darstellung im Graphen generiert sowie dem Array *values* hinzugefügt.

Für Letzteres wird dabei ebenfalls ein Datenstrom erzeugt, sodass jedes Hinzufügen eines Datenpunkts in das Array an den am Datenstrom registrierten Observer weitergeleitet wird. Der Aufruf *chart.update()* innerhalb des Observers sorgt dann für eine Aktualisierung des Liniendiagramms. ■

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

WEBSITE REDIRECT MANAGEMENT

Website-Relaunch

Ohne richtige Planung und korrekte Weiterleitungen greift auch das beste Relaunch-Konzept nicht.

Die Ziele beim Relaunch einer Unternehmenswebsite sind oft hoch gesteckt: bessere Performance und Bedienbarkeit, mehr Umsatz, nicht zuletzt ein moderneres Design. Vorübergehende leichte Verluste im Ranking und Traffic nehmen dabei viele bewusst in Kauf; oft genug wird aus Unwissenheit und schlechter Planung jedoch auch ein dramatischer Einbruch riskiert. Dabei können Verluste mit der richtigen Planung und korrekten Weiterleitungen vermieden werden.

Weiterleitungen rechtzeitig planen

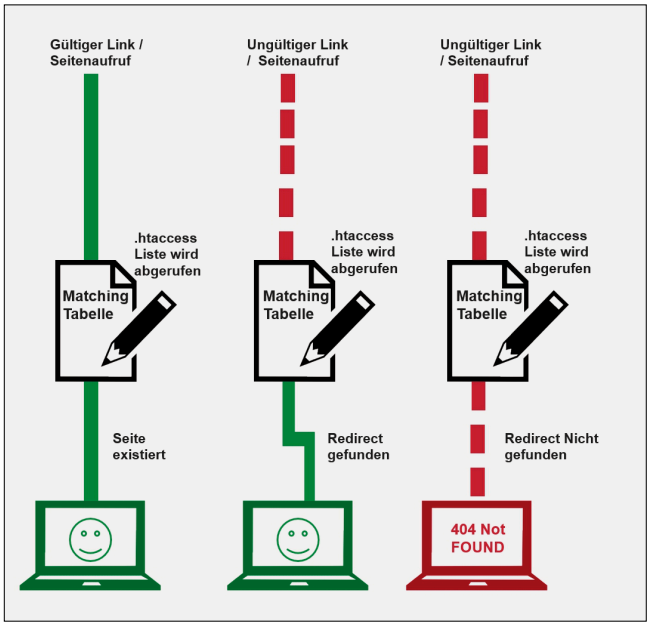
Wohl jede Checkliste für einen Wohnungsumzug enthält die Punkte Nachsendeauftrag und Ummeldungen. Schließlich ist klar, dass man die Adressänderung anderen auch mitteilen muss, um erreichbar zu bleiben und Informationen und wichtige Unterlagen zuverlässig zu erhalten.

Wird eine Website neu aufgesetzt, ändert sich viel mehr als nur eine Adresse. Es gibt zahllose neue URLs, und externe Verlinkungen auf bestehende Unterseiten werden ungültig und laufen ins Leere. Bedacht wird das längst nicht immer. Ein Relaunch ist aufwendig, es wird über Struktur, Inhalte und Design diskutiert, unterschiedliche Interessen sind zu berücksichtigen. Zwischen Meetings und Testläufen kann das Thema Redirects leicht aus dem Blickfeld geraten. Leider schiebt es sich oft erst dann prominent in den Vordergrund, wenn es an die Messung der Ergebnisse geht: Nach dem Relaunch brechen Positionen im organischen Ranking und damit Traffic und Umsatz regelrecht weg.

Was ist passiert? Links, die auf die ursprünglichen Seiten verweisen, funktionieren nicht mehr. Nutzer erhalten die Fehlermeldung »404 Seite nicht gefunden« – und springen häufig ab, der Traffic sinkt. Aber auch Suchmaschinen finden die Seiten zunächst nicht mehr und sortieren sie folgerichtig aus ihren Suchergebnissen. Werden die neuen Seiten von Suchmaschinen gefunden und bewertet, sind die Rankings häufig schlechter als mit den alten Inhalten.

Es fehlt ihnen die Suchmaschinenhistorie – die Information, wie häufig sie in der Vergangenheit aufgerufen, verlinkt und genutzt wurden. Im Ergebnis landet eine neue Seite, selbst wenn sie inhaltlich unverändert oder besser ist, nicht selten im Ranking viel weiter unten, und es dauert sehr lange, bis sie sich den ursprünglichen Platz wieder erarbeitet hat.

Das lässt sich vermeiden, indem man Weiterleitungen, sogenannte Redirects, nutzt. Für dauerhafte Weiterleitungen wie im Fall eines Relaunchs nutzt man eine 301-Umleitung. Eine 302-Weiterleitung dagegen signalisiert nur eine vorübergehende Änderung und ist nur seltenen Fällen sinnvoll. Beispielsweise dann, wenn ein Produkt vorübergehend nicht verfügbar ist.



Die Redirect-Verwaltung per *.htaccess*-Datei (Bild 1)

Mit Hilfe von Redirects kann der Großteil der ursprünglichen Linkstärke (verschiedene Untersuchungen gehen von 90 bis 98 Prozent aus) und Seitenhistorie auf die neue Seite übertragen werden. Somit ist der wichtigste Schritt tatsächlich, rechtzeitig an die Umleitungen zu denken. Alles Weitere lässt sich planen.

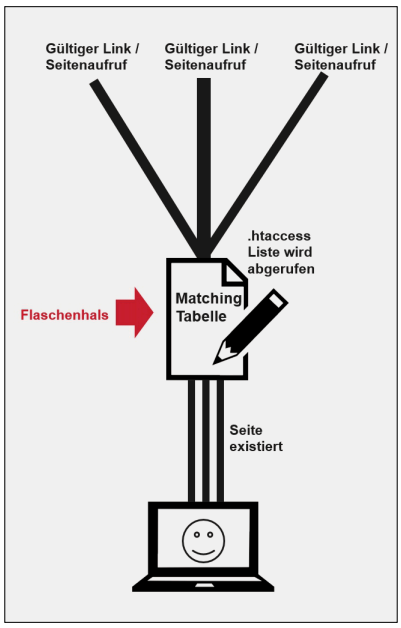
Seiten sorgfältig erfassen und zuordnen

Im zweiten Schritt sind Sorgfalt und Fleiß angesagt. Für eine optimale Weiterleitung sollte jeder früheren Seite eine neue Seite zugeordnet werden. Bei Online-Shops ist das häufig automatisiert oder semiautomatisiert möglich, indem alte und neue Artikelseiten maschinell ausgelesen und abgeglichen werden – etwa mit einem Abgleich von statischen Artikelnummern. Werden Inhalte nur leicht angepasst oder sogar unverändert übernommen, lassen sich diese auch auf anderen Seiten häufig mit maschineller Unterstützung zuordnen.

Bei einer kompletten Umstrukturierung der Seiten und der Inhalte stößt das Verfahren jedoch an seine Grenzen. Da aber hier schon in der Konzeptphase bekannt ist, welche Inhalte mit welchem Ziel geändert oder verschoben werden, empfiehlt es sich, dies auch während der gesamten Überarbei-

tung nachvollziehbar zu dokumentieren. Daraus ergibt sich dann die passende Weiterleitung. Darüber hinaus gilt es einige Punkte zu beachten:

- Bevor Seiten wegfallen, sollte immer zuerst deren Traffic und Ranking geprüft werden. Zeigt sich, dass die Seiten noch genutzt werden, sollten sie im neuen Stil aufbereitet werden oder zumindest über eine Archivfunktion erhalten bleiben.
- Es ist möglich, mehrere alte Seiten einer neuen zuzuordnen. Schwierig wird es im umgekehrten Fall, wenn eine bestehende Seite aufgesplittet wird. Da nicht einzelne Abschnitte einer Seite den jeweils neuen Seiten zugeordnet werden können, kann es nur einen Alleinerben geben. Es ist dann zu entscheiden, welche der neuen Seiten die Linkstärke der ursprünglichen Seite bekommen soll.
- Die Seiten sollten passend weitergeleitet werden, es sollte also nicht einfach im großen Stil auf die Home-Seite verlinkt werden, wenn es einen bestimmten Artikel nicht mehr gibt. Für den Nutzer ist das frustrierend, und auch bei Google wird dies als Soft-404 bewertet und somit einem Wegfall der Seite quasi gleichgesetzt.



Bei hoher Seitenzahl und Frequentierung leidet die Performance (Bild 2)

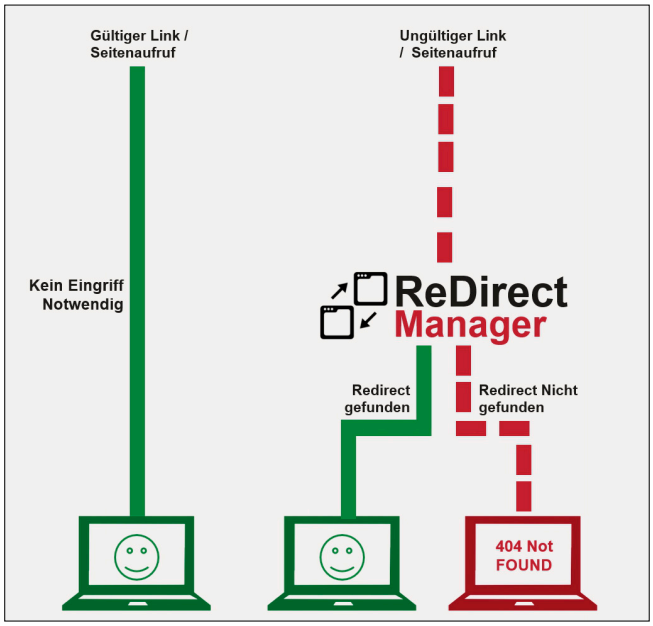
Traditionell werden Redirect-Regeln in einer Matching-Tabelle in der *.htaccess*-Datei auf dem Server hinterlegt und bei jedem Seitenaufruf abgearbeitet (Bild 1). Je nachdem, welches Content-Management-System verwendet wird, ist eine entsprechende oder ähnliche Weiterleitungsfunktion bereits integriert. In aller Regel bedeutet dies jedoch, dass bei jedem Aufruf zunächst die Matching-Tabelle vollständig abgeglichen wird – unabhängig davon, ob der Nutzer einen gültigen, ungültigen oder weitergeleiteten URL aufrufen möchte. Enthält die Liste nur wenige Seiten und ist die Zahl der Nutzer und somit der Zugriffe eher gering, so funktionieren Weiterleitungen und Seite meist problemlos.

Lange Ladezeiten vermeiden

Anders ist dies bei einer großen Anzahl von Seiten und Weiterleitungen, gepaart mit hohen Zugriffszahlen. Die Weiterleitung ist auch hier gewährleistet – aber es dauert. Bei größeren Webauftritten kann eine Matching-Tabelle durchaus mehrere Tausend Einträge umfassen. Je höher die Seitenzahl und die Frequentierung, desto mehr leidet die Performance, wenn bei jeder Anfrage erst die gesamte Liste abgearbeitet werden muss. Das führt zu langen Ladezeiten. Nutzer sind frustriert und springen häufig ab. Im Extremfall kann gar der Server zusammenbrechen. Dem ließe sich durch eine Aufstockung der Serverkapazitäten vorbeugen, das ist jedoch teuer (Bild 2).

Eine einfache Lösung bietet der ReDirect Manager (Bild 3). Dabei handelt es sich um eine von Goldbach Interactive (Germany) AG entwickelte Softwarelösung, die Redirects in hoher Zahl ermöglicht. Kann eine Unterseite nicht erreicht werden, wird das Tool automatisch aufgerufen und leitet die Anfrage weiter. Dabei wird zunächst ein kurzer Code auf dem eigenen 404-Template eingebunden. Die Zuordnungstabelle liegt auf einem sicheren, externen Server.

Der entscheidende Unterschied ist: Die Matching-Liste wird nur noch im Bedarfsfall abgerufen, also nur dann, wenn ein 404-Fehler ausgemacht wird. Nur in diesem Fall wird der Aufruf auf den externen Server umgeleitet und mit der Tabelle abgeglichen. Der Nutzer bemerkt davon nichts, er wird direkt auf die in der Liste aufgeführte neue Seite geführt. Alle gültigen Aufrufe dagegen führen ohne Umweg zur bestehenden Seite. Engpässe und lange Ladezeiten werden so vermieden. Mit einer zusätzlichen Reporting- und Optimierungsfunktion können als 404 einlaufende URLs verfolgt, nachträglich in die Mapping-Liste eingepflegt und in Redirects umgewandelt werden. ■



Der ReDirect Manager ermöglicht Redirects in hoher Zahl (Bild 3)

Christoph Spannagel
ist Head of Search Engine Optimization bei der Goldbach Interactive AG. Das Unternehmen sieht sich als Pionier-Agentur in Sachen Suchmaschinen-Werbung und Suchmaschinen-Optimierung.
www.goldbachinteractive.de

IOS SPRITEKIT SCENE EDITOR

Szene machen

Mit dem SpriteKit Scene Editor lassen sich Szenen in Spielen visuell entwickeln.

SpriteKit ist die Bibliothek, um Spiele für iOS, tvOS, watchOS und macOS zu schreiben. Apple unterstützt den Entwickler mit vielen nützlichen Funktionen und Klassen, die diesem viel (Programmier-)Arbeit abnehmen. Mit der Klasse *SKSpriteNode* lässt sich beispielsweise ein grafisches Objekt ableiten, das Methoden zur Positionierung und Animation von Grafiken enthält.

Diese Klasse ist aber nicht das einige Werkzeug, das in der Bibliothek enthalten ist. Die Klasse *SkScene* ermöglicht es, einen kompletten Rahmen für eine (Spiel-)Szene zu erstellen. Einer Instanz dieser Klasse können unterschiedliche SKSpriteNodes zugewiesen werden und diese sind dann Bestandteil der Szene. Beispielsweise für einen oder mehrere Hintergründe und ein oder mehrere Sprites für die Spielfiguren.

Objekte aktualisieren

Objekte der Klasse *SkScene* beziehungsweise deren untergeordnete Nodes (zum Beispiel Objekte der Klasse *SKSpriteNode)* können durch Aufruf der Methode *update* aktualisiert werden. So ist es beispielsweise möglich, innerhalb der Methode die Position der SKSpriteNode-Objekte zu ändern.

Verwendet man die in SpriteKit integrierte Physik-Engine, so können durch einen Aufruf der Methode *didSimulatePhysics* innerhalb einer Szene physikalische Anweisungen durchgeführt werden. Dieser Komfort hat aber seinen Preis: Es muss viel codiert werden, wenn auf Basis von SpriteKit entwickelt wird. Oder gibt es vielleicht eine Alternative?

Listing 1: Die Klasse GameViewController

```
import UIKit
import SpriteKit

class GameViewController: UIViewController {

  override func viewDidLoad() {
    super.viewDidLoad()

  if let scene = GameScene(fileNamed:"GameScene") {

    let skView = self.view as! SKView
    skView.showsFPS = true
    skView.showsNodeCount = true
    skView.ignoresSiblingOrder = true
    scene.scaleMode = .AspectFill
    skView.presentScene(scene)
  }
}
```

Visuell statt textuell

Apple hat die SpriteKit-Bibliothek seit der ersten Vorstellung stark weiterentwickelt. Aber nicht nur die Bibliothek wurde erweitert, auch die Unterstützung durch Xcode bei der Entwicklung von Spielen wurde vorangetrieben. So hat Apple mittlerweile in seine Entwicklungsumgebung einen visuellen Editor für Spiele integriert. Dieser funktioniert ähnlich wie der Interface Builder beziehungsweise das Storyboard für normale Apps. Objekte, beispielsweise ein Sprite, können visuell im Scene Editor von Xcode abgelegt und dann im Code angesprochen werden.

Viele Parameter lassen sich so sehr schnell visuell konfigurieren. Das Schreiben von Code ist zwar weiterhin erforderlich, allerdings erspart diese Vorgehensweise doch die ein oder andere Zeile und geht relativ schnell von der Hand.

Ausgangspunkt für das Beispiel ist ein neues Projekt, das auf Basis des Game-Templates von Xcode erzeugt wird. Ein solches Projekt besteht neben den Standardkomponenten (zum Beispiel LaunchScreen, Storyboard) noch aus weiteren Dateien. Als Erstes ist da der klassische ViewController, der unter der Bezeichnung *GameViewController* im Projekt angelegt wird. Neben den alten Bekannten gibt es aber noch neue Dateien: *GameScene.sks* und *GameScene.swift*.

Was das Storyboard für Standard-Apps ist, ist die *GameScene.sks* für Spiele. Neben der SKS-Datei gibt es auch noch zusätzlich eine Codedatei *(GameScene.swift)*, die den Programmcode zur Spielszene enthält.

Nachdem das Projekt angelegt wurde, wird innerhalb der Klasse *GameViewController* der Code zum Erzeugen einer SpritkeKit-Scene eingefügt (Listing 1).

Den automatisch durch das Template erzeugten Quellcode in dieser Datei können Sie löschen und durch den Code in Listing 2 ersetzen. Innerhalb der Methode *viewDidLoad* wird im ersten Schritt eine Instanz der Klasse *GameScene* erzeugt. Anschließend wird ein Objekt der Klasse *SKView* angelegt. Eine Instanz dieser Klasse wird benötigt, um Inhalte, die auf SpriteKit basieren, anzeigen zu können.

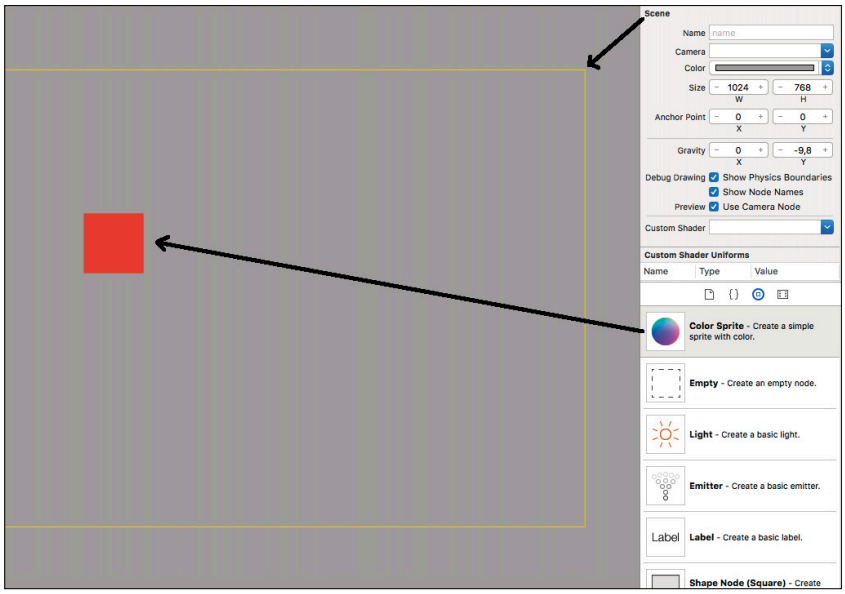
Im Beispiel werden noch einige Optionen aktiviert (zum Beispiel *showFPS)*, die für die spätere App nicht mehr benötigt werden und nur zu Testzwecken vorhanden sind. Steht diese Klasse, so kann man sich im nächsten Schritt um die Entwicklung der (Spiel-)Szene kümmern. Hierfür wird der Scene Editor von Xcode geöffnet. Die Auswahl der entsprechenden SKS-Datei im Project Navigator von Xcode aktiviert den passenden Editor in Xcode.

Listing 2: Initialisierung und erste Bewegung

```
import SpriteKit

class GameScene: SKScene, SKPhysicsContactDelegate  {

  //Spieler
  var player: SKSpriteNode?
  let playerSpeed: CGFloat = 150.0

  //Hintergründe
  var background: SKSpriteNode?
  var backgroundwater: SKSpriteNode?

  //Letzte bekannte Position
  var lastTouch: CGPoint? = nil

  override func didMoveToView(view: SKView) {
  physicsWorld.contactDelegate = self
  player = self.childNodeWithName(
    "player") as? SKSpriteNode

  background = self.childNodeWithName(
    "background") as? SKSpriteNode
  backgroundwater = self.
  childNodeWithName("backgroundwater")
      as? SKSpriteNode
    mickDefaultAnimation()
  }

  private func mickDefaultAnimation() {
  let wait: SKAction = SKAction.
    waitForDuration(0.1)
  let changeImage1 = SKAction.
    setTexture(SKTexture(imageNamed:
      "mickdefault1"))

  let changeImage2 = SKAction.setTexture(
  SKTexture(imageNamed: "mickdefault2"))
  let moveForever = SKAction.
    repeatActionForever(SKAction.
      sequence([wait, changeImage1,
        wait, changeImage2]))
  player!.runAction(moveForever)
  }
...

}
```

Öffnet sich der Scene Editor von Xcode, so sieht man außer einer grauen Fläche erst einmal nichts (Bild 1). Bei genauerer Betrachtung fällt der rechteckige, gelb eingefärbte Rahmen auf. Was ist das? Im Listing 1 wurde eine Instanz der Klasse *GameScene* angelegt. Klickt man auf die gleichnamige Swift-Datei im Project Navigator, so wird eine zugehörige Klasse geöffnet. Bei dem gelben Rahmen handelt es sich um das zugehörige Objekt. Innerhalb des Scene-Objekts werden ja die zum Spiel gehörenden Sprites angezeigt. Dieses Objekt bildet somit den Rahmen für das Spiel.

Ein Spiel kann übrigens natürlich mehr als ein Scene-Objekt enthalten, analog zu den Views in einem Storyboard. Eigenschaften der Klasse *SKScene* sind beispielsweise *Name, Size, Color* oder auch *Camera*. Während die ersten drei genannten Eigenschaften leicht zuzuordnen sind, so ist das bei *Camera* sicherlich nicht unbedingt der Fall.

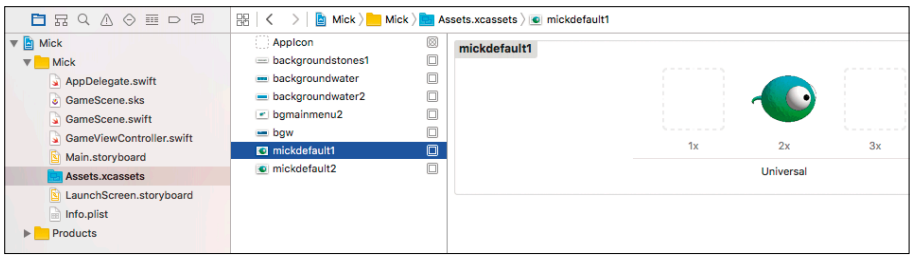


Der Scene Editor nach dem ersten Öffnen (Bild 1)

Mittels eines *SKCameraNode*-Objekts können in einer Szene andere Objekte (zum Beispiel ein *SKSpriteNode)* beleuchtet beziehungsweise ausgeleuchtet werden. Soll lassen sich Lichtquellen und Schatten in Verbindung mit einem Objekt in eine Szene einfügen. Im Beispiel wird die Szene so konfiguriert, dass ein Ausschnitt von 1024 x 768 Pixeln als Spielfläche angezeigt wird.

Sichtbarer Bereich

Die gewählte Größe muss sich aber nicht zwangsweise auf die komplette Szenengröße beziehen, sondern kann auch nur den sichtbaren Bereich beinhalten. Sobald die Szene definiert wurde, kann man damit beginnen, Sprites in diese einzufügen. Das geschieht natürlich ebenfalls visuell. Als Erstes sollte die Object Library von Xcode geöffnet werden. Analog zur Gestaltung ▶



Grafiken für das Spieler–Sprite werden hinzugefügt (Bild 2)

eines User Interface im Storyboard werden in der Object Library nun Elemente angezeigt, die in die Szene eingefügt werden können. Via Drag and Drop wird ein *SKSpriteNode* dort abgelegt.

Sprites in Szene setzen

Solange dem Sprite noch kein Bild zugeordnet worden ist, wird dieses als rotes Quadrat angezeigt. Die Konfiguration des Objekts beginnt mit der Erfassung des Namens. Hier sollte natürlich am besten ein sprechender Name gewählt werden. Im Beispiel wird in dieser Eigenschaft *player* eingetragen. Deutlicher geht es nicht.

Automatisch gesetzt wurde die Eigenschaft *Parent*. Hierbei handelt es sich um die zuvor konfigurierte Szene. Neben diesen beiden Eigenschaften kann auch noch die Größe (Size) oder auch die Position festgelegt werden. Wichtiger aber ist zu diesem Zeitpunkt das Hinzufügen einer Textur. Das Spieler-Sprite soll ja auch nach etwas aussehen.

Aus diesem Grund wird im nächsten Schritt dem Projekt eine Grafik für die Spielfigur hinzugefügt. Hierzu wird im Projekt im *Assets*-Ordner eine entsprechende Grafik, genauer gesagt zwei Grafiken, dem Projekt hinzugefügt (Bild 2). Warum zwei Grafiken? Nun, bei der Spielfigur handelt es sich um eine Kaulquappe. Diese bewegt sich durch ein Gewässer mit Hilfe des Schwanzes fort. Die Links- und Rechtsbewegung werden jeweils durch eine Grafik dargestellt. Deshalb werden zwei Grafiken *(mickdefault1 und mickdefault2)* benötigt.

Neben der Spielfigur müssen außerdem noch weitere Grafiken zum Beispiel für den Hintergrund eingefügt werden. Aus diesem Grund wird dem SKScene-Objekt eine Hintergrundgrafik *(bgw)* in der Eigenschaft *Texture* zugewiesen. Eine weitere Grafik wird für eine zusätzliche *SKSpriteNode*-Instanz benötigt.

Wenn Mick, die Kaulquappe, sich bewegt, soll der Hintergrund sich ebenfalls bewegen. Um diese Illusion zu realisieren, wird eine Grafik mit Pflanzen und Steinen einem weiteren Sprite zugewiesen.

Nachdem alle Bestandteile der Szene im Scene Editor hinzugefügt worden sind, ergibt sich die in Bild 3 ersichtliche Komposition. Bis zu diesem Punkt wurde noch nicht in größerem Umfang codiert.

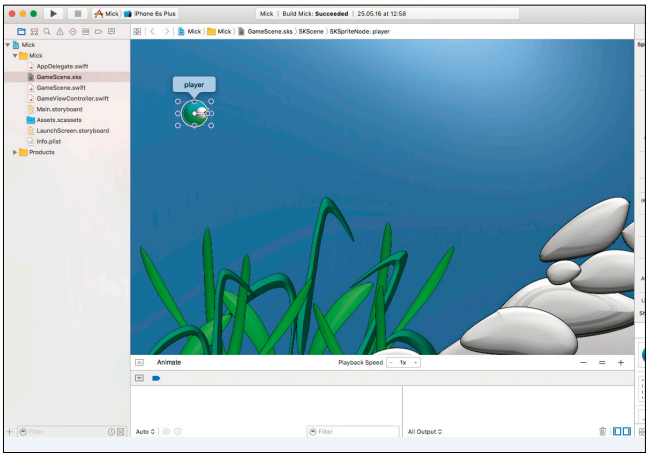
Und Action ...

Das Ganze ist zurzeit natürlich noch statisch. Höchste Zeit also, etwas Bewegung ins Spiel zu bringen. Ohne Code geht es nicht weiter. Zuerst müssen innerhalb der Klasse *GameScene* einige Referenzen zu den SKSpriteNode-Objekten hergestellt werden. Danach kann man dann die ersten Animationen umsetzen.

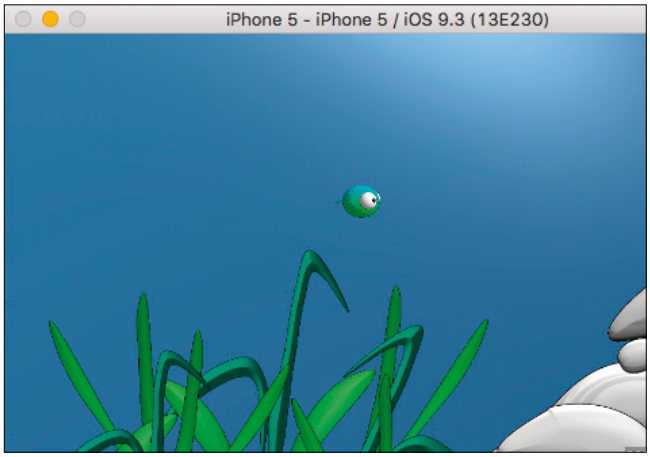
Die Klasse *GameScene* wird abgeleitet von *SKScene,* und außerdem wird das Protokoll *SKPhysicsContactDelegate* implementiert. Das Protokoll wird benötigt, wenn später Kollisionsabfragen im Programm implementiert werden sollen.

Zu Beginn der Klasse werden dann die Referenzvariablen, zum Beispiel für den Spieler und den Hintergrund, definiert. Eine interessante Variable ist *playerSpeed*. Diese Variable wird später benötigt, um die Position des Spielers im Spielfeld zu aktualisieren.

In der Methode *viewDidLoad* wird zuerst die Referenz auf die Implementierung von *SKPhsicsContactDelegate* gesetzt. Anschließend werden die zuvor deklarierten Referenzvariablen für Sprites mit den im Scene Editor erzeugten Definitionen verknüpft. Zuletzt erfolgt in der Methode *viewDidLoad* der Aufruf einer Methode, die Code beinhaltet, mit dem die Spielfigur animiert werden kann.



Konfiguration des Sprites im Scene Editor (Bild 3)



Das Spiel im Simulator (Bild 4)

Listing 3: Steuerung von Mick

```
override func touchesBegan(touches: Set<UITouch>,
withEvent event: UIEvent?) {
  handleTouches(touches)
}

override func touchesMoved(touches: Set<UITouch>,
withEvent event: UIEvent?) {
  handleTouches(touches)
}

override func touchesEnded(touches: Set<UITouch>,
withEvent event: UIEvent?) {
  handleTouches(touches)
}

private func handleTouches(touches: Set<UITouch>) {
  for touch in touches {
    let touchLocation = touch.
      locationInNode(self)
    lastTouch = touchLocation
  }
}

// Muss die Position des Spielers aktualisiert werden?
private func shouldMove(currentPosition
currentPosition: CGPoint, touchPosition: CGPoint) ->
Bool {
  return abs(currentPosition.x -
    touchPosition.x) >
      player!.frame.width / 2 ||
        abs(currentPosition.y -
          touchPosition.y) >
            player!.frame.height/2
}

// Position des Spielers wird aktualisiert
func updatePlayer() {
  if let touch = lastTouch {
    let currentPosition = player!.
      position
    if shouldMove(currentPosition:
      currentPosition, touchPosition:
        touch) {

      let angle = atan2(
        currentPosition.
        y - touch.y, currentPosition.x
          - touch.x) + CGFloat(M_PI)
      let velocotyX = playerSpeed *
      cos(angle)
      let velocityY = playerSpeed *
        sin(angle)

      let newVelocity = CGVector(dx:
        velocotyX, dy: velocityY)
       player!.physicsBody!.
         velocity = newVelocity;
    } else {
      player!.physicsBody!.
        resting = true
    }
  }
}

override func didSimulatePhysics() {
  if let _ = player {
    updatePlayer()
  }
}
```

In der Methode *mickDefaultAnimation* wird eine Instanz der Klasse *SKAction* verwendet, um die Schwanzbewegung von Mick umzusetzen. *SKAction* stellt unterschiedliche Eigenschaften und Methoden bereit. So wird im ersten Schritt eine Variable *wait* angelegt, die eine Pause erzeugt. Anschließend werden zwei *SKAction*-Objekte verwendet, um die beiden Bewegungsbilder von Mick anzuzeigen. Eine letzte Variable vom Typ *SKAction* wird für den Aufruf einer Endlosschleife (Methode: *repeatActionForever)* verwendet.

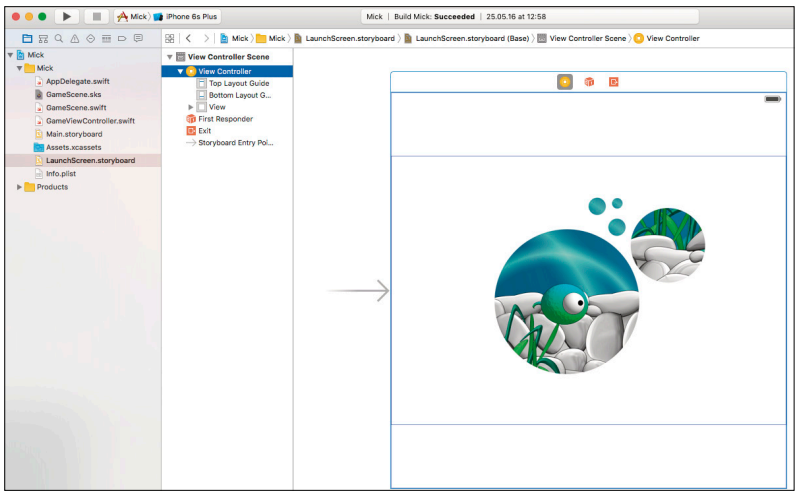
Dieser Methode werden die zuvor erzeugten Variablen als Parameter in einem Array übergeben. Durch einen Aufruf der Methode *runAction* und Übergabe des *SKAction*-Objekts wird die Animation im Code aktiviert.

Mick steuern

Mick kann jetzt also schon einmal seinen Schwanz bewegen. Aber nett wäre es doch, wenn der Spieler ihn auch steuern könnte. Die Steuerung von Mick soll mittels entsprechender Berührungen auf dem Touchscreen umgesetzt werden. Berührt der Spieler eine Stelle auf dem Bildschirm, an der Mick sich nicht befindet, so sorgt die Touchgeste dafür, dass Mick nach Berührung zu der entsprechenden Stelle schwimmt.

Die Umsetzung dieser Anforderung verläuft in mehreren Stufen. Zuerst wird festgestellt, dass der Bildschirm berührt wurde. Anschließend wird die Position der Berührung ermittelt. Als Nächstes wird überprüft, ob die Position von Mick aktualisiert werden muss. Zuletzt, sofern die Prüfung positiv verlaufen ist, wird die Position von Mick innerhalb der Szene angepasst. Das Listing 3 enthält den hierfür erforderlichen Quellcode.

Eingeleitet wird eine Positionsänderung durch die Berührung des Touchscreens. Innerhalb der entsprechenden Methoden *touchesBegan, touchesMoved* und *touchesEnded* wird die Methode *handletouches* aufgerufen, in der die zuletzt durchgeführte Bewegung ausgelesen wird. Anschließend ist es ein Zusammenspiel der Methoden *should-* ▶



Erstellung eines Ladebildschirms (Bild 5)

Move, *updatePlayer* und *didSimulatePhysics*, das dafür Sorge trägt, dass Mick via Touchscreen gesteuert werden kann.

Endlose Bilder

Startet man die App jetzt im Simulator, kann man Mick zwar bewegen, und auch der Schwanz wackelt schon. Mehr tut sich aber noch nicht auf dem Bildschirm (Bild 4). Es fehlt noch etwas Bewegung, um die Illusion zu vervollständigen. Um das zu ändern, wird die Methode *update* implementiert (Listing 4).

Die Methode *update* wird automatisch zeitgesteuert aufgerufen. Somit lässt sich diese Methode prima nutzen, um eine zurückgelegte Strecke zu simulieren. Das erreicht man, indem die Position der beiden Hintergrundgrafiken verändert wird. Die Position der Grafiken wird nur in einer Richtung, von rechts nach links, gesrcollt. Dadurch entsteht der Eindruck, dass Mick durch den Fluss schwimmt.

Bei jedem Aufruf der Methode werden im ersten Schritt von der aktuellen Position der beiden Grafiken Pixel subtrahiert. Durch diese Änderung werden die Grafiken von rechts nach links bewegt. Das funktioniert natürlich nur einmal. Wurden beide Grafiken komplett durch den Ausschnitt bewegt, dann sind sie weg. Damit die dann trostlose Szene nicht so bleibt, wird die aktuelle Position der jeweiligen Grafik mit der folgenden Fallunterscheidung geprüft. Hat die Grafik eine bestimmte Position erreicht, so wird sie einfach vorne, also am rechten Rand, wieder eingefügt. Die Illusion ist nun perfekt: Mick kann jetzt auch schwimmen.

Listing 4: Animaton der Hintergrundgrafiken

```
override func update(currentTime: CFTimeInterval) {

  background!.position =
    CGPointMake(background!.
      position.x-10, background!.
        position.y)
  backgroundwater!.position =
    CGPointMake(backgroundwater!.
      position.x-1, backgroundwater!.
        position.y)

  if background!.position.x < -
    (background!.size.width-
      (background!.size.width/2)) {
    background!.position =CGPointMake(
      background!.size.width,
        background!.position.y)
  }

  if backgroundwater!.position.x < 0 {
    print(backgroundwater!.position.x)
      backgroundwater!.position =
        CGPointMake(backgroundwater!.
          size.width/2,
            backgroundwater!.
              position.y)
  }
}
```

Ladebildschirm

Im Simulator kann je nach Mac das Laden der App einen Moment dauern. Auch auf langsamen iOS-Geräten kann es einen Moment dauern, bis die App startet. Es wäre doch nett, wenn diese Wartezeit durch einen Ladebildschirm etwas überbrückt würde.

Für diese Ergänzung muss nicht einmal programmiert werden. Im LaunchScreen.storyboard fügt man einfach ein ImageView-Control sowie ein hübsches Bild ein (Bild 5). Man sollte darauf achten, dass die erforderlichen Constraints zur Zentrierung gesetzt werden, damit das Bild auch immer, unabhängig vom Gerät, in der Mitte des Bildschirms angezeigt wird. Wurde auch dieser Punkt umgesetzt, so hat man schon fast ein kleines Spiel.

Fazit

Durch die Einführung der SpriteKit-Bibliothek hat Apple Entwicklern einiges an Arbeit abgenommen. Der SpriteKit Scene Editor in Xcode erleichtert durch die visuellen Gestaltungsmöglichkeiten die Entwicklung von Spielen erheblich. ■

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

DWX Developer Week 2017

26.-29. Juni 2017,
Messe Nürnberg

developer-week.de #dwx17 DeveloperWeek

MOBILE APPS CROSS-PLATFORM ENTWICKELN MIT QT 5.7

Hervorragende Basis

Die neuen Qt Quick Controls sind eine hervorragende Basis für mobile Anwendungen.

In den vorangegangenen beiden Ausgaben der **web & mobile developer** habe ich eine Einführung in die neuen leichtgewichtigen Qt Quick Controls 2 gegeben sowie das Thema der App-Navigation ausführlich besprochen.

In dieser letzten Folge der kleinen Artikelserie werfen wir einen Blick auf weitere Aspekte bei der Entwicklung anspruchsvoller Mobile-Apps: Data Binding, Caching, Listen, Lifecycle und mehr.

Datum und Uhrzeit auswählen

Bevor wir uns den Daten zuwenden, ein weiteres Beispiel der Customization von Qt Quick Controls 2. Nicht nur in Business-Apps müssen Datum oder Uhrzeit ausgewählt werden. Out of the box liefert Qt dazu nur Tumbler an – also einfache Drehräder. Das funktioniert, ist aber nicht das, was man von einer modernen App erwartet. Also habe ich einen Date Picker und einen Time Picker auf der Grundlage anderer Qt Quick Controls 2 selbst gebaut. Alles ist natürlich wie immer bei GitHub als Open Source verfügbar.

Betrachten wir zunächst den Date Picker. Qt bietet dazu als TechPreview im Namespace *qt.labs.Calendar 1.0* an:

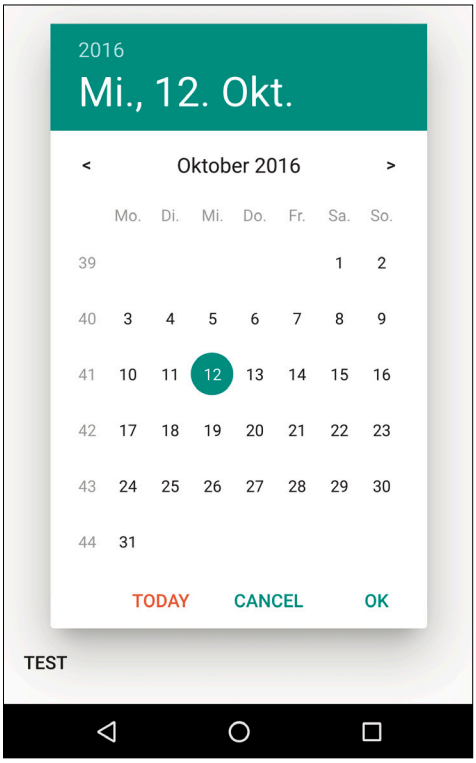
- CalendarModel,
- DayOfWeekRow,
- MonthGrid,
- WeekNumberColumn.

Den eigentlichen Kalender muss man sich aber selbst bauen. Dazu habe ich ein GridLayout mit Customized Label und Buttons gewählt. Aufgerufen wird der Date Picker über ein Pop-up. Am besten sehen Sie sich dazu den Sourcecode an – da ist keine große Hexerei dabei. Bild 1 zeigt den Date Picker im Einsatz.

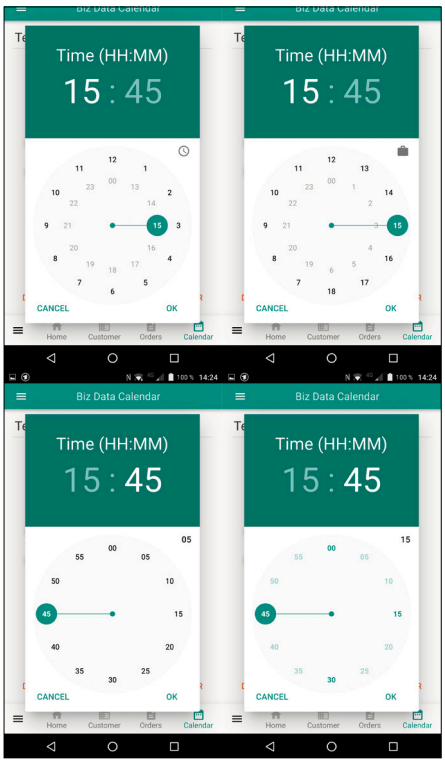
Der Time Picker war etwas aufwendiger. Es muss ja zwischen Stunden- und Minuteneingabe gewechselt werden, und bei der Stundeneingabe gibt es zwei Ringe von je zwölf Stunden. Um diese Stundenringe zu erzeugen, werden Buttons im Kreis angeordnet und so gedreht, dass der Text (die Stundenzahl) lesbar ist. Dazu wird ein Repeater genutzt. Listing 1 zeigt den Code für den äußeren Stundenring. Es mussten also nur Buttons und Label angepasst werden. Der Vorteil eines selbst gebauten Controls liegt ja auch darin, weitere Features zu implementieren. So stört es mich, dass der äußere Stundenring von 01 bis 12 und der innere von 13 bis 00 Uhr geht, man aber in einer Business-App die meisten Termine tagsüber vereinbart. Daher habe ich ein Flag eingebaut, das die Businesszeiten (07–18 Uhr) außen zeigt und die Nachtzeiten innen (19–06 Uhr). Bei der Minutenanzeige kann noch konfiguriert werden, welches Intervall erlaubt ist. So könnte beispielsweise sichergestellt werden, dass nur Zeiträume zu je 15 Minuten ausgewählt werden dürfen. Neben den beiden Stundenringen brauchen wir noch den Mittelpunkt und die Linie vom Mittelpunkt zum ausgewählten Button (Listing 2).

Damit bekannt ist, welcher Button von den zwölf geklickt wurde, sind die Buttons *checkable* und zu einer *ButtonGroup* zusammengefasst:

```
ButtonGroup {
  id: outerButtonGroup
}
```



Date Picker: Einfache Auswahl des Datums (Bild 1)



Time Picker: Elegante Auswahl der Uhrzeit (Bild 2)

Listing 1: Äußerer Stundenring

```
Repeater {
 id: outerRepeater
 model: timePicker.timePickerModel
 delegate: Button {
  id: outerButton
  text: timePicker.pickMinutes? modelData.m :
  timePicker.useWorkTimes? modelData.d : modelData.c1
   font.bold: checked || timePicker.pickMinutes &&
   timePicker.onlyQuartersAllowed
   x: timePicker.timeButtonsPaneSize / 2 - width / 2
   y: timePicker.timeButtonsPaneSize / 2 - height / 2
   width: 40
   height: 40
   checked: index == timePicker.outerButtonIndex
   checkable: true
   enabled: timePicker.pickMinutes &&
   timePicker.onlyQuartersAllowed? modelData.q : true
    onClicked: {
     timePicker.innerButtonIndex = -1
     timePicker.outerButtonIndex = index
     if(timePicker.pickMinutes) {
      timePicker.minutesDisplay =
      timePicker.timePickerDisplayModel[index].m
     } else {
     if(timePicker.useWorkTimes) {
      timePicker.hrsDisplay =
      timePicker.timePickerDisplayModel[index].d
     } else {
      timePicker.hrsDisplay =
      timePicker.timePickerDisplayModel[index].c1
     }
     if(timePicker.autoSwapToMinutes) {
      timePicker.onMinutesButtonClicked()
     }
    }
   }
   ButtonGroup.group: outerButtonGroup
  property real angle: 360 *
  (index / outerRepeater.count)
  transform: [
   Translate {
    y: -timePicker.timeButtonsPaneSize * 0.5 +
    outerButton.height / 2
   },
   Rotation {
    angle: outerButton.angle
    origin.x: outerButton.width / 2
    origin.y: outerButton.height / 2
   }
  ]
  contentItem: Label {
   text: outerButton.text
   font: outerButton.font
   minimumPointSize: 8
   fontSizeMode: Text.Fit
   opacity: enabled || outerButton.highlighted ||
   outerButton.checked ? 1 : 0.3
   color: outerButton.checked ||
   outerButton.highlighted ? textOnPrimary :
   timePicker.pickMinutes &&
   timePicker.onlyQuartersAllowed?primaryColor :
   "black"
   horizontalAlignment: Text.AlignHCenter
   verticalAlignment: Text.AlignVCenter
   elide: Text.ElideRight
   rotation: -outerButton.angle
  } // outer content label
  background: Rectangle {
   color: outerButton.checked ? primaryColor :
   "transparent"
   radius: width / 2
  }
 } // outer button
} // outerRepeater
```

```
ButtonGroup {
  id: innerButtonGroup
}
```

Bild 2 zeigt die Varianten des Time Pickers. Der Aufruf des Time Pickers ist einfach:

```
ButtonFlat {
  text: "Time Picker"
  onClicked: {
    timePicker.open()
    timePicker.setDisplay(root.myTime)
  }
}
TimePicker {
  id: timePicker
  modal: true
  onClosed: {
    if(timePicker.isOK) {
      root.myTime = timePicker.hrsDisplay + ":" +
      timePicker.minutesDisplay
    }
  }
};
```

Nach diesem Ausflug in das UI kommen wir zum Thema Daten. Keine Businessanwendung kommt ohne Daten aus. Diese Daten werden oft von REST-Services abgerufen und angezeigt, oder aber am Gerät erfasst und an einen WebService übermittelt. ▶

Listing 2: Mittelpunkt und Linie Mittelpunkt – Button

```
Rectangle {
  // line to outer buttons
  visible: timePicker.outerButtonIndex >= 0
  x: timePicker.timeButtonsPaneSize / 2
  y: timePicker.timeButtonsPaneSize / 2 - height
  width: 1
  height: timePicker.timeButtonsPaneSize / 2 - 40
  transformOrigin: Item.Bottom
  rotation: outerButtonGroup.checkedButton?
  outerButtonGroup.checkedButton.angle : 0
  color: primaryColor
  antialiasing: true
} // line to outer buttons

Rectangle {
  // line to inner buttons
  visible: timePicker.innerButtonIndex >= 0 &&
  !timePicker.pickMinutes
  x: timePicker.timeButtonsPaneSize / 2
  y: timePicker.timeButtonsPaneSize / 2 - height
  width: 1
  height: timePicker.innerButtonsPaneSize / 2 - 40
  transformOrigin: Item.Bottom
  rotation: innerButtonGroup.checkedButton?
  innerButtonGroup.checkedButton.angle : 0

  color: primaryColor
  antialiasing: true
} // line to outer buttons

Rectangle {
  // centerpoint
  anchors.centerIn: parent
  width: 10
  height: 10
  color: primaryColor
  radius: width / 2
}
```

Qt bietet viele Möglichkeiten an, Daten an UI Controls zu binden und dabei verschiedenste Patterns wie MVC umzusetzen. Wer möchte, kann auch alles direkt in QML und JavaScript implementieren und über Signals und Slots verknüpfen. Der bessere und performantere Weg ist aber, die Daten in C++ zu verwalten.

Das kann bei Qt beispielsweise über XML- oder SQL-Datenmodelle erfolgen, oder aber über QObject*. Wer meinen Beiträgen in den letzten Jahren gefolgt ist, der weiß, dass ich für BlackBerry Cascades mit QObject*-Datenmodellen gearbeitet habe und die Daten vorrangig als JSON-Dateien gespeichert habe und nur bei großen Datenmodellen in der SQLite. Dies wollte ich jetzt auch unter Qt umsetzen.

Um Entities als QObject* zu beschreiben und die Properties an UI-Elemente zu binden, ist erst einmal etwas Fleißarbeit notwendig, um all das einzutippen, und die Wartung ist aufwendig, wenn sich Properties ändern. Aber das ist ja exakt der Grund, warum ich mir für BlackBerry 10 Cascades dazu einen auf Eclipse Xtext basierenden Codegenerator erstellt habe, der jetzt neu als mobaDSL für die Cross-Plattform-Codegenerierung gebaut wird.

Hier ein kurzer Einblick. Die *Entity*-Klassen müssen von *QObject* ableiten und das Makro *Q_OBJECT* definieren. Für alle Properties muss ein *Q_PROPERTY* definiert warden (Listing 3).

Zu den Properties müssen dann die Variablen und die Getter und Setter gebaut werden und bei Änderungen ein entsprechendes Signal senden, damit das UI informiert wird:

```
Q_SIGNALS:
void nrChanged(int nr);
void orderDateChanged(QDate orderDate);
void remarksChanged(QString remarks);
void expressDeliveryChanged(bool expressDelivery);
void customerChanged(int customer);
void positionsPropertyListChanged();
```

Listen sind noch komplizierter, da hierzu statische Methoden implementiert werden müssen, die es dann erlauben, aus dem UI per JavaScript wie auf ein JS Array zuzugreifen:

```
private:
  int mNr;
  QDate mOrderDate;
  QString mRemarks;
  bool mExpressDelivery;
  int mCustomer;
  bool mCustomerInvalid;
  QList<Position*> mPositions;
  static void appendToPositionsProperty(QQmlListProperty
  <Position> *positionsList, Position* position);
  static int positionsPropertyCount(QQmlListProperty
  <Position> *positionsList);
  static Position* atPositionsProperty(QQmlListProperty
  <Position> *positionsList, int pos);
  static void clearPositionsProperty(QQmlListProperty
  <Position> *positionsList);
```

Allerdings kann man da derzeit nur Daten ändern oder anzeigen, aber keine neuen hinzufügen oder Datensätze löschen – das geht nur aus C++ heraus. Das braucht man natürlich auch im UI – also muss das selbst implementiert werden und wird per *Q_INVOKABLE*-Makro aus dem UI aufrufbar gemacht.

```
  Q_INVOKABLE
  void addToPositions(Position* position);
```

```
  Q_INVOKABLE
  bool removeFromPositions(Position* position);
```

Das waren jetzt nur die Angaben im Header. Ich erspare mir hier die Ausgabe des Sourcecodes der Implementierung in der *.cpp*. Das kann jeder im GitHub-Projekt nachvollziehen.

Getter, Setter, Remove, Add sind aber nur die Grundfunktionalitäten. Wir bekommen ja die Daten beispielsweise vom Webservice als JSON geliefert.

Dieses JSON kann dann in eine *QVariantMap* für ein JSON-Objekt oder *QVariantList* für ein JSON-Array transformiert werden. Und dann müssen wir diese Daten ins *Entity* bringen. Umgekehrt ist es beim Speichern der Daten – dann muss alles in ein korrektes JSON-Objekt oder Array gewandelt werden. Bei all diesen Operationen können Tippfehler passieren. Ich lasse das alles generieren. Das Einzige, was ich benötige, ist ein Datenmodell:

```
dto Order {
  domainKey int nr;
  @DateFormatString("yyyy-MM-dd")
  var Date orderDate;
  var QString remarks;
  var bool expressDelivery;

  ref cascade Position [1..*] positions opposite
  orderHeader;
  ref lazy Customer [1] customer;
}
```

Die DSL (Domain Specific Language) generiert dann daraus den entsprechenden Code. Damit die *QObject**-Entities im UI bekannt sind, müssen sie zunächst entsprechend deklariert werden:

```
qmlRegisterType<Customer>("org.ekkescorner.data", 1, 0,
"Customer");
qmlRegisterType<Order>("org.ekkescorner.data", 1, 0,
"Order");
qmlRegisterType<Position>("org.ekkescorner.data", 1, 0,
"Position");
qmlRegisterType<SettingsData>("org.ekkescorner.data", 1,
0, "SettingsData");
```

Jetzt muss man nur *org.ekkescorner.data 1.0* in QML importieren und kann auf diese C++-Klassen wie auf normale Datentypen zugreifen. Ein Labeltext *myCustomer.name* zeigt den Namen an und reagiert automatisch auf Änderungen in C++. Verändert man den Wert im UI, sind diese by-magic auch sofort im C++-QObject* geändert. Alles durch direkte Referenzierung des Pointers – keine extra by-value-copies. Aufpassen muss man mit dem *parent* der QObject*. Diese müssen immer in C++ sein, damit die Objekte nicht gelöscht werden, wenn ein UI Control zerstört wird. Ich erzeuge dazu die Objekte immer in einer speziellen *DataManager*-Klasse über eine *Q_INVOKABLE*-Methode auch aus dem UI heraus.

Listing 3: Q_PROPERTY

```
class Order: public QObject {
  Q_OBJECT
  Q_PROPERTY(int nr READ nr WRITE setNr NOTIFY
  nrChanged FINAL)
  Q_PROPERTY(QDate orderDate READ orderDate WRITE
  setOrderDate NOTIFY orderDateChanged FINAL)
  Q_PROPERTY(QString remarks READ remarks WRITE
  setRemarks NOTIFY remarksChanged FINAL)
  Q_PROPERTY(bool expressDelivery READ
  expressDelivery WRITE setExpressDelivery NOTIFY
  expressDeliveryChanged FINAL)
  Q_PROPERTY(int customer READ customer WRITE
  setCustomer NOTIFY customerChanged FINAL)
  Q_PROPERTY(Customer* customerAsDataObject READ
  customerAsDataObject WRITE
  resolveCustomerAsDataObject NOTIFY
  customerAsDataObjectChanged FINAL)
  Q_PROPERTY(QQmlListProperty<Position>
  positionsPropertyList READ positionsPropertyList
  NOTIFY positionsPropertyListChanged)
```

Caching und Lifecycle

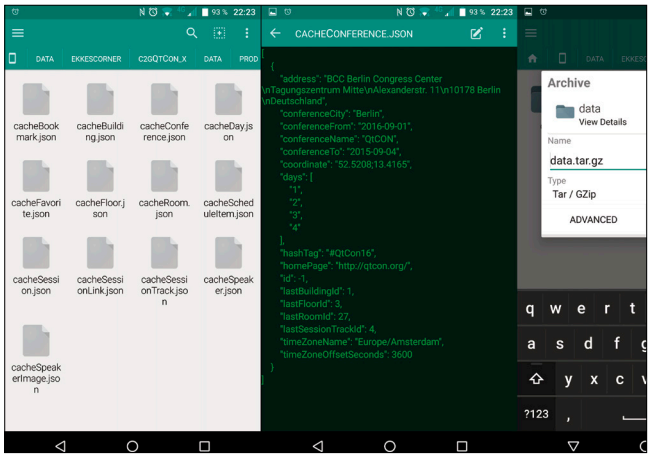
Wie bereits erwähnt, möchte ich Daten als JSON-Modelle speichern. In BlackBerry 10 Cascades gibt es dazu die Klasse *JsonDataAccess*. Qt hat etwas Ähnliches: *JsonDocument*. Der notwendige Code, um eine JSON-Datei zu lesen oder zu speichern, ist vergleichbar. Hier die Qt-Variante zum Einlesen eines Arrays:

```
QJsonDocument jda;
QVariantList cacheList;
QString cacheFilePath = dataPath(fileName);
QFile dataFile(cacheFilePath);
jda = QJsonDocument::fromJson(dataFile.readAll());
cacheList = jda.toVariant().toList();
return cacheList;
```

Und hier der Code zum Speichern:

```
void DataManager::writeToCache(const QString& fileName,
QVariantList& data)
{
  QString cacheFilePath = dataPath(fileName);
  QJsonDocument jda = QJsonDocument::fromVariant(data);
  QFile saveFile(cacheFilePath);
  saveFile.write
  (jda.toJson(QJsonDocument::Compact:QJsonDocument));
}
```

Hier entdeckt man ein Feature, das ich in Cascades immer vermisst habe: das Speichern im kompakten Format. Während der Testphase ist es ja ganz gut, einen lesbaren Code zu haben; im Produktivbetrieb aber werden die Datenmen- ▶



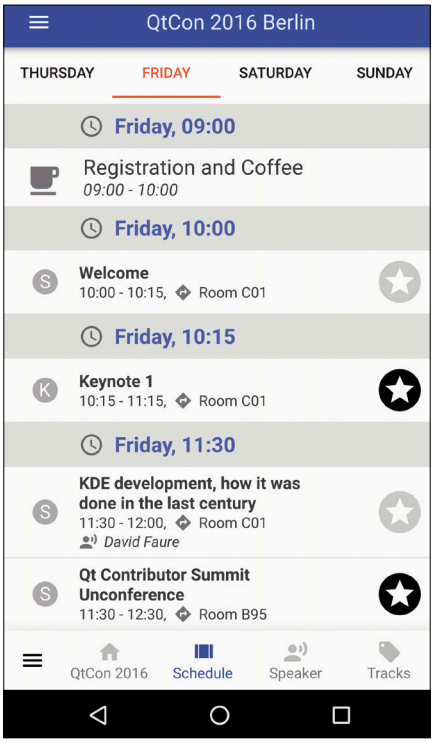
GenericDataLocation: Zugriff unter Android über den FileExplorer (Bild 3)

gen durch kompakten Code spürbar verringert. Nun können wir also auch unter Qt mit JSON-Datenmodellen für unseren Cache arbeiten.

Bisher war der Code zwischen BlackBerry 10 Cascades und Qt 5.7 zu 90 Prozent identisch – was den C++-Teil angeht. In der Architektur gibt es aber einen entscheidenden Unterschied, wenn es um den Lifecycle geht. Sobald eine App in den Hintergrund geht, kann es passieren, dass diese vom Android OS gekillt wird – und zwar, auch ohne ein entsprechendes Signal zu senden. Unter BlackBerry 10 konnte das nicht passieren und ich habe den Cache immer zum Programmende gespeichert. Jetzt speichere ich immer sofort, wenn das Signal kommt, dass die App suspendiert wird:

```
void ApplicationUI::
onApplicationStateChanged(Qt::
ApplicationState applicationState) {
  if(applicationState == Qt::
  ApplicationState::
  ApplicationSuspended) {
    startCaching();
    return;
  }
  if(applicationState == Qt::
  ApplicationState::
  ApplicationActive) {
    resetCaching();
  }
}
```

Es ist jetzt wichtig zu prüfen, wann welche Daten wirklich in den Cache geschrieben werden müssen. Ein weiterer zu beachtender Gesichtspunkt ist das Verhalten im Hintergrund: Die Apps arbeiten dann normalerweise nicht mehr, und unter Android sollte Background-Aktivität mit einem Service abgewickelt werden. Unter Qt gibt es seit 5.7 auch Services für Android. Außerdem kann über die Android Externals auch auf Android-Code beziehungsweise analog bei iOS auf Objective-C zugegriffen werden.



ListView, Gruppierung und Item-Typen im Einsatz (Bild 4)

Sandbox und File Locations

Wir können also jetzt JSON-Dateien lesen und schreiben. Aber wohin? Welche Verzeichnisse sind *read-write* bei Android und iOS? Qt bietet den Zugriff über Standardverzeichnisse an. Ich habe die *AppDataLocation* gewählt, die unter Android und iOS beschreibbar ist und innerhalb der App-Sandbox liegt. Bei iOS kann man Daten sowieso nur in der App-Sandbox schreiben – einzige Ausnahme: Mediendateien. Die *AppDataLocation* gibt es bei beiden Betriebssystemen:

```
mDataRoot = QStandardPaths::standardLocations
(QStandardPaths::AppDataLocation).value(0);
mDataPath = mDataRoot+"/data/";
mDataAssetsPath = ":/data-assets/";
qDebug() << "Data Path: " << mDataPath << "
data-assets: " << mDataAssetsPath;
bool ok = checkDirs();
```

Wichtig: Unter iOS gibt es das Directory nicht – es muss erst angelegt werden.

Wer den obigen Code betrachtet, wundert sich vielleicht über den *:/data-assets*-Pfad. Das ist ein Verzeichnis in den Qt-Ressource-Dateien, in dem ich unter anderem Dateien mit ausliefere. Auf diese Verzeichnisse kann nur lesend zugegriffen werden, und wenn man Dateien aus den Ressourcen in die *AppDataLocation* kopiert, dann sind diese dort ebenfalls *read-only*. Das irritiert, denn neue Dateien, die in der *AppDataLocation* angelegt werden, sind sofort beschreibbar. Um die Read-only-Dateien beschreibbar zu machen, muss die Permission dafür gesetzt werden:

```
dataFile.setPermissions(QFileDevice:
:ReadUser | QFileDevice::WriteUser);
```

Unter BlackBerry 10 Cascades gab es zum Testen ein cooles Feature in Eclipse Momentics: Über den *TargetFileSystemNavigator* konnte direkt in die App-Sandbox geschaut werden, solange der *DEBUG TOKEN* installiert war und die App aus Momentics deployt wurde. Somit war es einfach, mal schnell in eine JSON-Datei hineinzuschauen. Das geht unter Android oder iOS so nicht – von außen kann ich nicht einfach in die *ApplicationData* blicken. Unter Android habe ich einen Weg gefunden: Sofern die App nicht im Release-Mode gestartet ist, ermögliche ich es dem Entwickler, die Daten in einem anderen Verzeichnis zu speichern, das von außen einsehbar ist:

```
#ifdef QT_DEBUG
  qDebug() << "Running a DEBUG BUILD";
  if (mSettingsData->hasPublicCache()) {
    mDataRoot = QStandardPaths::standardLocations
    (QStandardPaths::GenericDataLocation).value(0);
    mDataRoot += mSettingsData->publicRoot4Dev();
    mDataPath = mDataRoot+"/data/";
  }
```

Achtung: Unter iOS befinden sich die Dateien der *GenericDataLocation* weiterhin unter der Obhut der App – unter Android aber kann ich mit einem FileExplorer hineinschauen und dann direkt in die JSON-Datei oder aber mein Cache-Verzeichnis komprimieren und schließlich das Ergebnis per Mail an mich senden (Bild 3).

Noch ein Hinweis zu den Zugriffsrechten: Wenn ich Read-only-Dateien aus den Ressourcen in die *GenericDataLocation* kopiere, dann sind diese – anders als in der *AppDataLocation* – auch schreibberechtigt, und das Setzen der Permissions bringt einen Fehler.

ListViews und Repeater

Um Daten in Listen darzustellen, wird normalerweise eine ListView verwendet, der ein Datenmodell zugeordnet wird – in diesem Fall also eine *List<MyObject*>*. Unter Cascades habe ich in 90 Prozent der Fälle ein GroupDataModel eingesetzt, da ich dann automatisch sortierte und gruppierte Listen bekommen habe. Leider gibt es diesen Komfort bei den angebotenen Datenmodellen nicht. Das Sortieren und Gruppieren muss selbst gebaut werden.

Auch unter Qt kann ich Listen gruppieren – dazu müssen die Daten aber in der richtigen Sortierung geliefert werden. Derzeit nutze ich die oben beschriebenen QQmlPropertyLists zur Anzeige in Listen. Habe ich verschiedene Inhaltstypen, dann ist das ebenfalls möglich:

```
ListView {
  id: listView
  model: dataManager.sessionPropertyList
  delegate:
    Loader {
    id: sessionLoader
    sourceComponent: hasScheduleItem()?
    scheduleRowComponent : sessionRowComponent
```

In diesem Fall arbeite ich mit einem Datenmodell aus *QObject**, das de facto eine *QList<Session*>* ist. In diesen Sessions gibt es normale Sessions und ScheduleItems. Im *delegate* sieht man, dass ein Loader verwendet wird, der abhängig von der Methode *hasSchedule()* in der Session-Klasse dann die entsprechende Component zur Aufbereitung der Daten nimmt. Für die Gruppierung ist eine *section*-Komponente zuständig:

```
section.property: "sortKey"
section.criteria: ViewSection.FullString
section.delegate: sectionHeading
```

Links zum Thema

- Qt Quick Controls
  http://doc-snapshots.qt.io/qt5-5.7/qtquickcontrols2-index.html
- Qt Quick Controls 2 Tumbler
  http://doc.qt.io/qt-5/qml-qtquick-controls2-tumbler.html
- QtCon Conference App bei GitHub
  http://github.com/ekke/c2gQtCon_x
- Beispiel-App bei GitHub
  http://github.com/ekke/biz_data_x
- Qt Labs Calendar
  http://doc.qt.io/qt-5/qtlabscalendar-index.html
- ListView
  http://doc.qt.io/qt-5/qml-qtquick-listview.html
- Repeater
  http://doc.qt.io/qt-5/qml-qtquick-repeater.html
- Models und Views
  http://doc.qt.io/qt-5/qtquick-modelviewsdata-modelview.html

Die Details der Implementierung der Komponenten können im Sourcecode nachgelesen werden. Alles im Einsatz zeigt Bild 4. Anstelle einer ListView kann auch ein Repeater gewählt werden, um mehrere UI Controls anhand der Daten eines Datenmodells aufzubereiten:

```
Repeater {
  model: speaker.sessionsPropertyList
  // here are the UI Controls
}
```

Ich nutze Repeater meist, wenn es sich nur um wenige Elemente handelt. Im Beispiel sind es die Sessions, die ein Speaker präsentiert – das sind meist nur ein bis drei Datensätze.

Fazit

Qt 5.7 Quick Controls 2 sehen gut aus und bieten mit den anderen Features von Qt eine gute Basis für mobile Apps. In den nächsten Monaten werde ich sporadisch weitere Artikel zu diesem Thema veröffentlichen und meine Beispiel-Apps, Patterns und Customized Controls bei GitHub bereitstellen. ■

Ekkehard Gentz
ist Autor, Trainer und Speaker auf Konferenzen. Er entwickelt als Independent Software Architect mobile Anwendungen für internationale Kunden, ist BlackBerry Elite Member und bloggt unter:
http://ekkes-corner.org

SMART DATA
Developer Conference
Big Data & Smart Analytics

Für Softwareentwickler und IT-Professionals

06. Dezember 2016, Köln

web & mobile DEVELOPER Leser erhalten 15 % Rabatt mit Code SMART16wmd

Themenauswahl:

- Effizienter durch standardisierte Reports
- Datenqualität erhöhen
- Recommender Algorithmen: R vs. Spark
- Streaming = Zukunft von Big Data

Ihre Experten (u.a.):

Damir Dobric
DAENET Corporation

„German Cloud für Entwickler und IT-Pros“

Constantin Klein
Freudenberg IT SE & Co. KG

„Eine Branche im Daten-Goldrausch“

Dr. Hanna Köpcke
Webdata Solutions GmbH

„Datenqualität: Big Data Matching“

Stefan Papp
The unbelievable Machine Company GmbH

„Streaming mit Apache Flink“

Tobias Trelle
codecentric AG

„Einführung in Graphen-Datenbanken mit Neo4j“

smart-data-developer.de #smartddc SMARTDATADeveloperConference

Präsentiert von: dotnetpro web & mobile DEVELOPER

SMART DATA
Developer Conference

Big Data & Smart Analytics **06. Dezember 2016, Köln**

08.45	**Begrüßung**
09.00 – 09.55	**Keynote: Eine Branche im Goldrausch** *Constantin Klein* Die Datenexplosion ist nicht mehr aufzuhalten. Zwischen Hype und Realität entdecken Sie einen praxisorientierten Ansatz, um individuell Ihren Claim in diesem Goldrausch abzustecken.
10.00 – 10.55	**Datenqualität: Big Data Matching** *Dr. Hanna Köpcke* Im Rahmen dieses Vortrages werden Herausforderungen beim Matching von Big Data beleuchtet und Lösungsansätze auf Basis von Map/Reduce aufgezeigt.
11.30 – 12.25	**4x4: Big Data in der Cloud?** *Danny Linden* Daten in die Cloud auslagern? Warum und wenn ja, bei welchem Provider? Anhand von vier Beispielen können Sie eine geeignete Lösung finden.
12.30 – 13.25	**NoSQL: Einführung in Graphdatenbanken mit Neo4j** *Tobias Trelle* Warum überhaupt nicht-relational? Lernen Sie Neo4j, die populärste Graphen-Datenbank, inkl. Datenstrukturen, Query-Language Cypher und API kennen - eine spezielle NoSQL-DB.
14.00 – 14.30	**Lunch-Session: „German Cloud“ für Devs und IT-Pros** *Damir Dobric und Andreas Erben* Was geht, und was nicht? Was ist anders?
14.30 – 15.25	**Implementierung von Recommender-Algorithmen** *Dr. Henrik Behrens* Anhand eines konkreten Anwendungsfalls, der Programmierung eines Recommender-Systems, vergleichen wir eine konventionelle Implementierung in R mit zwei Implementierungen in der Big-Data-Technologie Spark (Spark DataFrames und Spark MLlib).
16.00 – 16.55	**Smart Analytics: Streaming mit Apache Flink** *Stefan Papp* Lernen Sie Apache Flink kennen und wie man eine Streaming-Applikation damit schreibt. Streamingplattformen sind die Zukunft von Big Data.
17.00 – 18.00	**Visualisierung, Active & Mobile Reports – ein Anwendungsbeispiel** *Holger Gerhards* Welche Vorteile bieten Standardisierungen im Reporting? Sehen Sie, wie Reports in wenigen Minuten mit wenigen Klicks erstellt werden. Entdecken Sie die Vorteile und Möglichkeiten von HICHERT®SUCCESS im IBM Analytics Umfeld.

Programmänderung vorbehalten

Anmeldung: smart-data-developer.de

Veranstalter: developer media — Neue Mediengesellschaft Ulm mbH

DATA BINDING FÜR ANDROID

Automatische Verbindung

Data Binding wurde von Microsoft zusammen mit XAML eingeführt.

Die Technik erlaubt Entwicklern, die in Steuerelementen vorgehaltenen Informationen automatisch mit in diversen Klassen vorenthaltenen Informationen zu verbinden.

Google stellte im Jahr 2015 eine Datenbindungsbibliothek vor, die das bisher notwendige Erzeugen von Glue Code massiv reduzieren sollte. Mittlerweile ist das Produkt Teil von Android Studio und erfuhr sogar mehrere Aktualisierungen. Wer eine aktuelle Version von Googles IDE verwendet, kann sofort loslegen.

Abwärtskompatibilität

Die folgenden Schritte basieren auf Version 2.1.2 von Android Studio. Als Basis-SDK diente die mittlerweile durchaus weit verbreitete Version 5.0. Als Mindestversionen gibt Google Android 2.1 und Gradle 1.5.0-alpha1 an. Für Abwärtskompatibilität dürfte also gesorgt sein. Achten Sie allerdings darauf, dass der Editor erst ab Version 2.1 die fortgeschrittenen Datenbindungs-Strings interpretieren kann.

Zum Einbinden von Data Binding ist heute kein großflächiger Eingriff in die Projektstruktur mehr notwendig. Öffnen Sie die zum App-Modul gehörende *.gradle*-Datei und ergänzen Sie sie um die *dataBinding*-Passage:

```
apply plugin: 'com.android.application'
android {
  ...
  }
  buildTypes {
    ...
    }
  }
  dataBinding {
    enabled = true
  }
}
```

Damit ist die Einbindung ins Projekt auch schon abgeschlossen. Die eigentlichen Änderungen erfolgen in den Layoutdateien, die nun nach folgendem Schema aufzubauen sind:

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/
res/android">
  <data>
    <variable name="<NAME>" type="<KLASSE>"/>
  </data>
  <VIEW>
</layout>
```

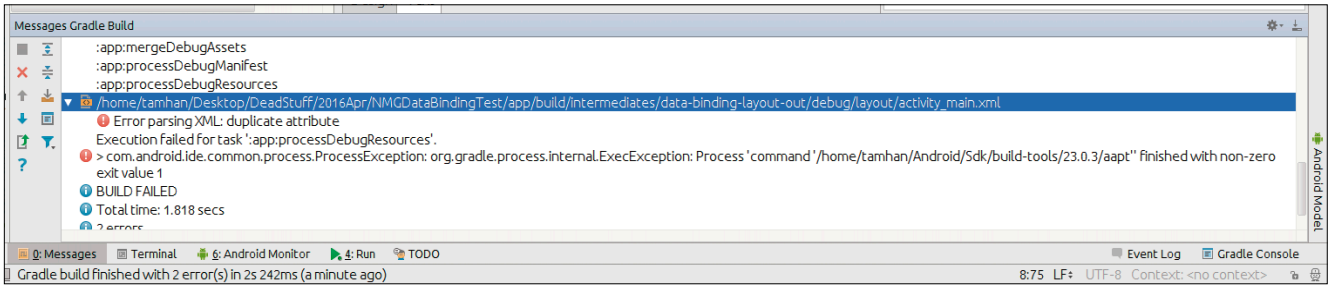
Anstatt die anzuzeigenden Steuerelemente wie bisher direkt zu implementieren, ist nun ein Layout-Element erforderlich. Dieses hat zwei Kinder: erstens ein *data*-Element mit Informationen für die Binding-Engine; zweitens ein beliebiges, für die Darstellung zuständiges View-Steuerelement. In den Eigenschaften dieses Widgets finden sich dann Bindeausdrücke, die zur Laufzeit mit den im Modell befindlichen Werten belebt werden.

Generator herbei

Wer einmal mit JavaScript-Data-Binding-Frameworks wie Knockout gearbeitet hat, weiß über die Schwierigkeiten mit der Erzeugung von Bindungsklassen Bescheid. Google hilft seinen Entwicklern hier mit einem Generatorwerkzeug weiter, das die Erstellung der Brückenklassen automatisiert.

Listing 1: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/
res/android">

  <data>
    <variable
      name="nmg" type="com.tamoggemon.
      nmgdatabindingtest.NMGSample1">
    </variable>
  </data>
  <RelativeLayout
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    ...
    tools:context=
    "com.tamoggemon.nmgdatabindingtest.MainActivity">
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@{nmg.einText}" />
  </RelativeLayout>
</layout>
```



xmlns:android darf nur einmal vorkommen (Bild 1)

Aus didaktischen Gründen bietet sich an dieser Stelle das Anlegen eines praktischen Beispiels an. Öffnen Sie die im Projektskelett enthaltene Datei *activity_main.xml* und ersetzen Sie ihren Inhalt durch die in Listing 1 gezeigte Passage.

Neben der im vorigen Abschnitt besprochenen Data-Deklaration ist hier auch der Aufbau der Textbox interessant. Sie bekommt statt einem gewöhnlichen Text einen nach dem Schema *@{<String>}* aufgebauten Wert eingeschrieben. Es handelt sich dabei um eine Binding-Expression, die von der Engine während der Programmausführung bearbeitet wird.

Bei der Adaptierung von schon vorhandenen Layoutdateien lauert eine kleine Falle, die sich durch die in Bild 1 gezeigten Fehler der Bauart *Error:(16) Error parsing XML: duplicate attribute* artikuliert. Die Fehler entstehen durch mehrfaches Vorhandensein der Deklaration *xmlns:android="http://schemas.android.com/apk/res/android"* – beachten Sie dabei, dass die von der IDE monierte Datei ein Temporärfile ist, dessen Editieren nicht zur Lösung des Problems führt.

Im nächsten Schritt müssen wir uns der Implementierung einer Datenklasse zuwenden. Für erste Gehversuche reicht eine nach dem POJO-Prinzip aufgebaute Klasse völlig aus. Ergänzen Sie unsere Lösung um die Klassendatei *NMGSample1*, deren Inhalt folgendermaßen aufgebaut ist:

```
package
com.tamoggemon.nmgdatabindingtest;
public class NMGSample1 {
  public final String einText;
  public NMGSample1()
  {
    einText="NMGSample bereit!";

  }
}
```

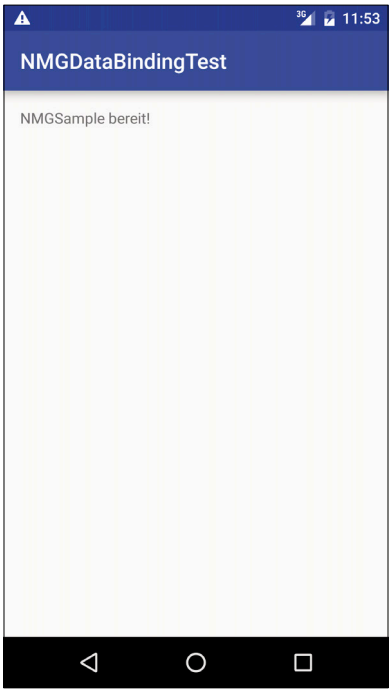
Das eigentliche Beleben der Bindungsbeziehung obliegt der alleinigen Verantwortung des Entwicklers. Öffnen Sie die Datei *MainActivity.java* und passen Sie den Inhalt von *onCreate* folgendermaßen an:

```
public class MainActivity extends
AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActivityMainBinding binding =
    DataBindingUtil.setContentView(this,
    R.layout.activity_main);
    NMGSample1 aClass = new NMGSample1();
    binding.setNmg(aClass);
  }
}
```

ActivityMainBinding ist die vom Framework automatisch generierte Binderhilfsklasse. Ihr Name entsteht durch das Zusammenfassen des Dateinamens der Layoutdatei und Anfügens des Strings *Binding* unter Beachtung der pascalschen Namenskonventionen.

Android Studio kann den (automatisch generierten) Code der Klasse nicht per Reflexion auf den Bildschirm holen. Als Lösung bietet sich hier das manuelle Durchsuchen des Projektverzeichnisses an – auf der Workstation des Autors fand sich der Code in folgenden Unterverzeichnissen:

- *app/build/intermediates/incremental-verifier/debug/com/tamoggemon/nmgdatabindingtest/databinding*
- *app/build/intermediates/classes/debug/com/tamoggemon/nmgdatabindingtest/databinding*



Dank Data Binding wandert der String automatisch in die Textbox (Bild 2)

Die Binding-Hilfsklasse wird im Zusammenhang mit DataBindingUtil zum Anzeigen des Formulars genutzt. Der in normalen Android-Apps notwendige Aufruf von *setContentView* entfällt.

Damit ist unsere App auch schon zum Einsatz bereit (Bild 2).

Ändere dich periodisch

Die Entwickler von Data-Binding-Frameworks müssen in den meisten Programmiersprachen gegen eine systeminhärente Problematik ankämpfen: Sowohl C als auch Java kennen keine Möglichkeit, um eine Variable bei Änderungen Ereignisse emittieren zu lassen. ▶

Als Workaround bietet sich die Realisierung einer hauseigenen Listenerklasse an, die bei korrekter Nutzung die notwendigen Events emittiert und der Binding-Logik so beim Anpassen der Steuerelemente hilft.

Googles Data-Binding-Bibliothek löst dies durch eine Gruppe von *Observable*-Klassen. Für kleinere Aufgaben bietet sich die Nutzung von *BaseObservable* an. Eine Reflexion dieser Klasse zeigt, dass Google hier eine grundlegende Listener-Verwaltung implementiert hat. Das *PropertyChangeRegistry*-Objekt dient dabei als Speicher für die einzelnen Listener:

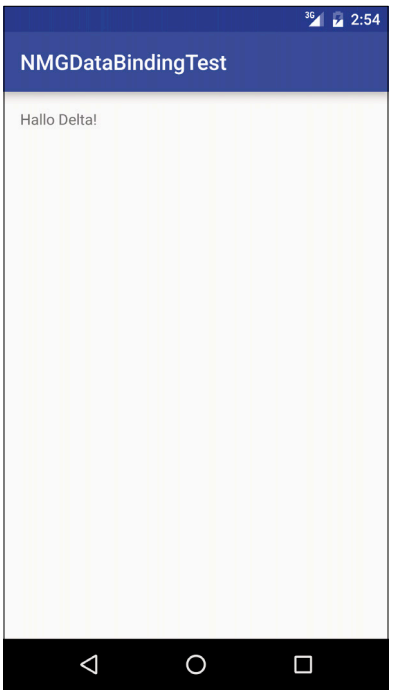
```
public class BaseObservable implements
Observable
{
  private transient
  PropertyChangeRegistry mCallbacks;
  public BaseObservable() {
  }
```

Anmeldungen neuer Listener erfolgen durch die Methode *addOnPropertyChangedCallback*, die die Änderungen direkt in den unterliegenden Speicher weiterschreibt. Das Entfernen von Listenern drucken wir an dieser Stelle aus Platzgründen nicht ab:

```
@Override
public synchronized void addOnPropertyChangedCallback
(OnPropertyChangedCallback callback) {
  if (mCallbacks == null) {
    mCallbacks = new PropertyChangeRegistry();
  }
  mCallbacks.add(callback);
}
```

Viel interessanter ist in diesem Zusammenhang der Rest der Klasse: Google sieht keine Möglichkeit vor, um Änderungen automatisch beim Verändern eines Parameters an die Listener weiterzugeben. Entwickler müssen stattdessen die beiden *notifyChange*-Methoden aufrufen, um die angemeldeten Interessenten zu informieren: *notifyChange()* zeigt eine Änderung aller Parameter an, während *notifyPropertyChanged* für Änderungen eines bestimmten Werts zuständig ist:

```
public synchronized void notifyChange() {
  if (mCallbacks != null) {
    mCallbacks.notifyCallbacks(this, 0, null);
  }
}

public void notifyPropertyChanged(int fieldId) {
  if (mCallbacks != null) {
    mCallbacks.notifyCallbacks(this, fieldId, null);
  }
}
```



Die vom Runnable durchgeführten Änderungen wirken sich auf das Steuerelement aus (Bild 3)

Mit diesem Wissen können wir uns an die Implementierung einer neuen Version von *NMGSample1* heranwagen. Die neue Version der Klasse sieht so aus:

```
public class NMGSample1 extends
BaseObservable{
  private String einText;
  ...
  @Bindable
  public String getEinText() {
    return this.einText;
  }
```

getEinText verhält sich im Großen und Ganzen wie eine Settermethode. Interessant ist in diesem Zusammenhang nur die Annotation *@Bindable*, die den im Hintergrund arbeitenden Codegenerator darüber informiert, dass er es hier mit einem bindbaren Feld zu tun hat. In *setEinText* wird der angelieferte Wert in das Speicherfeld geschrieben. Der Aufruf von *notifyPropertyChanged* setzt ein Flag voraus, das den geänderten Wert identifiziert. Die Data-Binding-Bibliothek generiert eine Klasse namens *BR*, die die diversen Flags enthält:

```
public void setEinText(String _in) {
  this.einText = _in;
  notifyPropertyChanged
  (com.tamoggemon.nmgdatabindingtest.BR.einText);
}
```

Da sich der Name des Feldes aus Sicht der Binding-Engine nicht verändert hat, sind in der Layoutdatei keine Änderungen notwendig. Wer das Programm in der vorliegenden Form ausführt, sieht den aus dem vorigen Schritt bekannten String.

Zum Test des Funktionierens der Bindung ergänzen wir *MainActivity* um einen Handler, der alle paar Sekunden die in einem Runnable enthaltene Textveränderungslogik anwirft. Der dazu notwendige Code sieht so aus:

```
public class MainActivity extends AppCompatActivity {
  Handler myHandler = new Handler();
  NMGSample1 myAkzessor;
```

Neben einer im Haupt-Thread lebenden Instanz der Handler-Klasse legen wir ein Feld an, das dem Runnable den Zugriff auf die Datenbindungsklasse ermöglicht. Die eigentliche Änderungslogik sitzt dann in einem Runnable, das sich durch Aufrufen von *postDelayed* regelmäßig selbst aufruft:

```
Runnable runnableCode = new Runnable() {
  @Override
```

```
  public void run() {
    myAkzessor.setEinText("Hallo Delta!");
    myHandler.postDelayed(runnableCode, 2000);
  }
};
```

Die Integration in *onCreate* besteht im Bevölkern der Variablen und dem Anstoßen des Runnables. Da bei Aufrufen von *post()* kein Delay anfällt, ändert sich der Inhalt der Textbox sofort:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  ...
  NMGSample1 aClass = new NMGSample1();
  ...
  myAkzessor=aClass;
  myHandler.post(runnableCode);
}
```

Damit ist auch die zweite Variante des Data-Binding-Beispiels einsatzbereit. Führen Sie es aus, um die in Bild 3 gezeigten Änderungen im Benutzerinterface zu überprüfen.

In beide Richtungen

Steuerelemente sind nur in den wenigsten Fällen passiv. Sie dienen in vielen Fällen zur Entgegennahme von Eingaben. Als nächste Übung wollen wir einen grundlegenden Leistungsrechner realisieren, der aus den eingegebenen Werten für Strom und Spannung die am Verbraucher anfallende Leistung ausgibt.

Als erster Akt ist eine Anpassung des Layouts nowendig. Öffnen Sie die Datei *activity_main.xml*, um das von Haus aus angelegte RelativeLayout durch ein für unsere Bedürfnisse besser geeignetes LinearLayout zu ersetzen (Listing 2).

Neben dem Einpflegen von zwei EditText-Steuerelementen findet sich hier auch eine geänderte Version des Bindungs-Strings. Mit @= eingeleitete Bindungsbeziehungen stellen eine Zwei-Wege-Bindung dar: Verbindet man sie mit einem Steuerelement, so werden im Widget erfolgende Änderungen automatisch ins Modell zurückgeschrieben.

Die klassische Bindungsvariante mit @ steht nach wie vor zur Verfügung, weil sie weniger Overhead verursacht. Wenn Sie die Daten nur aus dem Modell ins Widget schreiben wollen, können Sie so etwas Platz in der DEX-Datei einsparen.

Veraltete Dokumentation

Google leistet sich hier einen kleinen Schnitzer: Die unter https://developer.android.com/topic/libraries/data-binding/index.html bereitstehenden Informationen sind veraltet. Unter der Hand empfehlen Google-Entwicklerbetreuer die Nutzung des unter https://halfthought.wordpress.com/2016/03/23/2-way-data-binding-on-android bereitstehenden Blogposts.

Die Änderungen im Bereich des Observers fallen überschaubar aus. Anstatt wie bisher ein Element anzulegen, erzeugen wir nun drei von ihnen. Aus Platzgründen drucken wir hier nur den Akzessor für die Spannung ab: Leistung und Strom verhalten sich ähnlich; die Namensgenerierung per Pascal Case funktioniert problemlos:

```
public class NMGSample1 extends BaseObservable{
  private String myU;
  private String myI;
  private String myP;
  public NMGSample1() {  }
  @Bindable
  public String getMyU() {
    return this.myU;
  }
  public void setMyU(String _in) {
    this.myU = _in;
    notifyPropertyChanged
    (com.tamoggemon.nmgdatabindingtest.BR.myU);
  }
  ...
}
```

Wir haben das Data-Binding-Subsystem bisher nur zum Anbinden von Steuerelementen genutzt. Die Bibliothek lässt sich auch direkt mit Methoden verdrahten, was die Realisierung reaktiver beziehungsweise eventgetriebener Systeme ermöglicht. Sie zeichnen sich durch geringere Koppelung aus, was Wartbarkeit und Testbarkeit erhöht. Im Hauptformular unseres Leistungsrechners wird dies durch einen ▶

Listing 2: LinearLayout

```
<layout xmlns:android="http://schemas.android.com/
apk/res/android">
  <data>
    <variable
      name="nmg" type="com.tamoggemon.
      nmgdatabindingtest.NMGSample1"></variable>
  </data>
  <LinearLayout
    ...
    android:orientation="vertical">
    <EditText
      android:layout_width="match_parent"
      android:layout_height="40dp"
      android:text="@={nmg.myU}"/>
    <EditText
      android:layout_width="match_parent"
      android:layout_height="40dp"
      android:text="@={nmg.myI}"/>
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="40dp"
      android:text="@{nmg.myP}" />
  </LinearLayout>
</layout>
```

Event Handler vorgeführt, der als eine Instanz von *OnPropertyChangedCallback* entsteht (Listing 3).

Wer die weiter oben erwähnte Klasse reflektiert, stellt fest, dass das Integer mit dem ID-Wert beim Dispatching der Ereignisse keinerlei Bedeutung hat: Alle Listener bekommen alle anfallenden Ereignisse zugewiesen.

Unsere Methode beginnt aus diesem Grund damit, anhand des übergebenen Flags die Art der Änderung zu erkennen. Das Anliefern eines Nullflags nimmt dabei insofern eine Sonderstellung ein, als wir in diesem Fall von einer Veränderung aller in der Modellklasse enthaltenen Werte ausgehen müssen und dies durch gestaffeltes Abarbeiten von *deltaU* und *deltaI* abbilden. Leider kann der Listener nur dann auf die Inhalte von NMGSample zugreifen, wenn eine globale Zugriffsvariable angelegt wird. Dazu ist folgende Codeänderung notwendig:

Listing 3: Event Handler

```
public class MainActivity extends AppCompatActivity
{
  Observable.OnPropertyChangedCallback myCallback
  = new Observable.OnPropertyChangedCallback() {
    @Override
    public void onPropertyChanged(Observable
    sender, int propertyId){
      if(propertyId==0){
        deltaU();
        deltaI();
      }
      else if(propertyId==
      com.tamoggemon.nmgdatabindingtest.BR.myU){
        deltaU();
      }
      else if(propertyId==
      com.tamoggemon.nmgdatabindingtest.BR.myI){
        deltaI();
      }
    }
  };
```

```
public class MainActivity extends AppCompatActivity
{
  NMGSample1 myAkzessor;
```

Damit ist die Arbeit an der Infrastruktur im Großen und Ganzen abgeschlossen. In *onCreated* wird die Activity im ersten Schritt bevölkert, darauf folgt ein Aufruf von *addOnPropertyChangedCallback*. Die Methode meldet das Callback bei der Data-Binding-Engine an und schaltet es scharf:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  ...
  binding.setNmg(aClass);
  myAkzessor=aClass;
  aClass.addOnPropertyChangedCallback(myCallback);
}
```

DeltaU wäre eigentlich primitiv. Leider gibt es bei Data-Binding-Engines das Problem, dass während des Baus des Formulars beziehungsweise des Modells leere Elemente anfallen können, die bei der Programmausführung zu NullReference-Exceptions führen. Aus diesem Grund nutzen wir eine *if*-Selektion, die die Methodenausführung bei einem ungültigen Betriebszustand vorzeitig beendet:

```
private void deltaU()
{
  Log.e("NMG", "Delta U aufgerufen");
  if(myAkzessor.getMyU()==null ||
  myAkzessor.getMyI()==null)return;
  float u=Float.parseFloat(myAkzessor.getMyU());
  float i=Float.parseFloat(myAkzessor.getMyI());
  myAkzessor.setMyP(new Float(u*i).toString());
}
```

DeltaI ist im Grunde eine Weiterleitung an *deltaU*. Aus didaktischen Gründen rufen wir die *Log*-Methode auf, um eine weitere Statusmeldung in die Konsole abzusetzen:

```
private void deltaI()
{
  Log.e("NMG", "Delta I aufgerufen");
  deltaU();
}
```

Im Internet finden sich immer wieder Codebeispiele, die eine Zwei-Wege-Bindung unter Nutzung eines von Hand programmierten Adapters realisieren. Diese vergleichsweise aufwendige Vorgehensweise ist nicht mehr notwendig, da die Bibliothek seit einigen Monaten Zwei-Wege-Bindungen über das im Artikel beschriebene @=-Makro direkt unterstützt.

Android-Code erkennt man seit Jahr und Tag daran, dass der Entwickler die im DOM-Baum befindlichen Steuerelemente über einen mehr oder weniger komplexen Block von *findViewByID*-Aufrufen ansprechbar macht. In anderen Programmierumgebungen (Stichwort .NET) ist dies dank automatischem Data Binding nicht nowendig.

Wer sein Programm auf Basis der hier besprochenen Data-Binding-Engine aufbaut, kann sich diese lästige Arbeit ersparen. Als Beispiel dafür wollen wir den soeben realisierten Leistungsrechner noch um einen Button erweitern, in dessen Code etwas Schindluder mit den Steuerelementen getrieben wird. Öffnen Sie die Layoutdatei abermals, um folgende Deklarationen einzufügen:

```
<LinearLayout ...>
  <EditText ...
```

```
    android:id="@+id/TxtU"/>
  <EditText ...
    android:id="@+id/TxtI"/>
  ...
  <Button
    android:layout_width=
    "match_parent"
    android:layout_height="40dp"
    android:text="Los!"
    android:onClick=
    "@{nmg.onClickLos}"/>
</LinearLayout>
```

Da nicht alle Steuerelemente aus dem Code Behind ansprechbar sein müssen, ist das Einfügen von ID-Attributen unter Android optional. Alle mit einem solchen Attribut ausgestatteten Steuerelemente werden vom Generator automatisch in die Bindingklasse übertragen.

Dieses Verhalten lässt sich prüfen, indem man die Datei *ActivityMainBinding.java* in einem Editor öffnet – falls Sie das Layout eins zu eins übernommen haben, finden sich dort die folgenden beiden Membervariablen:

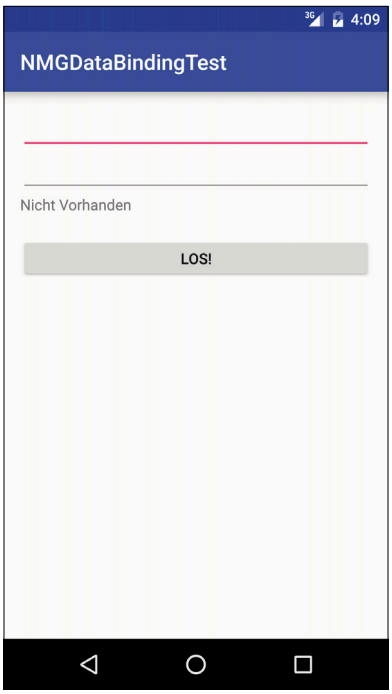
```
public class ActivityMainBinding extends
android.databinding.ViewDataBinding  {
  ...
  // views
  public final android.widget.EditText TxtI;
  public final android.widget.EditText TxtU;
```

Ein weiteres sehr elegantes Feature ist die Möglichkeit, Event Handler direkt aus der XML-Datei heraus anzulegen. Dabei tritt allerdings ein kleines Problem auf: Während die anhand des ID-Parameters generierten Felder im Binding-Objekt liegen, sitzen die Event Handler im Modell.

Der einfachste Lösungsweg hierfür ist das abermalige Anliegen einer Akzessorvariablen, die dann im Rahmen der Belebung des Modells wiederum mit Informationen befüllt werden muss:

```
public class NMGSample1 extends BaseObservable{
  public ActivityMainBinding myBindingAkzessor;
  ...
  public void onClickLos(View view)
  {
    myBindingAkzessor.TxtI.setText("0.0");
    myBindingAkzessor.TxtU.setText("0.0");
  }
}
```

Damit sind wir an dieser Stelle fertig. Passen Sie *onCreate* wie im Folgenden gezeigt an, um die notwendige Infrastruktur bereitzustellen. Nach getaner Arbeit lassen sich die eingegebenen Werte durch Drücken des Buttons ausnullen:



Dank Bindungsintelligenz wird ein nicht vorhandener Wert durch einen String ersetzt (Bild 4)

```
@Override
protected void onCreate(Bundle
savedInstanceState) {
  ...
  myAkzessor=aClass;
  aClass.myBindingAkzessor=binding;
  aClass.addOnPropertyChangedCallback
  (myCallback);
}
```

Bindende Arithmetik

Die in geschweifte Klammern gefassten Bindungs-Strings stellen keine reine Zuweisung dar: Sie werden vielmehr zur Laufzeit in eine für den Java-Compiler verständliche Form transpiliert.

Die vom Transpiler verstandene Sprache umfasst einen Gutteil des Java-Sprachumfangs. Zum Zeitpunkt der Drucklegung werden folgende Operatoren unterstützt:

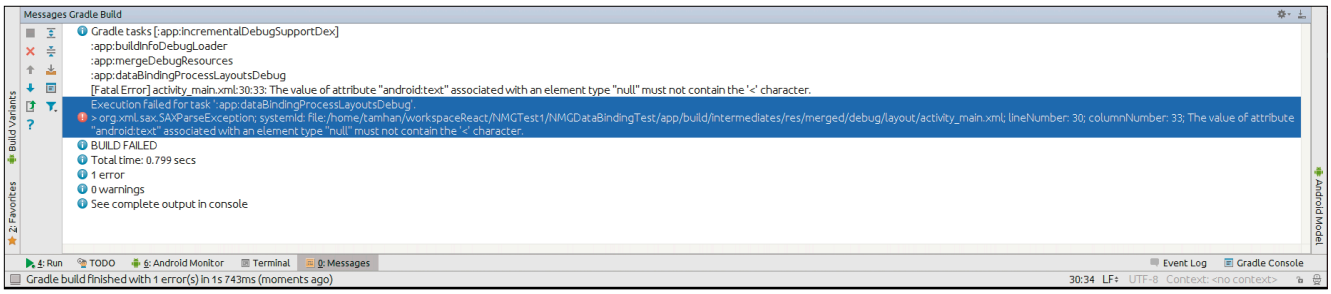
- + - / * % >> >>> << () als Berechnungsoperator
- + zur String-Konkatenation
- && || == > < >= <= für Logik
- & | ^ für boolesche Operationen
- Unäre Operatoren + - ! ~
- instanceof
- Literale der Typen *character, String, numeric* und *null*
- Casts
- Methodenaufrufe
- Feld- und Array-Zugriffe
- Ternär-Selektionsoperator *?:*

Beachten Sie dabei, dass die folgenden drei Befehle nicht erlaubt sind: *this, super* und *new*.

Die Nutzung des Transpilers ermöglicht einige interessante Konstruktionen, die den im Hintergrund anfallenden Code reduzieren. Als erstes Beispiel wollen wir eine Bindung realisieren, die den Leistungswert bei Nicht-Vorhandensein durch einen String ersetzt. Dazu ist in der XML-Datei folgende Änderung notwendig:

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{nmg.myP != null ? nmg.myP :
  "Nicht Vorhanden"}' />
```

Google zeigt in der offiziellen Dokumentation der Bibliothek nur Beispiele, in denen die Strings aus der Datenbindungsklasse entnommen werden. Wer die Zeichenketten direkt einbinden möchte, kann auf einen aus der JavaScript-Welt bekannten Trick zurückgreifen: Wird der äußere String ▶



Wer das Escape-Zeichen vergisst, wird vom XML-Checker ausgebremst (Bild 5)

durch ' terminiert, so kann man innen auf doppelte Anführungszeichen setzen. Das Ausführen des vorliegenden Codes führt zum in Bild 4 gezeigten Output.

In der Dokumentation finden sich arithmetische Vergleiche, die direkt in der XML-Datei angelegt werden. Ein Beispiel dafür wäre folgendes Snippet:

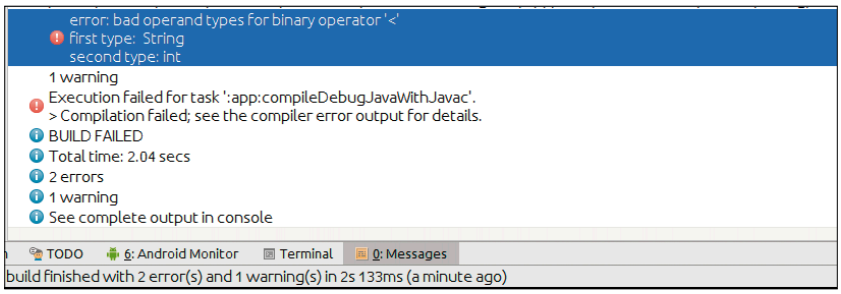
```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{nmg.myP < 20 ? nmg.myP :
  "Nicht Vorhanden"}' />
```

Wer den vorliegenden Code in die XML-Datei einfügt und das File kompiliert, wird daraufhin die in Bild 5 gezeigte Fehlermeldung erhalten.

Zur Umgehung dieses Problems bietet sich die aus klassischen XML bekannte Methode der Escapezeichen an – eine kompilierbare Version des Größenvergleichs sähe so aus:

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{nmg.myP &lt; 20 ? nmg.myP :
  "Nicht Vorhanden"}' />
```

Im Fall unseres vorliegenden Beispiels funktioniert auch diese Variante nicht: *myP* ist ein String, während numerische Vergleiche nur mit Zahlenwerten funktionieren. Android Studio erkennt dies während der Kompilation und weist darauf mit einem nach dem in Bild 6 gezeigten Schema aufgebauten Fehler hin.



Inkompatible Typen führen durch die Transpilation zu Fehlern bei der Kompilation der Binding-Klasse (Bild 6)

Da die Arbeit mit dem *?:*-Operator nicht jedermanns Sache ist, Abfragen der Art *Existiert X* allerdings häufig auftreten, spendiert Google einen darauf optimierten Operator. Seine Nutzung sieht folgendermaßen aus:

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{nmg.myP ?? "Nicht Vorhanden"}' />
```

Diese nur in Binding-Strings zulässige Syntax ersetzt die angewiesene Variable durch den angegebenen Defaultwert, wenn das angewiesene Bindungsziel *null* ist. Angesichts der Möglichkeit des Auftretens von NullReferenceExceptions findet sich der Operator in praktischem Code relativ häufig. Wer von anderen Personen erzeugte Programme wartet, sollte sich die Syntax einprägen.

Unübliche Bindungen

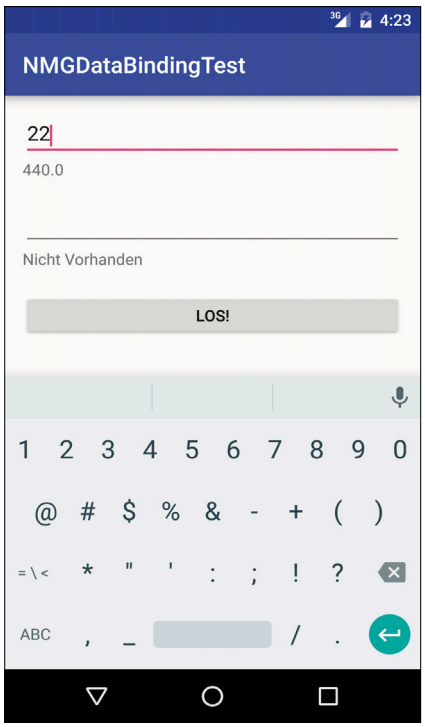
Ein weiteres bemerkenswertes Feature ist das direkte Realisieren von Arithmetik. Ein Beispiel dafür wäre die folgende Erweiterung der XML-Datei:

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{Float.toString(Float.parseFloat
  (nmg.myU ?? "0.0")*20)}' />
```

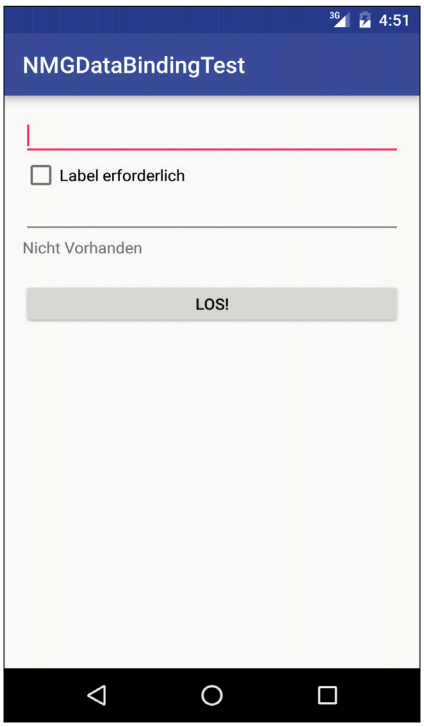
Dieser Bindungs-String geht aus systemtechnischer Sicht an die Grenzen des Sinnvollen: Ab einem gewissen Komplexitätsgrad ist es vernünftiger, die Logik in Form von leichter dokumentierbaren und debuggbaren Code Behinds anzulegen.

Schon aus didaktischen Gründen wollen wir das Element trotzdem Schritt für Schritt durchgehen. Als Innerstes findet sich ein *??*-Operator, der Nullwerte in *myU* durch einen leeren String ersetzt und so eine NullReferenceException vermeidet.

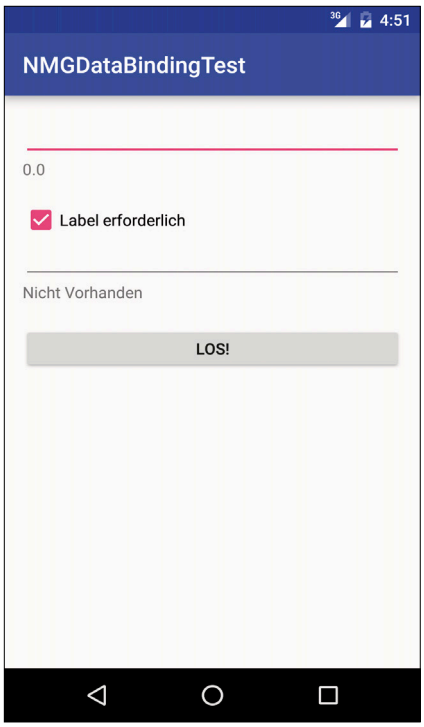
Dieser sichere String wird im nächsten Schritt in *Float.parseFloat* eingefügt, wo er in einen numerischen Wert konvertiert wird. Nach einer Multiplikation mit der Konstante *20* folgt ein Aufruf von *Float.To-*



Dank Bindungsartihmetik kommt dieses Label ohne Code Behind aus (Bild 7)



Das Label verschwindet beim Deaktivieren der Checkbox ... (Bild 8)



... um danach wieder am Bildschirm zu erscheinen (Bild 9)

String, um das Konvolut in ein für die Text-Bindung verständliches Format zu bringen. Wenn Sie die neue Version des Programms ausführen und einen Wert für die Spannung eingeben, so sehen Sie, dass das Feld wie in Bild 7 gezeigt mit einem inkrementierten Wert bevölkert wird.

Data Binding ist nicht auf das Festlegen von Texten beschränkt. Ein weiteres nettes Beispiel ist das Sichtbarmachen von Steuerelementen in Abhängigkeit von im Modell gespeicherten Werten.

Dazu muss der *Data*-Abschnitt im ersten Schritt um ein *Import*-Tag erweitert werden. Es handelt sich dabei um eine Anweisung an die Binding-Engine, die in der Klasse enthaltenen Konstanten und Methoden für Binding-Strings ansprechbar zu machen:

```
<layout xmlns:android=
"http://schemas.android.com/apk/res/android">
  <data>
    <...
    <import type="android.view.View"/>
  </data>
```

Das aus dem vorigen Abschnitt bekannte TextView-Steuerelement bekommt nun ein *Visibility*-Attribut eingeschrieben, das ebenfalls mit einem Data-Binding-String verbunden wird. Er nutzt die per Import eingebundene Viewklasse, um auf die Konstanten *VISIBLE* und *GONE* zurückzugreifen:

```
<LinearLayout ...>
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{Float.toString(Float.parseFloat
  (nmg.myU ?? "0.0")*20)}'
  android:visibility=
  "@{ChkAugment.checked ? View.VISIBLE : View.GONE}"/>
```

Damit fehlt uns nur noch die Checkbox, die mit einem ID-Attribut ausgestattet wird. Würde dieses fehlen, so könnten wir sie nicht aus dem Binding-String heraus ansprechen:

```
<CheckBox
  android:layout_width="match_parent"
  android:layout_height="30dp"
  android:text="Label erforderlich"
  android:id="@+id/ChkAugment"/>
```

An dieser Stelle wartet ein kleines Gotcha: Wer den Binding-String auf Basis der Methode *isChecked* aufbaut, erhält eine nicht funktionierende Bindung. Das liegt daran, dass das Betriebssystem ja keinen Weg hat, um Änderungen im Wert von *isChecked* zu erreichen.

Wer dem Autor nicht glaubt, kann die folgende Version des Codes ausprobieren:

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="40dp"
  android:text='@{Float.toString
  (Float.parseFloat(nmg.myU ?? "0.0")*20)}'
  android:visibility="@{ChkAugment.isChecked() ?
  View.VISIBLE : View.GONE}"/>
```

▶

Damit ist auch diese Version des Programms einsatzbereit. Bild 8 und Bild 9 zeigen, dass die Sichtbarkeit des Steuerelements ohne vom Entwickler zu schreibenden Code angepasst wird.

Sprechen Sie Bean?

Java Beans sind im Grunde genommen etwas bessere Klassen, die ihre privaten Mitglieder zugunsten eines Containers über ein spezifiziertes Format exponieren. Googles Data-Binding-Bibliothek ist im Rahmen des Aufbaus einer Data-Binding-Beziehung zur Reflexion befähigt: Das folgende Objekt würde den Beziehungs-String *firstName* erfüllen:

```
public class User {
  private final String firstName;
  ...
  public String getFirstName() {
    return this.firstName;
  }
}
```

Die Intelligenz der Data-Binding-Bibliothek ist nicht auf das Verdrahten von Steuerelementen beschränkt. Google spendiert Entwicklern eine Gruppe von Spezialklassen, die das Absetzen von Ereignissen für bestimmte Datenfelder automatisieren.

Der eigentliche Aufbau der Klassen ist in allen Fällen identisch, weshalb wir hier mit der Reflexion der für Booleans zuständigen Variante vorlieb nehmen wollen:

```
public class ObservableBoolean extends BaseObservable
implements Parcelable, Serializable {
  ...
  public ObservableBoolean() {
  }
```

Der Getter funktioniert dabei im Großen und Ganzen so, wie man es anhand der Beschreibung erwarten würde. Interessant ist, dass der Setter die zum Absetzen einer Änderungsmeldung notwendige Funktion *notifyChange()* automatisch aufruft:

```
  public boolean get() {
    return mValue;
  }

  public void set(boolean value) {
    if (value != mValue) {
      mValue = value;
      notifyChange();
    }
  }
```

Neben Booleans bietet Google auch ein gutes Dutzend weiterer Versionen an. Tabelle 1 bietet eine Kurzbeschreibung aller im Namespace enthaltenen Klassen, die für die Verwaltung einzelner Datentypen zuständig sind. Zudem gibt es auch noch einige Varianten, die Felder, Listen und Maps abbilden. An dieser Stelle bietet sich das Anpassen der weiter oben erstellten Datenklasse an. Der Korpus von *NMGSample1* wird durch die Nutzung der *ObservableField*-Klassen folgendermaßen reduziert:

Tabelle 1: Kurzbeschreibung der Klassen

Klasse	Rolle
ObservableBoolean	Handhabt Feld vom Typ *boolean*.
ObservableByte	Handhabt Feld vom Typ *byte*.
ObservableChar	Handhabt Feld vom Typ *char*.
ObservableDouble	Handhabt Feld vom Typ *double*.
ObservableField<T>	Handhabt Feld mit generischen Daten.
ObservableFloat	Handhabt Feld vom Typ *float*.
ObservableInt	Handhabt Feld vom Typ *int*.
ObservableLong	Handhabt Feld vom Typ *long*.
ObservableParcelable	Handhabt Parcelables.
ObservableShort	Handhabt Feld vom Typ *short*.

```
public class NMGSample1{
  public ObservableField<String> myU;
  public ObservableField<String> myI;
  public ObservableField<String> myP;
  public NMGSample1() {
    myU=new ObservableField<String>();
    myI=new ObservableField<String>();
    myP=new ObservableField<String>();
  }
}
```

In der für die Berechnung zuständigen Methode *deltaU* sind insofern Änderungen erforderlich, als wir nun die Getter-Methoden aufrufen müssen:

```
private void deltaU() {
  Log.e("NMG", "Delta U aufgerufen");
  if(myAkzessor.myU.get()==null ||
  myAkzessor.myI.get()==null)return;
  float u=Float.parseFloat(myAkzessor.myU.get());
  float i=Float.parseFloat(myAkzessor.myI.get());
  myAkzessor.myP.set(new Float(u*i).toString());
}
```

Die in der Modellklasse entstehende Vereinfachungen werden durch etwas längeren Code im Hauptformular bezahlt. Der Korpus von *onCreate* sieht nun folgendermaßen aus:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
  ...
  myAkzessor=aClass;
  aClass.myBindingAkzessor=binding;
  aClass.myI.addOnPropertyChangedCallback(myCallback);
  aClass.myU.addOnPropertyChangedCallback(myCallback);
}
```

Aufgrund des Einsatzes von zwei dedizierten Feldern gibt es keinen zentralisierten Einsprungpunkt mehr, der für das Absetzen von Änderungsereignissen zuständig ist. Angesichts des sehr einfachen Berechnungsprozesses umgehen wir dieses Problem mit einem kleinen Trick: Jede Änderung löst eine komplette Rekalkulation aus.

In praktischem Code würde es sich stattdessen anbieten, mehrere Instanzen von *OnPropertyChangedCallback* anzulegen:

```
Observable.OnPropertyChangedCallback myCallback = new
Observable.OnPropertyChangedCallback() {
  @Override
  public void onPropertyChanged(Observable sender,
  int propertyId)
  {
    deltaU();
  }
};
```

Damit ist auch diese Variante des Codes einsatzbereit. Wir drucken an dieser Stelle allerdings keinen Screenshot ab, weil sich das visuelle Verhalten des Programms nicht vom Vorhergehenden unterscheidet.

Die für Ihre Anwendung ideale Vorgehensweise ist von Fall zu Fall unterschiedlich. Manche Programme profitieren von der manuellen Implementierung einer Datenklasse. Achten Sie allerdings darauf, im Interesse der Konsistenz ein einmal ausgewähltes Verfahren in der gesamten Applikation beizubehalten.

Listensystem von Android

Direktes Binding gegen Steuerelemente ist insofern einfach, als keine komplexen Beziehungen erforderlich sind. Das für seine relativ eigenwillige Vorgehensweise bekannte Listensystem von Android dient als ideale Grundlage für unser Meisterstück. Die letzte Applikation dieser Übung möchte eine Liste generieren, deren einzelne Elemente per Data Binding in Modellklassen vorgehalten werden.

Das Hauptformular der Applikation beginnt mit der Einbindung des magischen URL http://schemas.android.com/apk/res-auto. Er verweist immer auf das Paket der gerade aktuellen Applikation:

```
<?xml version="1.0" encoding="utf-8"?>
<layout
  xmlns:android="http://schemas.android.com/apk/res/
  android"
  xmlns:bind="http://schemas.android.com/apk/
  res-auto">
  <data>
    <variable name="liststore" type=
    "com.tamoggemon.nmgdatabindinglist.ListStore"/>
  </data>
```

Im Layout findet sich eine Instanz von *ListView*, die über den Bindungs-String *bind:items* mit einer noch anzulegenden Datenklasse verbunden wird:

```
<LinearLayout
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  ...
  tools:context=
  "com.tamoggemon.nmgdatabindinglist.MainActivity">
  <ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    bind:items="@{liststore.myStore}"/>
</LinearLayout>
</layout>
```

Listing 4: listitem.xml

```
<LinearLayout android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <CheckBox
    android:layout_width="40dp"
    android:layout_height="wrap_content"
    android:checked="@{listclass.flag}"/>
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="@{listclass.name}"/>
</LinearLayout>
</layout>
```

Die anzuzeigenden Zeilen entstehen – wie unter Android üblich – in einer eigenen XML-Datei, die auf den Namen *listitem.xml* hört. Sie beginnt mit der Einbindung einer Modellklasse, die die für das einzelne Element zuständigen Informationen vorhält:

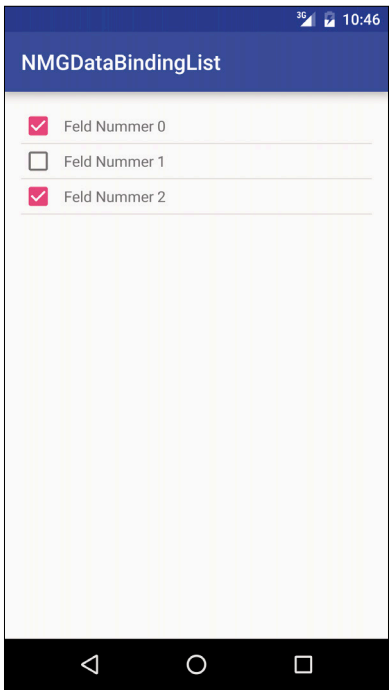
```
<?xml version="1.0" encoding="utf-8"?>
<layout
  xmlns:android="http://schemas.android.com/apk/res/
  android">
  <data>
    <variable name="listclass" type=
    "com.tamoggemon.nmgdatabindinglist.ListClass"/>
  </data>
```

Im Bereich der Steuerelemente findet sich keine Wissenschaft. Sowohl die Checkboxen als auch die *TextView* erhalten einen Bindungs-String, der auf das zum jeweiligen Element gehörende Modell verweist (Listing 4).

ListStore nutzt die in der Data-Binding-Bibliothek enthaltene Klasse *ObservableArrayList*. Sie implementiert ein Array, das beim Hinzufügen und/oder Entfernen von Elementen Ereignisse emittiert und der Engine so das Anpassen von damit verbundenen Steuerelementen erlaubt.

Aus didaktischen Gründen legen wir im Konstruktor zudem drei Elemente an, um gleich nach dem Programmstart eine bevölkerte Liste vorzufinden:

```
public class ListStore {
  public ObservableArrayList<ListClass>
  myStore = new
  ObservableArrayList<>();
  public ListStore(){
    myStore.add(new ListClass());
    myStore.add(new ListClass());
    myStore.add(new ListClass());
  }
}
```



Die im Modell angelegten Items erscheinen in der Liste (Bild 10)

ListBinder erzeugt einen Binding-Adapter, der für das Weiterreichen der Daten zwischen Steuerelement und Bindungs-Engine zuständig ist. Der in der Annotation angegebene String beschreibt dabei den in der XML-Datei zu verwendenden Namen. Achten Sie auf korrekte Parametertypen:

```
public class ListBinder {
  @BindingAdapter("bind:items")
  public  static void bindList
  (ListView view, ObservableArrayList
  <ListClass> list) {
    NMGListAdapter adapter =
    new NMGListAdapter();
    adapter.myBacking=list;
    view.setAdapter(adapter);
  }
}
```

Damit können wir wieder zur standardisierten Arbeit zurückgehen. Listen setzen in Android eine Adapterklasse voraus, die zwischen dem Datenspeicher und dem Anzeigesteuerelement makelt. Hier nutzen wir aus Bequemlichkeit die vom Betriebssystem bereitgestellte Klasse *BaseAdapter* (Listing 5).

Die ersten drei Methoden sind Standardcode, der in jeder Adapterklasse gleichermaßen eingesetzt wird. Der Inhalt von *getView* ist insofern interessanter, als wir nun anstelle der für die Anzeige zuständigen Strukturen eine Data-Binding-Klasse erzeugen. Die Methode *getRoot* liefert das Materelement der in der jeweiligen Klasse enthaltenen Struktur zurück und ermöglicht dem GUI-Stack das Einbinden des neu generierten Elements in die Liste:

```
@Override
public View getView(int position, View convertView,
ViewGroup parent) {
  if (myInflater == null) {
    myInflater = (LayoutInflater) parent.getContext()
    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
  }
  ListitemBinding binding = DataBindingUtil.inflate
  (myInflater, R.layout.listitem, parent, false);
  binding.setListclass(myBacking.get(position));
  return binding.getRoot();
}
}
```

Die für die einzelnen Elemente zuständige Modellklasse ist ebenfalls nicht sonderlich kompliziert. Aus Bequemlichkeitsgründen nutzen wir hier die weiter oben angesprochenen Convenience-Klassen. Der vergleichsweise lange Code entsteht durch die Realisierung eines Konstruktors, der anhand eines statischen Variable für etwas Varianz in den generierten und angezeigten Elementen sorgt:

```
public class ListClass {
  public ObservableBoolean flag;
  public ObservableField<String> name;
  static int counter=0;
  public ListClass() {
    flag=new ObservableBoolean();
```

Listing 5: Adapterklasse

```
public class NMGListAdapter extends BaseAdapter {
  public ObservableArrayList<ListClass> myBacking;
  private LayoutInflater myInflater;
  @Override
  public int getCount() {
    return myBacking.size();
  }
  @Override
  public Object getItem(int position) {
    return myBacking.get(position);
  }
  @Override
  public long getItemId(int position) {
    return position;
  }
```

```
    name=new ObservableField<String>();
    if(counter%2==0)this.flag.set(true);
    else this.flag.set(false);
    name.set("Feld Nummer " + counter);
    counter++;
  }
}
```

onCreate erzeugt die beiden Linkklassen. Die Belebung des als XML-Datei vorliegenden Layouts erfolgt über die weiter oben besprochenen Methoden der Data-Binding-Engine:

```
public class MainActivity extends AppCompatActivity {
  ListStore aStore;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActivityMainBinding binding = DataBindingUtil.
    setContentView(this, R.layout.activity_main);
    aStore = new ListStore();
    binding.setListstore(aStore);
  }
}
```

Damit ist auch diese Variante des Codes einsatzbereit. Bild 10 zeigt, dass die im Modell vorliegenden Informationen automatisch weitergeleitet werden. Weil das manuelle Zusammenbauen von ListViews nicht jedermanns Sache ist, steht unter der Adresse https://github.com/evant/binding-collection-adapter ein Repository bereit, das eine Gruppe von Hilfsklassen anbietet.

Fazit

Als Microsoft Data Binding mit XAML einführte, wunderten sich Entwickler über den Sinn der Technik – mittlerweile ist das Verfahren vollständig implementiert. Wer den (zugegebenermaßen nicht trivialen) Lernaufwand hinter sich gebracht hat, will die durch Data Binding entstehenden Vereinfachungen im Allgemeinen nicht missen. Die aus Wartungs- und Qualitätssicherungssicht beste Codezeile ist und bleibt die, die gar nicht erst geschrieben wird. ■

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Er lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter:
www.tamoggemon.com

SKALIERENDE ANWENDUNGEN MIT MICROSOFT ORLEANS

Lokal oder in der Cloud

Nie war es einfacher, Anwendungen zu entwerfen, die gleichermaßen lokal oder unter Microsoft Azure laufen können.

Wäre es nicht schön, wenn das Schreiben verteilter, skalierbarer Anwendungen einfacher wäre und man den dafür jedesmal erforderlichen Overhead so weit wie irgend möglich einfach hinter einer schönen Abstraktion verschwinden lassen könnte?

Genau das ist der Anspruch, den das in Microsoft Research beheimatete Projekt Orleans erfüllen will. Nach seiner Veröffentlichung als Open Source Anfang 2015 erreichte das Projekt daher auch schnell eine gewisse Popularität unter .NET-Entwicklern.

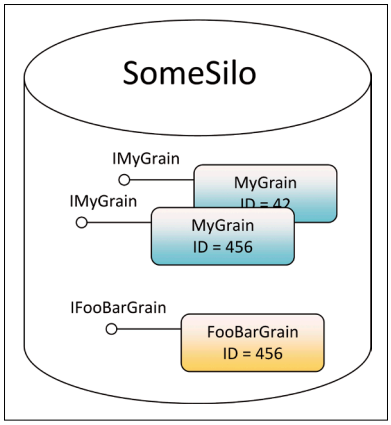
Viele Konzepte, die sich in Orleans wiederfinden, sind nicht unbedingt neu. Wie viele Projekte und Produkte in der schnelllebigen IT-Welt baut auch Orleans auf Erkenntnissen und Ideen auf, die in älteren Projekten gewonnen wurden. Eines dieser Konzepte ist das der aus Erlang, Akka und Scala bekannten Aktoren. Die Entsprechung in Orleans hört auf den Namen Grain. Das ist nun allerdings nicht einfach nur eine andere Bezeichnung. Grains verhalten sich zwar sehr ähnlich zu herkömmlichen Aktoren, sind aber dennoch anders.

In Orleans werden Grains auch als virtuelle Aktoren bezeichnet – virtuell deshalb, weil Grain-Instanzen scheinbar immer existieren. Grains werden weder erstellt noch zerstört, sie können lediglich aktiv oder inaktiv sein. Das System abstrahiert Erzeugung und Zerstörung der entsprechenden Instanzen nach außen hin vollständig. Fordert man einen Verweis auf eine Grain an, so ist nicht ohne Weiteres erkennbar, ob die Instanz soeben erstmalig erstellt wurde, ob sie deserialisiert wurde oder ob sie vielleicht bereits vor dem Aufruf geladen und aktiv war. Konsequent kann (und muss) man als Entwickler deshalb einfach davon ausgehen, dass eine bestimmte, gerade benötigte Grain per se vorhanden ist.

Bild: shutterstock / Jiri Perina

Grains unter der Lupe

Was steckt nun aber genau in einer solchen Grain? Eine Grain ist zunächst nur ein einfaches Objekt, das eine abgegrenzte Aufgabe ausführt. Ein typisches Beispiel wäre eine Grain, die einen konkreten Kunden darstellt und dessen Daten managt. Nach außen werden verfügbare Optionen über ein Interface offengelegt. Jede Grain hat eine ID, diese kann mit Bordmitteln ein Integer, eine GUID oder eine Zeichenkette sein. Mit etwas mehr Aufwand sind auch zusammengesetzte IDs möglich, aber die drei eingebauten Optionen reichen für viele Anwendungsfälle bereits aus.

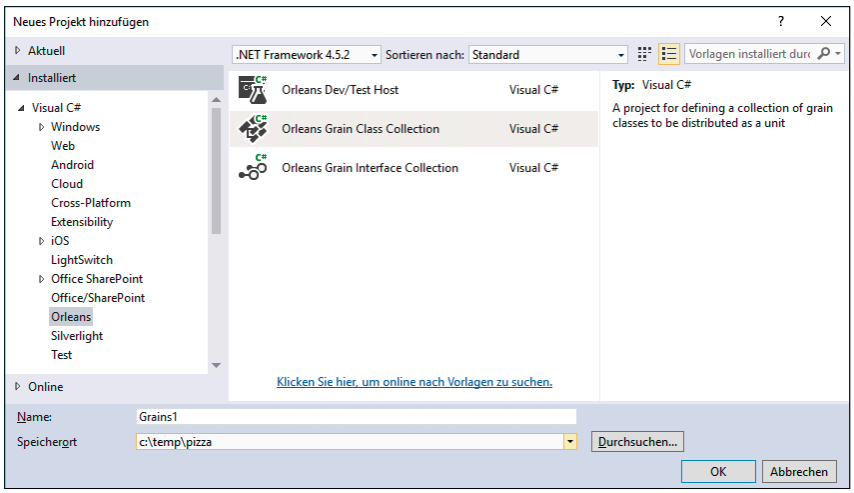


Grains leben in Silos (Bild 1)

Benötigt man eine Singleton-artige Grain, hat sich hierbei als Konvention die Verwendung einer Integer-ID mit dem Wert *0* eingebürgert. Neben der ID kann eine Grain auch einen persistenten internen Status haben, dazu später mehr. Auch statuslose Grains oder solche mit lediglich transientem Status sind durchaus üblich und sinnvoll.

Wie bereits erwähnt, werden Grains aus semantischer Sicht weder erzeugt noch zerstört. Konsequenterweise programmiert man für Grains auch keine Konstruktoren oder Finalizer. Das klingt zunächst ungewohnt, de facto reduziert dies jedoch sogar die Komplexität des Systems.

Ebenfalls die Komplexität reduzierend wirkt die Forderung, dass analog zum Aktoren-Modell eine Grain ihre Aufgabe stets in einem einzelnen Thread ausführt. Orleans stellt sicher, dass es keine konkurrierenden Zugriffe auf Grain-Instanzen gibt, somit entfällt automatisch jegliche Notwendigkeit für Locking.



Orleans-Vorlagen in Visual Studio (Bild 2)

Da unser Ziel ja eine verteilte und möglichst gut skalierende Anwendung ist, solllten Grains idealerweise als eher leichtgewichtige Objekte designt werden. Weil Grains im Cluster anhand ihrer ID auf einzelne Nodes verteilt werden und ihre Methoden innerhalb Orleans de facto die kleinste Ausführungseinheit darstellen, wäre es folglich eher kontraproduktiv, schwergewichtige Grains zu designen.

Stattdessen sind kleine, kurz laufende Methoden das Ideal, um eine im Cluster gut skalierende Applikation zu erhalten. Auf der anderen Seite muss man sich natürlich auch vor Augen halten, dass die Kommunikation zwischen einzelnen Grains über ein Message-Passing-Verfahren stattfindet. Folglich gilt es beim Design, eventuell auftretende Latenzen mit einzupreisen.

Um die zugedachte Funktion optimal ausführen zu können, leben Grains ihrerseits in größeren Einheiten, sogenannten Silos (Bild 1). Ein solcher Silo ist nichts anderes als ein Anwendungscontainer, der die notwendige Laufzeitumgebung für unsere Grains zur Verfügung stellt. Typischerweise läuft auf einer Maschine in der Produktivumgebung auch nur genau ein solcher Orleans-Silo, in dem beliebig viele Grains leben können.

Falls wir einen Cluster aus mehreren Silos betreiben, stimmen sich die Silos automatisch untereinander ab, welche Grain-Instanz in welchem Silo aktiviert wird. Das stellt zwei Dinge sicher: Zum einen eine möglichst optimale Verteilung der anfallenden Last, zum anderen wird eine bestimmte Grain-Instanz auch immer nur in genau einem Silo aktiviert.

Die erste Orleans-Grain

Bevor wir überhaupt loslegen können, sollten wir unbedingt die Microsoft Orleans Tools for Visual Studio installieren. Theoretisch ginge es zwar auch ohne diese Templates, doch nehmen uns diese wirklich eine ganze Menge Handarbeit in puncto Einrichtung der Visual Studio Solution ab. Die Templates bestehen aus drei Projekten (Bild 2).

Um nun eine erste Grain zu erstellen, müssen wir zuerst das Interface dafür definieren und greifen daher zu *Orleans Grain Interface Collection*. Im erzeugten Assembly benennen wir die Datei *IGrain1.cs* und das darin enthaltene leere Interface auf *IPerson* um. Als wesentliches Merkmal einer Person gilt ihr Name, weswegen wir einen Getter und einen Setter hierfür definieren (Listing 1).

Der aufmerksame C#-Programmierer erkennt in diesen wenigen Zeilen sofort zwei Besonderheiten. Die eine Besonderheit ist, dass wir gar keine Property definiert haben, sondern tatsächlich zwei Methoden. Dies ist eine Einschränkung von Orleans, die auf die Übergabe der Aufrufe mittels Message Passing zurückzuführen ist. Eine Property würde in dem Fall nicht funktionieren, weshalb Orleans diese Möglichkeit gar nicht erst vorsieht.

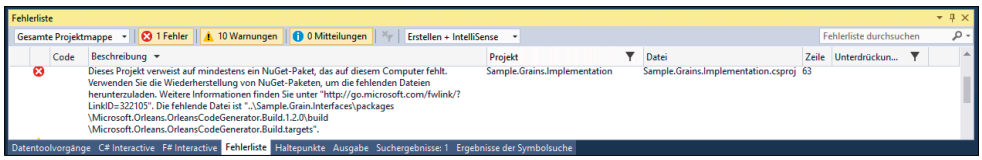
Die zweite Besonderheit ist der Rückgabewert der beiden Routinen *Task* beziehungsweise *Task<Typ>*. Auch dies ist ein grundlegendes Konzept in Orleans, denn alle veröffentlichten Methoden einer Grain sind immer asynchron aufrufbar. Ungeachtet der Tatsache, dass eine Grain-Methode immer in genau einem Thread läuft, macht Orleans intern ausgiebi- ▶

Message Passing

Beim Message-Passing-Interface kommunizieren Sender und Empfänger über eine nachrichtenbasierte Schnittstelle. Diese Schnittstelle ist beim Message-Passing-Verfahren seit 1994 in Form einer API-Definition standardisiert. In einem MPI-Programm werden alle parallelen Konstrukte ausschließlich explizit durch den Programmierer in Form von Funktionsaufrufen codiert. Es existiert eine ganze Reihe von Implementierungen für diverse Programmiersprachen und Plattformen. Das MPI ist insofern bemerkenswert, als es 1980 beginnend aus einer ganzen Reihe ähnlicher Ansätze hervorging. Es verdrängte im Lauf der Zeit praktisch alle anderen nachrichtenbasierten Verfahren, die bisher für paralleles Rechnen entwickelt wurden.

Listing 1: IPerson

```
using System.Threading.Tasks;
using Orleans;

namespace Sample.Grains.Interfaces
{
  public interface IPerson : IGrainWithGuidKey
  {
    Task SetName(string name);
    Task<string> Name();

  }
}
```



Die Funktion *NuGet-Pakete wiederherstellen* fügt notwendige Abhängigkeiten hinzu (Bild 3)

gen Gebrauch von asynchronen Aufrufen. Für den mit *async/await* und Co. weniger vertrauten Entwickler empfiehlt sich hier mindestens ein kleiner Auffrischungs-Crashkurs, da die sichere Beherrschung beider Schlüsselwörter und der dahinterstehenden Konzepte für Orleans faktisch überlebenswichtig ist.

Erst das Interface, dann die Implementation

Orleans-typisch werden Interfaces und Implementation in getrennten Assemblies abgelegt. Für das Implementations-Assembly öffnen wir folglich wieder den *Hinzufügen*-Dialog in Visual Studio und wählen diesmal *Orleans Grain Class Collection* aus. Wiederum ändern wir die Vorgabe so ab, dass aus *Grain1* eine *Person* wird. Damit das auch klappt, müssen wir natürlich noch die Referenz auf das Interface-Assembly hinzufügen. Außerdem implementieren wir Getter und Setter gemäß Listing 2.

Auch hier gibt es wieder eine kleine Besonderheit zu beachten. Da der Setter ja eigentlich nur eine Eingabe entgegennimmt und selbst keinen Rückgabewert liefert, bräuchten wir eigentlich auch gar keinen solchen. Nun will Orleans aber partout eine Rückgabe vom Typ *Task* von uns haben. Als Erleichterung für diesen häufig benötigten Fall stellt uns Orleans mit der statischen Property *TaskDone.Done* eine intuitiv verwendbare Abkürzung bereit.

Wenn wir nun versuchen, das Projekt zu kompilieren, erhalten wir unter Umständen zunächst nur eine Fehlermeldung wie im Bild 3. Das ist an dieser Stelle zunächst völlig normal. Durch das Projekttemplate wurden zwar die *NuGet*-Abhängigkeiten definiert, die beim ersten Build theoretisch auch heruntergeladen und installiert werden sollten. Leider schlägt dieser Vorgang gelegentlich fehl; das Problem lässt sich aber durch Rechtsklick auf der Projektmappe, gefolgt von einem beherzten Klick auf *NuGet-Pakete wiederherstellen*, schnell lösen.

Listing 2: Person

```
using System;
using System.Threading.Tasks;
using Orleans;
using Sample.Grains.Interfaces;

namespace Sample.Grains.Implementation
{
  public class Person : Grain, IPerson
  {
    private string name;

    public Task<string> Name()
    {
      return Task.FromResult(name);
    }

    public Task SetName(string name)
    {
      this.name = name;
      return TaskDone.Done;
    }
  }
}
```

Rudimentärer Rahmen

Eine solche Grain ist für sich genommen zwar recht nett, aber um wirklich etwas Nützliches zu tun, muss sie auch aktiviert werden und ihre Methoden müssen ausgeführt werden. Neben der Möglichkeit, einfach eine weitere Grain zu schreiben, die dies tut, wird auch eine Art Hauptprogramm benötigt, das die Dinge insgesamt ins Rollen bringt.

Genau diesem Zweck dient die dritte Projektvorlage *Orleans Dev/Test Host*. Diese Vorlage erstellt uns einen rudimentären Rahmen, der gleich zwei Zwecken dient. Neben der Aufgabenstellung, die eigentliche Orleans-Anwendung über die dafür vorgesehenen Grains zu starten, stellt sie außerdem eine Orleans-Hostumgebung bereit, die mit der EXE zusammen gestartet wird.

await und UI

Innerhalb von Ereignis-Handlern, sei es Web oder Desktop-UI, ist es problematisch, blockierende Aufrufe zu verwenden, auch wenn es sich um *async*-Aufrufe handelt. Das Hauptproblem ist, daß der UI-SynchronizationContext somit unter Umständen innerhalb einer Closure eingefangen wird. Jeder Synchronisierungskontext kann aber nur durch einen Vorgang betreten werden. Wird der *async*-Code mit seiner Aufgabe fertig, blockieren sich sowohl die *async*-Continuation als auch der auf das *Task.Result* wartende Code gegenseitig, mit dem Ergebnis, daß die Applikation hängt. Gerade bei grafischen Oberflächen ist es aber sowieso essenziell, den Code nichtblockerend zu schreiben, um die Reaktionsfähigkeit des UI sicherzustellen. Hier empfiehlt sich der Einsatz von *Task.ConfigureAwait(false)*. Das sorgt dafür, daß die Referenz auf den Kontext abgegeben wird. Die *async*-Continuation läuft im Ergebnis im Kontext eines Threadpool-Threads weiter, womit letztlich der Deadlock effektiv vermieden wird.

Listing 3: Hostseitiger Programmcode

```
public class Program
{
  static void Main(string[] args)
  {
    // The Orleans silo environment is initialized
    // in its own app domain in order to more
    // closely emulate the distributed situation,
    // when the client and the server cannot
    // pass data via shared memory.
    AppDomain hostDomain =
    AppDomain.CreateDomain("OrleansHost", null,
    new AppDomainSetup
    {
      AppDomainInitializer = InitSilo,
      AppDomainInitializerArguments = args,
    });
    var config =
    ClientConfiguration.LocalhostSilo();
    GrainClient.Initialize(config);


    // TODO: once the previous call returns, the
    // silo is up and running.
    // This is the place your custom logic, for
    // example calling client logic
    // or initializing an HTTP front end for
    // accepting incoming requests.
    // eine Person
    var anton = GrainClient.GrainFactory.
    GetGrain<IPerson>(Guid.NewGuid());
    anton.SetName("Anton");
    Console.WriteLine("First person is "+
    anton.Name().Result);

    // und noch eine
    var berta = GrainClient.GrainFactory.
    GetGrain<IPerson>(Guid.NewGuid());
    berta.SetName("Berta").Wait();
    Console.WriteLine("Second person is " +
    berta.Name().Result);

    // wait & shutdown
    Console.WriteLine("Orleans Silo is running.\
    nPress Enter to terminate...");
    Console.ReadLine();
    hostDomain.DoCallBack(ShutdownSilo);
  }

  // ... more code ...
}
```

Diese letztere Konstruktion dient allerdings lediglich dazu, dem Entwickler das Leben einfacher zu machen. Für die Überführung in die Produktivumgebung wird dieser Code üblicherweise entfernt und die Orleans-Umgebung über die spätestens nach dem ersten Erstellvorgang im *BIN*-Verzeichnis befindliche, dedizierte *OrleansHost.exe* gestartet. Möchte man diesen für Debug-Zwecke im Code belassen, bietet sich daher der Einsatz bedingter Kompilierung an, um den Code für das automatisch generierte AppDomain-Konstrukt in der Produktiv-Version zu eliminieren.

Listing 4: Erweiterte Version von IPerson

```
using System.Threading.Tasks;
using Orleans;
using System;

namespace Sample.Grains.Interfaces
{
  public interface IPerson : IGrainWithIntegerKey
  {
    Task SetName(string name);
    Task<string> Name();

    Task SendGreetingTo(long personId);
    Task<string> Greet(string text, IPerson
    greeter);

    Task<int> CountFriends();
  }
}
```

Bevor wir den Code gemäß Listing 3 modifizieren können, fügen wir Referenzen auf beide anderen Projekte hinzu, damit beide Assemblies auch als Projektabhängigkeiten gebaut und in den Ausgabeordner verfrachtet werden. Das Host-Projekt sollte bei der Gelegenheit auch gleich als Startprojekt festgelegt werden.

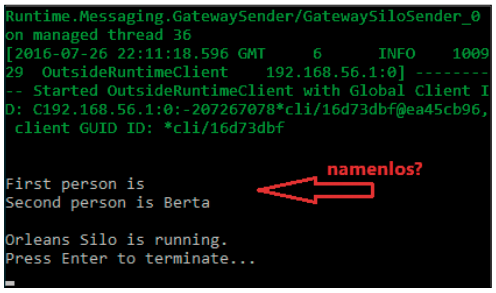
Wenn wir den Code nun ausführen, sehen wir zunächst ziemlich viele Log-Ausgaben, die uns aber nicht weiter interessieren. Der interessante Teil kommt ganz am Schluss. Hier erscheinen wie geplant die Namen unserer beiden Protagonisten – oder vielmehr fast. Denn tatsächlich erscheint nur ein Name. Was ist passiert? Ganz klar – wir haben beim Code für die erste Person *Anton* eine der Grundregeln von Orleans verletzt: Alles ist asynchron. Folglich hätten wir auf jeden einzelnen Aufruf warten müssen. Da wir das beim Setzen nicht getan haben, lesen wir im nächsten Aufruf sehr wahrscheinlich veraltete Daten aus, hier also den leeren Namen (Bild 4). Wie es richtig geht, zeigen die ansonsten funktional identischen Aufrufe für *Berta*.

Damit nun noch etwas mehr Interaktion in unser kleines Beispiel hineinkommt, sollen sich unsere beiden Protago- ▶

nisten gegenseitig begrüßen und dabei auch mit ihrem Namen ansprechen. Wir erweitern dazu den Code wie in Listing 4, Listing 5 und Listing 6 gezeigt.

Zum Interface *IPerson* kommen drei Methoden hinzu, die auch implementiert werden wollen, außerdem muss das Hostprogramm geringfügig erweitert werden, um den Grußvorgang anzustoßen (Bild 5). Damit wir *await* benutzen können und zudem den generierten Boilerplate-Code von unserem Nutzcode sauber trennen können, refaktorisieren wir bei dieser Gelegenheit Letzteren gleich in eine separate Methode, die im Beispiel einfach *RunMyGrains()* heißt.

Eine Möglichkeit, die Identität einer Grain zu ermitteln und weiterzugeben, ist ihr Primärschlüssel, der sich ohne *await* einfach und direkt von der Instanz abfragen lässt. Innerhalb *SendGreetingTo()* lässt sich die referenzierte Grain dann wie gehabt über die *GrainFactory* aktivieren. Vorausgesetzt natürlich, man weiß auch, welches Interface gefragt ist.



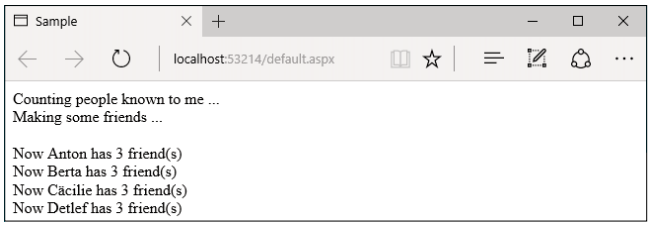
Fehler: Eine Person hat keinen Namen (Bild 4)

Da in Orleans sowieso fast alles asynchron abläuft, haben wir die beiden neuen Methoden mit *async* markiert, was uns außerdem erlaubt, das Schlüsselwort *await* zu verwenden. Und das sollten wir auch stets tun, wenn eine *Grain*-Methode aufgerufen wird.

Trotzdem würde es zu einem Deadlock kommen, sobald das Gegenüber in *Greet()* versucht, den Namen des diesmal per Zeiger übergebenen Aufrufers auszulesen. Warum ist das so? Der Grund wurde bereits weiter oben genannt. Orleans stellt ja sicher, dass der komplette Aufruf einer Methode einer bestimmten Instanz innerhalb eines einzigen Threads abläuft. Tatsächlich geht diese Garantie aber noch weiter, da Orleans auch dafür sorgt, dass von außen keine Methoden

Listing 5: Erweiterte Version von Person

```
using System;
using System.Threading.Tasks;
using Orleans;
using Sample.Grains.Interfaces;
using Orleans.Concurrency;
using System.Collections.Generic;

namespace Sample.Grains.Implementation
{
  [Reentrant]
  public class Person : Grain, IPerson
  {
    private string name;
    private int counter;
    private HashSet<string> friends = new
    HashSet<string>();

    public Task<string> Name()
    {
      ++counter;
      return Task.FromResult(name);
    }

    public Task SetName(string name)
    {
      this.name = name;
      return TaskDone.Done;
    }

    public async Task SendGreetingTo
    (long personID)
    {
      var person = GrainFactory.
      GetGrain<IPerson>(personID);
      var name = await person.Name();
      friends.Add(name);

      Console.WriteLine(string.Format
      ("Counter  before: {0}", counter));
      var antwort = await person.Greet("Hi " +
      name + " how are you?", this);
      Console.WriteLine(antwort);
      Console.WriteLine(string.Format("Counter
      after: {0}", counter));
    }

    public async Task<string> Greet(string text,
    IPerson greeter)
    {
      Console.WriteLine(text);

      // Callback, erfordert [Reentrant]
      var name = await greeter.Name();
      friends.Add(name);

      return "I'm fine " +name+", thank you.";
    }

    public Task<int> CountFriends()
    {
      return Task.FromResult(friends.Count);
    }
  }
}
```



Social Sample mit Grains in Aktion (Bild 5)

der Grain aufgerufen werden können, solange noch ausstehende Task-Objekte existieren. Dies schützt den internen Status der Grain zuverlässig vor unerwarteten Veränderungen und macht damit unterm Strich jegliches Locking im Grain-Code überflüssig.

Gelegentlich ergibt sich aber doch die Notwendigkeit, einen reentranten Aufruf zu erlauben, etwa für eine Callback-Funktion. Dazu stellt Orleans das Attribut *[Reentrant]* zur Verfügung. Damit signalisieren wir Orleans, dass wir um die mit Reentranz verbundenen Gefahren wissen und selbst entsprechende Vorsorge treffen werden.

Wie man in der Ausgabe des zweiten Beispiels auch sieht, wird die interne Statusvariable *counter* des Aufrufers durch den Aufruf von *greeter.Name()* in der *Greet()*-Methode sozusagen hinterrücks verändert. So ist die Ausgabe am Ende von *SendGreetingTo()* plötzlich eine andere als noch zu Beginn.

Nun haben es sich unsere beiden Protagonisten also auf der Orleans-Bühne bequem gemacht. Sie wissen, wie sie heißen und haben sich auch artig begrüßt. Wenn wir das Programm beenden und erneut starten, können wir auch unsere ▶

Listing 6: Erweiterte Version von Program

```
public class Program
{
  static void Main(string[] args)
  {
    // The Orleans silo environment is initialized
    // in its own app domain in order to more
    // closely emulate the distributed situation,
    // when the client and the server cannot
    // pass data via shared memory.
    AppDomain hostDomain =
    AppDomain.CreateDomain("OrleansHost", null,
    new AppDomainSetup
    {
      AppDomainInitializer = InitSilo,
      AppDomainInitializerArguments = args,
    });
    var config =
    ClientConfiguration.LocalhostSilo();
    GrainClient.Initialize(config);

    // TODO: once the previous call returns, the
    // silo is up and running.
    // This is the place your custom logic, for
    // example calling client logic
    // or initializing an HTTP front end for
    // accepting incoming requests.
    RunMyGrains();

    // wait & shutdown
    Console.WriteLine("\nOrleans Silo is running.\
    nPress Enter to terminate...\n");
    Console.ReadLine();
    hostDomain.DoCallBack(ShutdownSilo);
  }

  private async static void RunMyGrains()
  {
    // Cast (in alphabetical order)
    var anton = GrainClient.GrainFactory.
    GetGrain<IPerson>(123);
    var berta = GrainClient.GrainFactory.
    GetGrain<IPerson>(42);

    // was wissen unsere Kandidaten bereits?
    Console.WriteLine( "\nEntering stage:");
    Console.WriteLine("{0} has {1} friend(s)
    already", await anton.Name(), await
    anton.CountFriends());
    Console.WriteLine("{0} has {1} friend(s)
    already", await berta.Name(),
    await berta.CountFriends());
    Console.WriteLine();

    // den eignen Namen setzen, falls notwendig
    if (string.IsNullOrEmpty(await anton.Name()))
      await anton.SetName("Anton");
    if (string.IsNullOrEmpty(await berta.Name()))
      await berta.SetName("Berta");

    // Anton, sag Hallo zu Berta!
    await anton.SendGreetingTo
    (berta.GetPrimaryKeyLong());
    Console.WriteLine();

    // wie sieht's jetzt aus?
    Console.WriteLine("Now {0} has {1} friend(s)",
    await anton.Name(), await
    anton.CountFriends());
    Console.WriteLine("Now {0} has {1} friend(s)",
    await berta.Name(), await
    berta.CountFriends());
  }
  // ... more  code ...
}
```

Grains wieder aktivieren. Leider scheinen dann beide Mitwirkenden unter einer geheimnisvollen Amnesie zu leiden, denn sie haben nicht nur ihren Namen vergessen, sondern kennen auch ihre Freunde nicht mehr.

Das sollten wir beheben und zwar, indem wir ein wenig Persistenz einstreuen. Glücklicherweise bietet Orleans dazu ein Bordmittel namens deklarative Persistenz. Dieses Verfahren muss man nicht zwingend nutzen. Allerdings ist das Verfahren insofern vorteilhaft, als es sofort und ohne weitere Vorarbeiten auch mit Azure einsetzbar ist. Darüber hinaus ist der Persistenz-Layer von Orleans relativ offen. Es steht dem Entwickler frei, nach Bedarf auch eigene Storage-Provider zu bauen und diese plug-in-artig in das System zu integrieren.

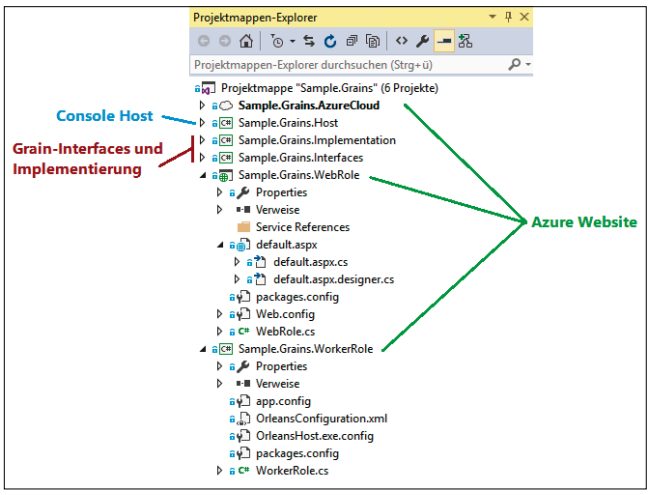
Per Default ist der *MemoryStorageProvider* im Code aktiviert, der aber leider nur für die Dauer des Prozesses seine Daten behält. Um etwas Dauerhafteres zu haben, können wir zum Testen den Microsoft Azure Storage Emulator auf der lokalen Maschine installieren. Zu finden ist dieser unter https://azure.microsoft.com/de-de/downloads, man muss auf der Seite allerdings wirklich bis ganz nach unten scrollen.

Um den Storage-Provider zu aktivieren, öffnen wir die Datei *OrleansHostWrapper.cs*. Dort suchen wir die Zeile

```
config.AddMemoryStorageProvider();
```

und ersetzen diese wie folgt:

```
config.AddAzureBlobStorageProvider(
  "Sample.Grains",
  "UseDevelopmentStorage=true"
);
```



Projektstruktur in Visual Studio (Bild 6)

Wichtig ist hierbei, dass wir uns den für den Storage vergebenen Namen, hier also *Sample.Grains*, gut merken, da wir ihn noch benötigen.

Als letzten Schritt ändern wir alle unsere Grains so ab, dass wie im Listing 7 alle zu persistierenden Daten in eine separate Klasse ausgelagert werden. Falls sich Visual Studio weigert, diesen Code zu akzeptieren, muss noch die Assembly *OrleansAzureUtils.dll* hinzugefügt werden.

Listing 7: Person mit Persistenz

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Orleans;
using Orleans.Concurrency;
using Orleans.Providers;
using Sample.Grains.Interfaces;

namespace Sample.Grains.Implementation
{
  public class PersonState
  {
    public string name { get; set; }
    public HashSet<string> friends { get; set; }
  }

  [Reentrant]
  [StorageProvider(ProviderName = "Sample.Grains")]
  public class Person : Grain<PersonState>, IPerson
  {
    private int counter;

    public override Task OnActivateAsync()
    {
      if (State.friends == null)
        State.friends = new HashSet<string>();
      return TaskDone.Done;
    }

    public Task<string> Name()
    {
      ++counter;
      return Task.FromResult(State.name);
    }

    public Task SetName(string name)
    {
      State.name = name;
      return base.WriteStateAsync();
    }


    // restlicher Code analog

  }
}
```

Damit eine Grain ihre Daten über die in Orleans eingebaute deklarative Persistenz speichert, muss die Grain angepasst werden. Zunächst werden alle speicherwürdigen Daten in eine Basisklasse verfrachtet und in öffentliche Properties umgewandelt.

Wichtig ist hier der Fakt, dass es sich tatsächlich um Properties handeln muss – einfache Felder reichen nicht aus. Danach ändern wir die Ableitung der Klasse so, dass sie anstatt von *Grain* jetzt von *Grain<Basisklasse>* erbt. Haben wir all dies erledigt, kann fortan auf die Daten der Klasse über *this.State* zugegriffen werden.

Listing 8: Zugriff auf SQL-Server

```
private async Task
WitdrawFromStock(List<PizzaIngredient> ingredients)
{
  var connect = new List<string>();
  connect.Add(@"Data Source=
  (localdb)\MSSQLLocalDB");
  connect.Add(@"Initial Catalog=Pizza");
  connect.Add(@"Integrated Security=True");
  var sConnect = string.Join(";", connect);
  using (var conn = new SqlConnection())
  {

    conn.Open();

    // ... more code ...

    using (var cmd = conn.CreateCommand())
    {
      cmd.CommandText = "INSERT INTO
      [IngredientsConsumption] "
      + "(id,ingredId,consumed,timestamp) "
      + "VALUES (@id,@ingred,@consumed,
      @timestamp)";
      cmd.Parameters.AddWithValue("id",
      Guid.NewGuid());
      cmd.Parameters.AddWithValue("ingred",
      entry.ingredId);
      cmd.Parameters.AddWithValue("consumed",
      consumed);
      cmd.Parameters.AddWithValue("timestamp",
      timestamp);

      var rows = await
      cmd.ExecuteNonQueryAsync();
      Debug.Assert(rows == 1);
    }

    // ... more code ...

  }
}
```

Um den aktuellen Zustand der Grain auch wirklich dauerhaft in das Storage-Medium zu persistieren, brauchen wir nur noch zwei Dinge. Das eine ist das Attribut an der Klasse, um Orleans zu signalisieren, dass und wohin die Klasse persistiert werden soll:

```
[StorageProvider(ProviderName = "Sample.Grains")]
```

Dass der Providername im Attribut derselbe ist wie der, den wir uns weiter oben hoffentlich sorgfältig notiert haben, ist also kein Zufall.

Die andere erforderliche Zutat sind Aufrufe der geerbten Methode *base.WriteStateAsync()*, die wiederum mit *await* zu erfolgen haben. Typischerweise ruft man *base.WriteStateAsync()* nach jedem abgeschlossenen Aufruf einer Methode auf, in der Daten manipuliert wurden.

Aufruf ist Pflicht

Der Aufruf ist dabei tatsächlich Pflicht, denn Orleans persistiert die Daten von sich aus erst einmal gar nicht, dies muss immer explizit angefordert werden. Begründet wird dieses Design zum einen damit, dass ein automatisches Speichern nach jedem Property-Setter die Performance unnötig strapazieren würde.

Zum anderen wäre das in vielen Fällen auch regelrecht kontraproduktiv, da die nicht unerhebliche Gefahr besteht, inkonsistente Daten zu speichern. Tritt während des Ablaufs der Methode eine Ausnahme auf, wird Orleans dies merken und den letzten gespeicherten Stand wiederherstellen. Aus naheliegenden Gründen sollten persistierte Zustandsdaten einer Grain also konsistent sein

Selbstverständlich kann man neben den Properties in *this.State* auch noch lokale Member in der Grain selbst haben, wie etwa das Feld *counter* im Listing. Allerdings wird dieses Feld dann natürlich auch nicht persistiert werden.

Still und heimlich haben wir auch noch eine weitere Methode in die Klassendeklaration geschmuggelt. *OnActivateAsync()* ist ein Event Handler, der während der Aktivierung der Grain aufgerufen wird. Analog gibt es auch noch die Möglichkeit, über eine unmittelbar bevorstehende Deaktivierung benachrichtigt zu werden; diese Methode heißt entsprechend *OnDeactivateAsync()*.

Diese beiden Methoden sind der richtige Platz für den Fall, dass doch das eine oder andere initialisiert oder deinitialisiert werden muss. Code, den man im Normalfall in einen Konstruktor packen würde, ist hier gut aufgehoben. Außerdem schreiben wir in unserem Beispiel die ganze Zeit direkt auf die Konsole, was aber streng genommen verpönt ist – hierfür ist der über *GetLogger()* abrufbare Logger gedacht.

Externe Datenquellen

In der Praxis kommt es gelegentlich vor, dass auch auf andere Datenquellen zugegriffen werden muss. Aus einer Grain heraus ist dies ganz normal möglich. Falls vorhanden, sollte man hierzu asynchrone Aufrufe nutzen, die sich optimal in Orleans einfügen, ohne den Ausführungs-Thread der Grain übermäßig zu blockieren. ▶

Wie man in Listing 8 sieht, unterscheidet sich der Zugriff nicht wesentlich von einem normalen SQL-Server-Zugriff. Als Schlussfolgerung nehmen wir also mit, dass die Einbindung externer Datenquellen umso besser funktioniert, je besser der Support für asynchrone Aufrufe ist. Das gilt aber nicht nur für Datenquellen, sondern ganz allgemein für jeden ausgehenden Aufruf, inklusive entfernter Prozeduraufrufe (RPC) oder etwa Webzugriffe, beispielsweise um Daten über ein REST-API oder eine Socketverbindung auszutauschen.

Gerät man in die Versuchung, lang laufende Operationen zu programmieren, so ist auch dies durchaus möglich. Hierzu sollte man allerdings nicht einfach nach Lust und Laune einen Thread starten, sondern stets den Weg über den Orleans-eigenen Scheduler nehmen. Da sich dieser beim Start quasi in das Framework einklinkt, kann dies recht einfach über await *Task.Factory.StartNew()* bewerkstelligt werden. Andere Verfahren würden den Scheduler umgehen, was im Kontext eines System wie Orleans eher weniger wünschenswert ist. Schließlich ist man ja an einer möglichst optimalen Gesamtperformance interessiert.

Nicht immer allerdings bedarf es tatsächlich eines Threads. Geht es nur darum, regelmäßig eine Aktion anzustoßen, sollte man stattdessen besser zu *RegisterTimer()* oder *RegisterOrUpdateReminder()* greifen. Erstere Funktion registriert einen Timer, der eher für kurzzeitige Intervalle ideal ist. Zu einem Reminder greift man, wenn man längerdauernde Intervalle benötigt. Reminder haben außerdem die Eigenschaft, dass sie persistent sind. Während ein Timer mit Deaktivierung der Grain oder einem Crash des Silos verschwindet, wird die Grain beim Ablauf des Reminder-Intervalls bei Bedarf reaktiviert. Dies gilt auch dann, wenn der Silo zwischenzeitlich neu gestartet wurde.

Listing 9: ASP-Website

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="default.aspx.cs" Inherits="Sample.
Grains.WebRole._default" Async="true" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <meta http-equiv="Content-Type"
    content="text/html; charset=utf-8"/>
    <title>Sample</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <asp:UpdatePanel ID="UpdatePanel1"
      runat="server">
        <ContentTemplate>
          <div>
            <asp:ScriptManager ID="ScriptManager1"
            runat="server"  />
            <asp:Label ID="GrainOutput"
            runat="server"> Loading ...
            </asp:Label>
            <asp:Timer ID="Reload" runat="server"
              Interval="1000" OnTick="Reload_Tick">
            </asp:Timer>
          </div>
        </ContentTemplate>
      </asp:UpdatePanel>
    </form>
  </body>
</html>
```

Wolkiges

Ein typischer Einsatzfall einer Orleans-basierten Applikation ist zweifellos eine Webumgebung. Folgerichtig unterstützt Orleans von Haus aus auch Windows Azure, allerdings ist das Setup einer komplett neuen Azure-Webanwendung durchaus als holprig zu bezeichnen (Bild 6). Theoretisch sollte es eigentlich reichen, wenn man die folgenden Pakete über die NuGet-Konsole in die jeweiligen Projekte *WebWorker* (Listing 9) und *WebRole* (Listing 10) installiert:

- Microsoft.Orleans.Client (WebRole),
- Microsoft.Orleans.Server (WebWorker),
- Microsoft.Orleans.OrleansAzureUtils (beide),
- Microsoft.Orleans.OrleansProviders (beide).

Falls etwas nicht wie erwartet funktioniert, empfiehlt sich zunächst ein Blick auf die Assembly-Referenzen, die nicht immer ganz aktuell sind. Hier muss man unter Umständen die *packages.config* aller Projekte auch einmal händisch nachbearbeiten. Häufigste Problemquellen sind veraltete Referenzen auf *Newtonsoft.Json* beziehungsweise inkonsistente Versionen der Orleans-Assemblies selbst. Letzteres ist mit ziemlicher Sicherheit der Fall, wenn man beim Start merkwürdige Fehlermeldungen à la *ReflectionTypeLoadException* erhält.

Hier empfiehlt es sich, alle Referenzen auf Orleans-Assemblies zu überprüfen, ob wirklich überall dieselbe Versionsnummer angegeben wurde. Der andere mögliche Grund ist natürlich recht banal: Das im Log gemeldete Assembly fehlt im Ausgabeordner. Hier geht der prüfende Blick zunächst in Richtung *Microsoft.Orleans.OrleansProviders*, denn dieses Assembly wird für die Serialisierung benötigt.

Ein anderer möglicher Grund für Fehler ist das Fehlen einer Konfigurationsdatei. In den meisten Fällen ist es eine fehlende *OrleansConfiguration.xml*, die bemängelt wird. Selbige kann man sich aber ganz einfach aus unserem ersten Projekt kopieren und entsprechend anpassen. Die Einträge der XML-Datei sind selbsterklärend, sodass wir hier aus Platzgründen nicht weiter darauf eingehen.

Etwas tricky kann auch die Implementierung des ASP-Code-Behinds sein (Listing 11), besonders da wir ja ständig *async* und *await* verwenden. Die mit Abstand empfehlenswerteste und darüber hinaus auch eleganteste Lösung besteht darin, alles bis herunter zum Event Handler *async* zu machen – eben »async all the way down«, um einen bekannten Merksatz zu zitieren.

Gern vergessen wird der Umstand, dass man auch die ASP-Seite selbst über die *@Page*-Direktive als *async="true"* mar-

Listing 10: Code Behind der Azure WebRole

```
namespace Sample.Grains.WebRole
{
  public partial class _default : System.Web.UI.Page
  {

    protected void Page_Load(object sender,
    EventArgs e)
    {
      if ( Page.IsPostBack)
      {
        EnsureInitialized();
      }
    }

    private void EnsureInitialized()
    {
      if (!AzureClient.IsInitialized)
      {
        FileInfo file = AzureConfigUtils.
        ClientConfigFileLocation;
        if (file.Exists)
          AzureClient.Initialize(file);
        else
          throw new FileNotFoundException
          ("config file missing"),
          file.FullName);
      }
    }


    static int friends = 0;

    protected async void Reload_Tick(object
    sender, EventArgs e)
    {
      if (++friends >= NAMES.Length)
      {
        Reload.Enabled = false;
        friends = NAMES.Length - 1;
      }
      GrainOutput.Text = await
      RunMyGrains(friends);
    }

    public static readonly string[] NAMES = {
    "Anton", "Berta", "Cäcilie", "Detlef",
    "Erwin", "Frank", "Gustel", "Harry", "Isolde",
    "Klaus" };

    private async Task<string> RunMyGrains
    (int friends)
    {
      if (friends < 1)
        throw new ArgumentException("friends
        must be > 0");

      var lines = new List<string>();
      try
      {
        // Cast (in alphabetical order)
        var people = new List<IPerson>();
        while (people.Count <= friends)
          people.Add( GrainClient.
          GrainFactory.GetGrain<IPerson>
          (people.Count+1));

        // den eignen Namen setzen
        lines.Add("Counting people known
        to me ...");
        foreach (var cast in people)
          if (string.IsNullOrEmpty(await
          cast.Name()))
            await cast.SetName(NAMES[cast.
            GetPrimaryKeyLong() - 1]);

        // Anton, sag‘ Hallo zu Berta!
        lines.Add("Making some friends ...");
        for (var i = 0; i < people.Count; ++i)
          for (var k = i + 1; k <
          people.Count; ++k)
            await people[i].SendGreetingTo
            (people[k].GetPrimaryKeyLong());
        lines.Add("");

        // wie sieht’s jetzt aus?
        foreach (var cast in people)
          lines.Add(
            string.Format(
              "Now {0} has {1}
              friend(s)",
              await cast.Name(),
              await
              cast.CountFriends()));
      }
      catch (Exception e)
      {
        lines.Add("ERROR: " + e.Message);
      }

      // render output
      for( var i = 0; i < lines.Count; ++i)
        lines[i] =
        HttpUtility.HtmlEncode(lines[i]);
      return string.Join("<br>", lines);
    }

  }
}
```

kieren muss, damit alles funktioniert. Tut man all dies nicht und lässt stattdessen den Aufruf einer *asnyc*-Routine innerhalb eines Event Handlers blockieren, kann man getrost darauf Wetten abschließen, wie lange die Konstruktion wohl ohne Deadlock funktionieren wird.

Entsprechend der Aufgabenstellung eignet sich Orleans wie alle parallelen Systeme vor allem für Aufgabenstellungen, bei denen gut partitionierbare Aufgaben vorliegen, die sich auf lose gekoppelte und unabhängig agierende Einheiten verteilen lassen. Idealerweise ist die Ausführung des Grain-Codes schon naturgemäß single-threaded. Obwohl Orleans auch die Erzeugung von Threads durch Grains unterstützt, entspricht dies nicht dem Ideal. Lang laufende Aufgaben, die wenige schwergewichtige Grains involvieren würden, eignen sich eher weniger für Orleans.

Listing 11: Azure WorkerRole

```
namespace Sample.Grains.WorkerRole
{
  public class WorkerRole : RoleEntryPoint
  {
    private AzureSilo orleansAzureSilo;

    public override void Run()
    {
       var config = new ClusterConfiguration();
       config.StandardLoad();

       // It is IMPORTANT to start the silo not
       // in OnStart but in Run.
       // Azure may not have the firewalls open
       // yet at the OnStart phase.
       orleansAzureSilo = new AzureSilo();
       bool ok = orleansAzureSilo.Start(config);

       Trace.WriteLine
       ("OrleansAzureSilos-OnStart Orleans silo
       started ok=" + ok, "Information");
       Debug.Assert(ok);

       orleansAzureSilo.Run();
       // Call will block until silo is shutdown
    }

    public override bool OnStart()
    {
      ServicePointManager.
      DefaultConnectionLimit = 12;
      return base.OnStart();
    }

    public override void OnStop()
    {
      if (orleansAzureSilo != null)
        orleansAzureSilo.Stop();

      base.OnStop();
    }

  }
}
```

Links zum Thema

- Microsoft Orleans Tools for Visual Studio
  https://visualstudiogallery.msdn.microsoft.com/36903961-63bd-4eec-9ca4-cf2319dc75f4
- Überblick über alle NuGet-Packages und deren Einsatzzweck
  http://dotnet.github.io/orleans/NuGets
- Azure-Speicheremulator
  https://azure.microsoft.com/de-de/downloads
- Stephen Toub »Await, and UI, and deadlocks! Oh my!«
  https://blogs.msdn.microsoft.com/pfxteam/2011/01/13/await-and-ui-and-deadlocks-oh-my

Fazit

Beachtet man ein paar Grundregeln, wie etwa den durchgängigen, konsequenten Einsatz von *async/await,* und achtet man auf eine gute Partitionierbarkeit der zu erledigenden Aufgaben, lassen sich schon nach kurzer Einarbeitungszeit mit Orleans produktiv Lösungen programmieren, die quasi von Natur aus schon recht gut skalieren sollten. Wie mit jeder Lösung, bei der Asynchronizität und Nebenläufigkeit eine Rolle spielen, ist auch für Orleans ein gutes Verständnis für Abläufe dieser Art durchaus hilfreich.

Die durchdachte Programmierschnittstelle und die exzellente Unterstützung durch die Visual-Studio-Templates machen es recht einfach, schnell erste Ergebnisse zu erzielen. Aber auch mit fortschreitender Entwicklung der Anwendung macht Orleans eine gute Figur, wenn man sich auf die dahinterstehenden Konzepte einlässt. Einzig das gelegentliche Versionschaos bei den nicht immer ganz konsistenten Assembly-Verweisen und die damit verbundenen Fehlermeldungen sorgen gelegentlich dafür, dass die Cloud-Lösung in eine Gewitterwolke mutiert. Hat man diese Hürden aber gemeistert, läuft die Lösung in der Tat erstaunlich stabil. ■

Jens Geyer
entwickelt bei der VSX Vogel Software GmbH Lösungen. Sein Fokus liegt auf der Entwicklung paralleler und verteilter Anwendungen. Er ist Committer/PMC-Member des plattformübergreifenden RPC-Frameworks Apache Thrift.
http://jensgeyer.net

Developer Newsletter

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

Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile

LAST- UND PERFORMANCE-TESTS MIT JMETER UND GATLING

Performance-Check

Performance wird bei Anwendungen eine immer kritischere Anforderung.

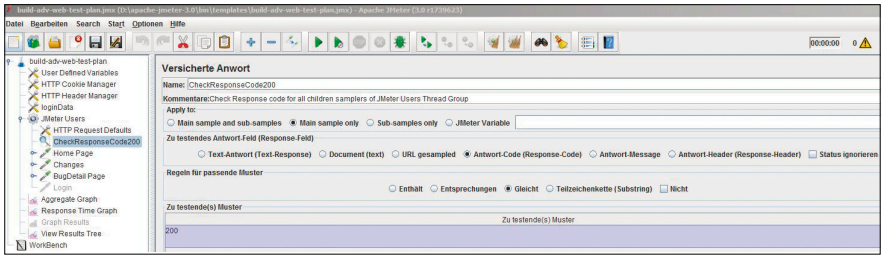
Es ist besser, wenn man recht früh die Grenzen seines Systems kennt, um geeignete Gegenmaßnahmen einzuleiten. Performance-Tests gehören oft zur leidigen Last für Projekte. Besser als aufwendige Ad-hoc-Aktionen ist hier die Einbindung in den kontinuierlichen Integrationstest und die Verwendung von Cloud-Ressourcen. Wir stellen hier zwei der beliebtesten Open-Source-Lasttest-Werkzeuge, den Platzhirsch JMeter und dessen Herausforderer Gatling, vor. Mit Taurus lassen sich die Stärken beider Werkzeuge vebinden, sodass man sogar beide zusammen einsetzen kann.

Das Testen von Anwendungen gehört nicht zu den Lieblingstätigkeiten von Entwicklern. In großen Firmen sind dafür oft eigene Abteilungen mit teuren Spezialwerkzeugen zuständig. Nach dem agilen Selbstverständnis muss das Entwicklerteam selbst die Verantwortung dafür übernehmen, dass seine Anwendung die Qualitätsansprüche der Praxis erfüllt. Zum Glück lassen sich die beliebtesten Open-Source-Lasttestwerkzeuge, JMeter und Gatling, gut in den kontinuierlichen Erstellungsprozess integrieren. So dürfte es weniger unliebsame Überraschungen in der Produktion geben.

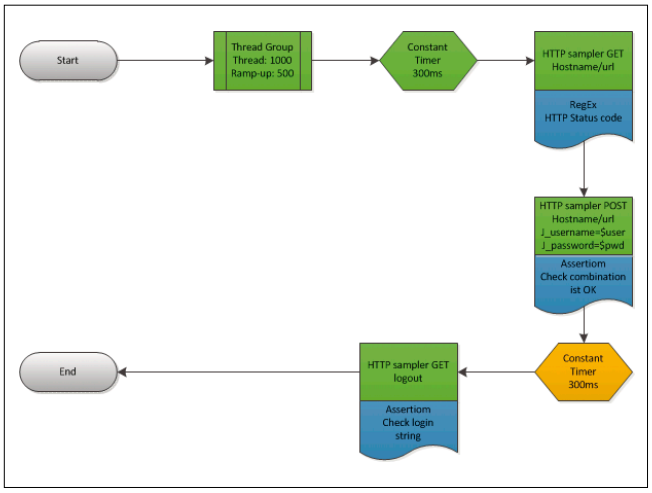
Apache JMeter – das Arbeitspferd

Die Entwickler des Apache Webservers Tomcat standen vor der Herausforderung, zu überprüfen, ob ihr Produkt skalierbar und robust genug war. So entwickelte das Tomcat-Team neben dem Build-Werkzeug Ant auch das Lasttestwerkzeug Apache JMeter. Beide wurden später eigenständige Apache-Projekte, die bis heute weiterentwickelt werden.

Ein wenig merkt man der Swing-Oberfläche von JMeter das Alter von 18 Jahren an. Doch letztendlich ist das Ganze nur ein Aufsatz, um damit eine XML-Datei zu erstellen, die einen Testplan enthält. Aufgerufen wird das Ganze später meist ohnehin auf der Kommandozeile. Sie ist im Lasttest einmalig oder im Rahmen des Maven-Builds als Jenkins-Job kontinuierlich durchzuführen. Wenn Sie hinter einem Firmen-Proxy sitzen, müssen Sie diesen vor dem Starten von JMeter als Parameter angeben:



Das Tomcat-Team entwickelte auch das Lasttest-Werkzeug Apache JMeter (Bild 1)



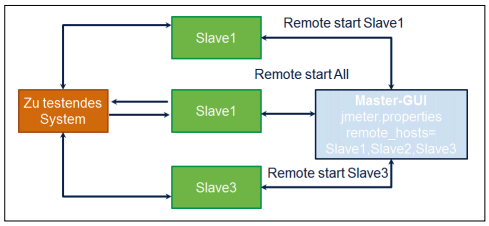
Hauptschritte zu einem JMeter-Testplan (Bild 2)

```
jmeter -H my. proxy.server -P 8000 -N localhost
```

Wenn Sie JMeter ohne Oberfläche starten wollen, um die Ergebnisse in einer separaten Datei abzulegen, so fügen Sie noch die Parameter *-n -t my_test.jmx -l log.jtl* hinzu. *jmeter -?* gibt die weiteren möglichen Startparameter auf der Kommandozeile aus (Bild 1).

Es gibt mehrere Möglichkeiten, einen Testplan zu erstellen. Entweder Sie passen einen bereits existierenden Testplan an Ihre Anforderungen an, oder Sie erstellen einen eigenen von Grund auf neu. Das Grundgerüst für einen eigenen Testplan kann man sich auch mit dem Test-Recorder erstellen lassen, indem man JMeter als Proxy für den Webbrowser konfiguriert und die angeklickten Testpfade in der Webanwendung aufzeichnet. Später wird dieser Testplan dann noch um Zeitmesser für Warte- und Denkzeiten, Parameter für verschiedene Szenarien und Umgebungen angepasst. Hinzufügen sollte man außerdem Zusicherungen zur Überprüfung des Abfrageergebnisses und Reports, um die Test-Ergebnisse zusammenzufassen und darzustellen.

Wichtig sind hier die Elemente HTTP Cookie Manager für die Cookie-Verwaltung, die HTTP Request Defaults, die für alle Request identisch sind, wie Host und Portnummer. Außerdem sollten Sie die Benutzervariablen in ein eigenes Element auslagern, um so den Testplan zum Beispiel für verschiedene Umgebungen zu verwenden oder später mehr Testdaten aus einer CSV in



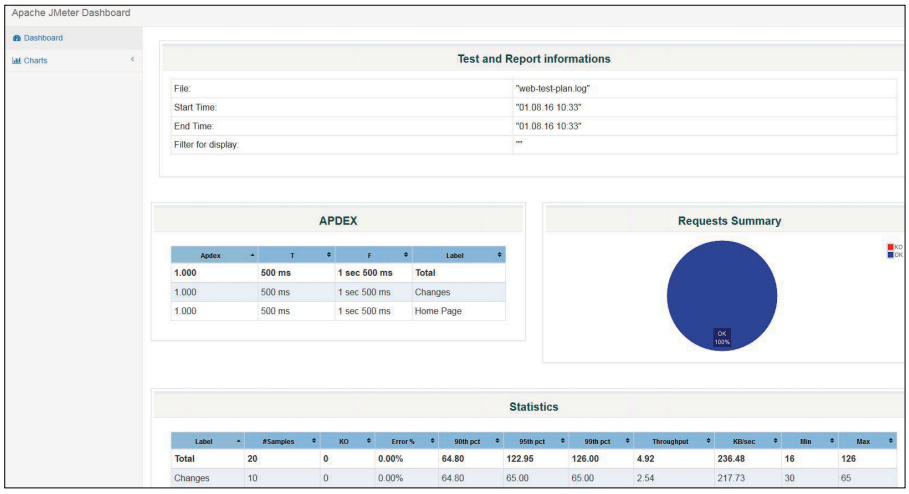
Das Master-Slave-Diagramm von JMeter (Bild 3)

diese Variablen einzulesen. Dazu platzieren Sie die CSV Data Set Config in der Thread Group.

In deren Konfiguration geben Sie neben dem Dateinamen auch die Spaltennamen der zu füllenden Variablen an. Jeder Thread übernimmt dann aus dieser Datei eine Datenzeile. Wenn Sie die Benutzervariablen vor dem Test-Recorder (unter den Nicht-Testelementen beim WorkBench-Eintrag) anlegen, kann der HTTP-Recorder, wenn Sie die Thread Group als Ziel für die HTTP-Request als Ziel-Controller angeben, bereits die Werte durch die Variablen ersetzen. Ansonsten müssen Sie die Werte im Nachgang manuell durch Variablen ersetzen. Später können Sie den Testplan noch um eine Zeitsteuerung und einen Logik-Controller erweitern.

Listener sind zum Testen sinnvoll, sollten aber später für den echten Test deaktiviert werden, da sie die Gesamtperformance beeinflussen und so die Ergebnisse verfälschen können. Hier sind vor allem die Results-Tree-View, der Response-Time-Graph, der Ergebnis-Graph und der Summary-Report hilfreich. Möchten Sie später die Ergebnisse entweder mit einem Berichtswerkzeug oder mit Excel separat auswerten, so verwenden Sie dazu den Simple Data Writer. Um sicherzugehen, dass die Requests erfolgreich durchlaufen wurden, sollten Sie die Response Assertion verwenden, um zum Beispiel den HTTP-Code 200 oder ein erwartetes HTML-Element abzufragen. Wenn die Bedingung nicht erfüllt ist, soll dieser Testdurchlauf abgebrochen werden.

Alternativ können Sie auch die Requests mit dem Entwicklerwerkzeug der Browser im HAR-Format (HTTP Archive) abspeichern und mit dem Groovy HAR-Archiv zu JMX-Testplan Konverter umwandeln.



Report: Ein mit JMeter generierter Report (Bild 4)

Das zur Dokumentation von REST-Services immer beliebtere Framework Swagger kann aus seiner Schnittstellenbeschreibung im JSON oder YAML-Format einen JMeter-Testplan generieren. Dazu lädt man die Schnittstellenbeschreibung in den http://editor.swagger.io/#/ hoch und generiert den JMeter-Client dafür (Bild 2).

Um möglichst realistische Tests zu erstellen, reicht ein Rechner zum Ausführen großer Lasttests nicht aus, da sonst die Netzwerkkarte oder das eigene Netzwerk zum begrenzenden Faktor für den Test wird. Manchmal kann es sogar sinnvoll sein, zum Beispiel bei mobilen Anwendungen bewusst mit einer herunterregulierten Netzwerkkarte oder über WLAN den Test durchzuführen, um die Fehlerquote dafür zu berechnen, wie Verbindungsabbrüche sich auf das Ergebnis auswirken.

JMeter-Instanzen starten

Hierzu starten Sie verschiedene JMeter-Instanzen, wobei diese über die *jmeter.properties*-Datei jeweils als *remote_hosts* eingetragen sind. So können die Tests auf den Slave-Clients *(jmeter-server.bat)* von JMeter zentral vom JMeter Master *(jmeter.bat Remote Start All remote_hosts)* gestartet werden und deren Ergebnisse auch dort verarbeitet werden. Alternativ kann man auch die Testergebnisse der Clients lokal speichern und im Nachgang zu einer Datei zusammenführen. Das empfiehlt sich, wenn man zum Beispiel die Tests in der Cloud mit Amazon-EC2-Instanzen durchführt, da der für die Master-Slave-Konfiguration benötigte RMI-Port 1099 dort nicht erreichbar ist (Bild 3).

Da die von JMeter in der Swing-Oberfläche mitgelieferten Diagramme doch zu einfach und zu wenig übersichtlich sind, bietet es sich an, einen aussagekräftigen Gesamtbericht mit unterschiedlichen Berichten und Grafiken im Nachgang zu erstellen. Dazu rufen Sie bei JMeter 3.0 das Skript

```
jmeter -n -t <test JMX file> -l <test log file> -e -o
<Path to output folder>
```

auf. Es erstellt ein dynamisches Report Dashboard mit einer APDEX-Tabelle (Application Performance Index). Alternativ können Sie auch einen PDF-Report über https://sense.blazemeter.com online aus einer Request-Datei erstellen (Bild 4).

Auch wenn alle Informationen zum JMeter-Testplan in der JMX-Datei vorliegen, ist XML nicht jedermanns Sache. Wer sich den Plan auch außerhalb des JMeter-Clients anschauen möchte, kann mit

```
apache-jmeter\extras\schematic.cmd
test-plan.jmx
```

den Testplan im HTML-Format erstellen lassen. Wer nur einen kurzen Überblick über den Testplan ha- ▶

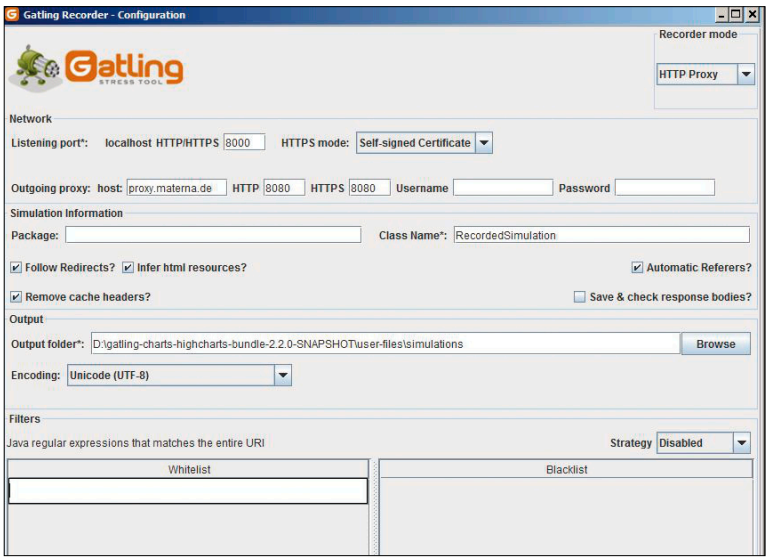
ben möchte, ruft dazu das eigentlich zum Verifizieren der Korrektheit der JMX-Datei vorgesehene Werkzeug mit *TestPlanCheck.bat --jmx test-plan.jmx --stats --tree-dump* auf.

Apache JMeter hat seinen Ursprung im Testen von Webanwendungen. Dadurch, dass auch andere Protokolle wie JDBC, FTP oder JMS unterstützt werden, kann das Werkzeug auch für andere Anwendungsarten eingesetzt werden.

Wie man sieht, bietet JMeter eine Vielzahl von Erweiterungen und wird kontinuierlich weiterentwickelt. Als bewährtes Arbeitspferd verrichtet JMeter immer noch gut seine Arbeit.

Gatling – der Herausforderer

Gatling wurde vor fünf Jahren als Testwerkzeug für Webanwendungen mit einer eigenen Domain-Specific Language (DSL) für die Testszenarien entwickelt. Die technische Basis dafür sind Scala, Akka und Netty, wie sie auch für moderne reaktive Anwendungen verwendet werden. Dadurch arbeitet Gatling asynchron und nichtblockierend. Als Ablaufumgebung benötigt Gatling ein JDK 8. Intern wird in der aktuellen Version 2.2.2 die funktionale Sprache Scala 2.11 verwendet. Sollten Sie noch Testpläne für die Einser-Version von Gatling haben, müssen diese neu übersetzt oder angepasst werden. Wegen der Verwendung von Scala ist Gatling eher etwas für Programmierer. Die Oberfläche und die dynamischen Berichte machen einen modernen Eindruck (Bild 5).



Recorder: Der Gatling GUI-Test-Recorder (Bild 5)

Zum Einstieg ist es sicher einfacher, einen der mitgelieferten Berichte an die eigenen Bedürfnisse anzupassen, als sich mit dem mitgelieferten Test-Recorder einen Bericht erstellen zu lassen (Bild 6). Die dabei erzeugte Datei mit der Testsimulation zeigt Listing 1. Dieser Testplan erinnert etwas an JUnit-Tests und lässt sich einfach, auch ohne Scala-Kenntnisse, anpassen. Wenn man jedoch eine Datei von Grund auf erstellen oder an spezielle Anforderungen anpassen möchte, ist man auf Scala-Programmierkenntnisse angewiesen. Beispiele für Testpläne zum Einsatz in fortgeschrittenen Szenarien finden Sie unter *simulations\computerdatabase\advanced*. Der Testplan wird mit

```
gatling.bat -s  simulation <className>
```

aufgerufen oder man wählt einen der Testpläne im Verzeichnis *user-files\simulations* aus. Dabei ist zu beachten, dass die Namen der Testpläne eindeutig sein müssen, da diese mit dem Java-Compiler ins Verzeichnis *Record target\test-classes* übersetzt werden.

Das interaktive Berichtsergebnis liegt unter *results\recordedsimulation-XYZ\index.html* und kann mit jedem normalen Browser geöffnet werden. Weitere Optionen von Gatling können mit *gatling.bat –h* angezeigt werden. Die Standardwerte für Gatling können in der Konfigurationsdatei *gatling.*

Listing 1: Mit dem Recorder erstellte Testklasse

```
class RecordedSimulation extends Simulation {

  val httpProtocol = http
    .baseURL("http://www.materna.de")
    .proxy(Proxy("proxy.materna.de",
     8080).httpsPort(8080))
    .inferHtmlResources()
    .acceptHeader("text/html,application/
     xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
    .acceptEncodingHeader("gzip, deflate")
    .acceptLanguageHeader("de,en-US;q=0.7,en;q=0.3")
    .userAgentHeader("Mozilla/5.0 (Windows NT 6.1;
    WOW64; rv:45.0) Gecko/20100101 Firefox/45.0")

  val uri1 = "www.materna.de"

  val uri2 = "intranet.materna.de"

  val scn = scenario("RecordedSimulation")
    .exec(http("request_0")
      .get("/")
      .check(status.is(403)))
    .pause(19)
    .exec(http("request_1")
      .get("http://" + uri2 +
       "/news/Seiten/2016-06-30-insight.aspx")
      .check(status.is(403)))

  setUp(scn.inject(atOnceUsers(1))).
protocols(httpProtocol)
```

conf oder für den Recorder in der Datei *recorder.conf* angepasst werden. Für größere Tests ist es praktischer, die Werte aus einer Datei auszulesen. Dazu bietet Gatling Leser für unterschiedliche Dateiformate an, wie:

```
val csvFeeder = csv("foo.csv")
val tsvFeeder = tsv("foo.tsv")
val ssvFeeder = ssv("foo.ssv")
val customSeparatorFeeder =
separatedValues("foo.txt", '#')
val jsonFileFeeder = jsonFile("foo.json")
jdbcFeeder("databaseUrl", "username",
"password", "SELECT * FROM users")
val feeder = sitemap("/path/to/sitemap/file")
```

Report: So sieht ein Gatling-Report aus (Bild 6)

Gatling ist gerade für reaktive Anwendungen, die oft in Scala entwickelt werden, oder für Webanwendungen gut geeignet. Es lässt sich sehr gut in die Entwicklungsumgebung und den Prozess integrieren. Dazu gibt es sowohl für Maven als auch Jenkins ein Plug-in. Mit einer Vorgängerversion von Gatling erstellte Testpläne können mit einer neueren Version nicht ausgeführt werden, da die Scala-Versionen nicht abwärtskompatibel sind.

Ein direkter Vergleich zwischen JMeter und Gatling ist schwierig. Wenn man auch mit beiden gute Ergebnisse erzielen kann, so können spezielle Anforderungen, zum Beispiel an Protokolle, oft ein Ausschlusskriterium sein. JMeter glänzt schon allein durch seinen großen Funktionsumfang und die vielen Erweiterungen und Dokumentationen. Dafür ist Gatling moderner, vor allem wenn man sich mit Scala auskennt und die Möglichkeit nutzt, per Programmiersprache direkt einen Testplan zu erstellen oder zu erweitern.

Manchmal muss man sich gar nicht zwischen beiden Werkzeugen entscheiden. Mit dem Werkzeug Taurus von Blazemeter ist es sogar möglich, beide Formate auszuführen. Das ist besonders für Firmen interessant, die über eine zentrale Testabteilung verfügen.

Fazit

Last und Lust im Hinblick auf Performance-Tests halten sich mit Apache JMeter und Gatling die Waage. Je unternehmenskritischer die Systeme werden, desto wichtiger sind aussagefähige Last- und Performance-Tests, die in Continuous Integration eingebunden sind. Beide Werkzeuge haben den Praxistest bestanden. Für welches Sie sich entscheiden, bleibt den persönlichen Vorlieben vorenthalten. Sie sollten jetzt weniger Ausreden haben und motivierter sein, solche Tests zukünftig häufiger durchzuführen. ■

Links zum Thema

- Neuerungen in Apache JMeter 3.0
  http://jmeter.apache.org/changes.html
- Gatling
  http://gatling.io
  http://gatling.io/#/cheat-sheet/2.2.2
  http://gatling.io/docs/2.2.2
- JMeter-Plug-ins
  http://jmeter-plugins.org
- JMeter-Erweiterungen
  http://sourceforge.net/projects/jmeterenhancements
- Refcard #228 Getting Started With Apache JMeter
  https://dzone.com/refcardz/getting-started-with-apache-jmeter
- Taurus
  https://blazemeter.com/taurus
- Online-Report
  https://loadosophia.org
- JMeter-Master-Slave-Test
  http://jmeter.apache.org/usermanual/remote-test.html
- JMeter-AWS-EC2-Skript
  https://github.com/oliverlloyd/jmeter-ec2
- HTTP-Archive-Format (HAR)
  https://confluence.atlassian.com/kb/generating-har-files-and-analysing-web-requests-720420612.html
  https://en.wikipedia.org/wiki/.har
- Groovy HAR-Archiv zu JMX-Testplan Konverter
  http://seitenbau.github.io/har2JMeter

Frank Pientka
ist Principal Software Architekt bei der Materna GmbH in Dortmund. Er beschäftigt sich seit mehreren Jahrzehnten mit Java EE und Open-Source-Software.
redaktion@webundmobile.de

FORMULAR-PLUG-IN FORMIDABLE FÜR WORDPRESS

Datensammler

Formulare sind nicht gerade der spannendste Teil einer Website, aber für viele Anwendungen notwendig.

Mit dem WordPress-Plug-in Formidable bauen Sie mit geringem Aufwand ansprechende und ausgefeilte Formulare für Ihre Website (Bild 1).

Formulare eignen sich für eine Vielzahl von Anwendungen: User-Befragungen, Bestellungen, Gewinnspiele, Kleinanzeigen und noch viel mehr. Das Plug-in Formidable für WordPress macht die Umsetzung solcher Aufgaben zur einfachen Übung: Es nimmt Ihnen viele der Mühen ab und glänzt durch seine vielfältigen Möglichkeiten.

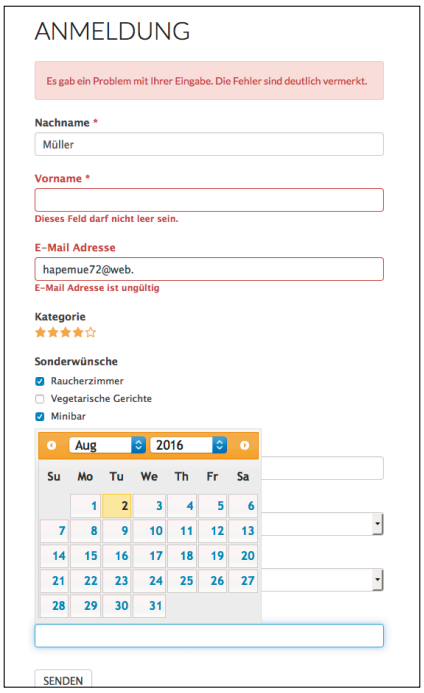
Zu den Bestandteilen eines Formular-Ökosystems gehört neben dem Formular-Generator selbst einiges mehr. So sollte die Weiterverarbeitung der eingegebenen Daten möglichst vielfältig angelegt sein, also zum Beispiel die Ablage in einer Datenbank, der Export in andere Programme oder die Umwandlung in einen Artikel innerhalb von WordPress. Des Weiteren braucht man fast immer Benachrichtigungen, etwa zum Informieren eines Admins, sobald etwas eingegeben wurde, oder an den Benutzer selbst, damit er per E-Mail eine Bestätigung seiner Dateneingabe erhält.

Erstellen eines Formulars

Nach der Installation des Plug-ins erhält die WordPress-Leiste links den neuen Eintrag *Formulare*. Mit *Neu hinzufügen* landen Sie sofort im Editor, der Ihnen schon einmal die ersten Schritte näherbringt (Bild 2).

Sie geben dem Formular einen aussagekräftigen Namen. Dieser Titel wird nicht gegenüber dem Benutzer sichtbar, sondern dient Ihnen dazu, das Formular in der Liste aller eingerichteten Formulare zu erkennen. Der im Frontend angezeigte Titel ergibt sich dagegen aus der Überschrift der Seite, auf der Sie das Formular später einbauen.

Um das Formular zusammenzustellen, ziehen Sie Elemente von der rechten Randleiste in den Arbeitsbereich. Nehmen Sie zum Beispiel das Element *Einzeiliger Text*. Das entspricht dem HTML-Element einer einfachen Input-Box. Nach dem Ziehen in die Mitte erscheint dort dann auch das passende Feld zusammen mit einer Feldüberschrift und einem optionalen Hinweistext darunter.



Mit Formidable lassen sich Formulare für WordPress zusammenstellen, die über alle wichtigen Funktionen wie Validierung oder nachgelagerte Aktionen verfügen (Bild 1)

Durch Anklicken und Ausfüllen können Sie die Überschrift, den Hinweistext und auch einen vorgegebenen Standardwert für den Feldinhalt festlegen.

Für Ihr erstes Formular können Sie nun weitere Felder einbauen und dabei die verschiedenen Feldtypen testen. Wollen Sie die Reihenfolge ändern, ziehen Sie einfach ein Feld per Drag and Drop an seinen neuen Platz.

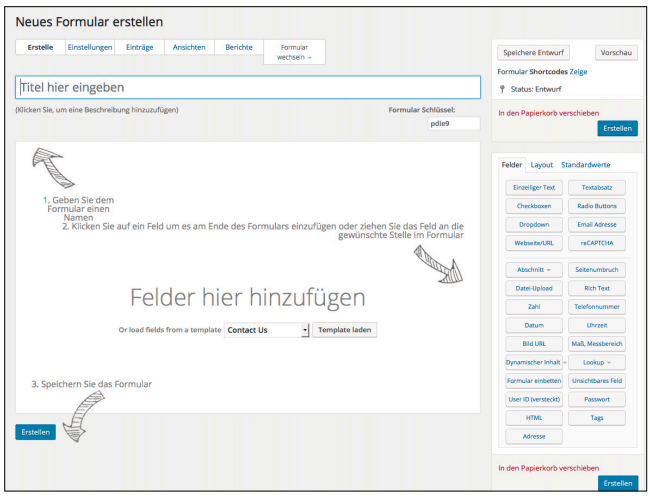
Formular veröffentlichen

Sind Sie fertig, speichern Sie das Formular durch einen Klick auf den Knopf *Erstellen*. In der Liste der Formulare finden Sie nun Ihre Kreation mit einer vom System zugewiesenen ID, dem Titel und weiteren Daten. Wichtig ist dabei vor allem der Inhalt der letzten Spalte *Shortcode*, die zum Beispiel so lauten könnte:*[formidable id=5]*.

Das ist der Code, mit dem Sie Ihr Formular in eine Seite oder einen Beitrag von WordPress einbauen. Kopieren Sie also diese Information in die Zwischenablage, legen Sie eine neue Seite an und fügen Sie den Shortcode als alleinigen Seiteninhalt ein.

Um das Formular zu sehen, müssen Sie die Seite nicht einmal veröffentlichen. Es genügt, sie in der Vorschau darstellen zu lassen. Sie sehen nun Ihre definierten Felder und den automatisch eingefügten *Submit*-Knopf mit der Standard-Beschriftung *Senden*. Geben Sie Daten ein und schicken Sie das Formular weg. Eine kurze Mitteilung erscheint, die sich für die Dateneingabe bedankt.

Wenn Sie nun im Backend die Übersicht der Formulare aufrufen, sehen Sie, dass der Zähler bei *Einträge* von *0* auf *1* gesprungen ist. Rufen Sie das Formular in der Liste auf und gehen zum Tab *Einträge*. Dort finden Sie Ihre Dateneingabe mit allen festgelegten Feldern und ein paar standardmäßig hinzugefügten Daten wie dem Erfassungszeitpunkt. Es gibt auch eine Spalte für das Aktualisierungsdatum. Denn Sie können festlegen, dass angemeldete WordPress-User die Eingabe der Daten jederzeit unterbrechen und später wieder aufnehmen dürfen. Bei umfangreichen Online-Fragebögen ist das sicher eine interessante



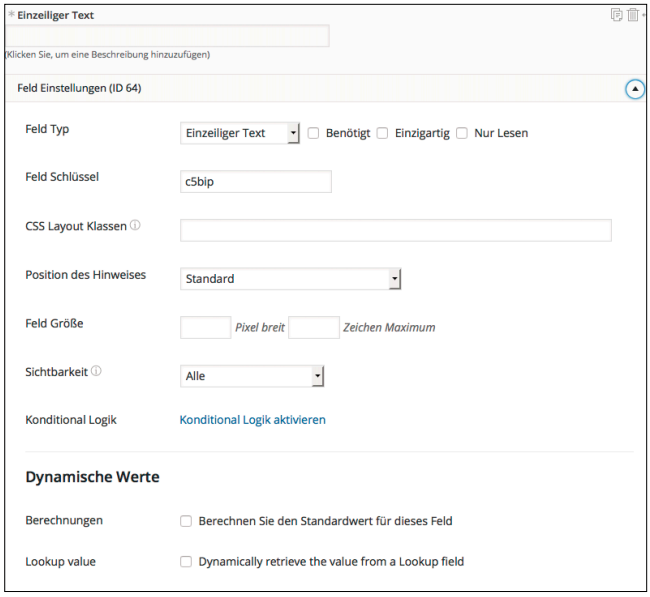
Wenn Sie ein neues Formular anlegen, zeigt Ihnen Formidable gleich, welche Schritte anfänglich notwendig sind (Bild 2)

Option. Mit dem Button *CSV herunterladen* holen Sie die bislang gespeicherten Einträge für die Weiterverarbeitung auf Ihren lokalen Computer herunter (nur Pro-Version). Sie sehen also: Ein Formular mit grundlegenden Anforderungen ist schnell umgesetzt.

Die Feldtypen von Formidable

Natürlich unterstützt das System alle von HTML bekannten einfachen Eingabefelder wie Checkboxen, Radio-Buttons, Select-Listen oder Dropdown-Felder. Diese funktionieren genauso wie bei der Verwendung im HTML-Quelltext, sind aber einfacher zu benutzen.

Als Besonderheit können Sie bei den Formularelementen, die mit mehreren Werten arbeiten, entweder die Optionen einzeln per Klick hinzufügen oder über den Link *Massenbearbeitung* alle Optionen auf einmal eingeben oder aus einer Vorlage per Zwischenablage hineinkopieren. Manche Feldtypen wie E-Mail-Adresse, Telefonnummer oder Zahl sind im Prinzip normale Textfelder, haben aber eine eingebaute Validierung entsprechend ihrem jeweiligen Zweck.



Öffnet man die Einstellungen für ein Formularfeld, bietet Formidable eine Vielzahl an Optionen für die Darstellung, Validierung und Weiterverarbeitung (Bild 3)

Eigenschaften von Feldern setzen

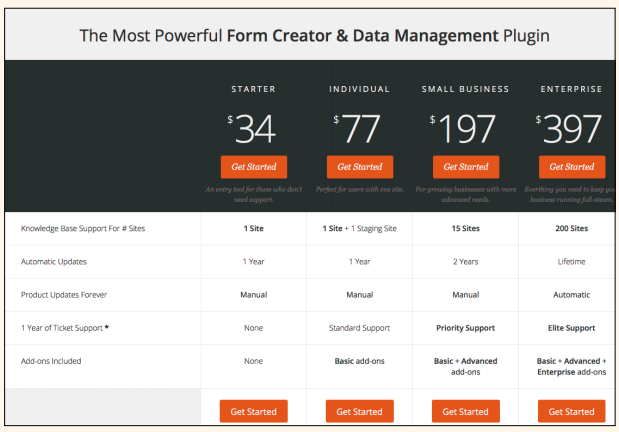
Klicken Sie auf die Schaltfläche *Feldeinstellungen* eines Feldes, so tut sich ein ganzes Arsenal an Optionen auf (Bild 3).

Für jedes Formular wichtig ist der Punkt *Benötigt*. Vergeben Sie dort ein Häkchen, so besteht Formidable darauf, dass der Benutzer hierin einen Eintrag vornimmt. Zur Kennzeichnung erhält das Feld auch einen kleinen roten Stern, der vorangestellt ist. ▶

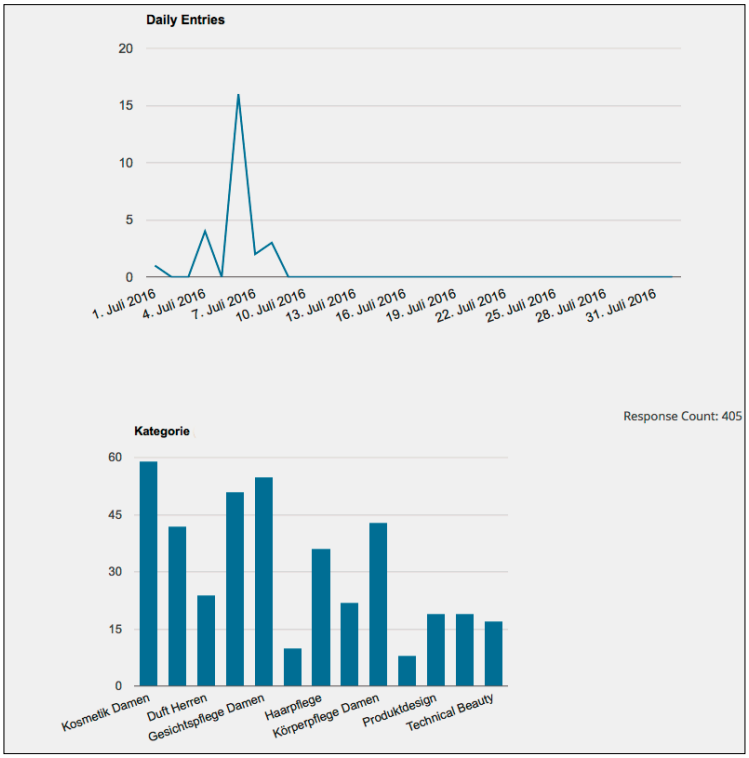
Die Versionen von Formidable

Das Plug-in ist unter der Bezeichnung Formidable Lite auch als kostenlose Version verfügbar.

Die Einschränkungen gegenüber der kostenpflichtigen Pro-Version betreffen so ziemlich alle Bereiche: Feldtypen, Feldoptionen und vor allem die Weiterverarbeitung von eingegebenen Daten. So erhalten Sie beispielsweise nur mit der Pro-Variante Statistik-Charts für die bisher eingegangenen Beiträge oder die Möglichkeit, durch selbst definierte Ansichten die Daten aufzubereiten. Auch der Einsatz von Add-ons ist der kostenpflichtigen Version vorbehalten. Die gibt es in verschiedenen Preisabstufungen. Für 34 Dollar erhält man die Grundversion, die den Einsatz auf einer einzigen Website erlaubt und ein Jahr lang automatische Updates beinhaltet. Nach Ablauf dieser Frist kommen Sie zwar immer noch in den Genuss von Programmauffrischungen, müssen die aber manuell einspielen. Interessant ist auch die nächste Stufe mit 77 Dollar Kosten: Sie erlaubt zusätzlich die Nutzung auf einem Staging-Server und beinhaltet eine Sammlung von Basic-Add-ons, etwa zur Verknüpfung von Formidable mit Mailchimp und der Nutzung von Bootstrap als Layout-Grundlage. Mehr zu den weiteren Varianten ersehen Sie auf der Seite *Pricing* auf der Formidable-Homepage.



Formidable gibt es in verschiedenen Lizenzvarianten, bei denen schon die günstigste Version gut einsetzbar ist



Über den Menüpunkt Berichte eines Formulars haben Sie jederzeit eine gute Übersicht, wie viele Eingaben gemacht wurden und wie die Verteilung auf die vorgegebenen Auswahloptionen aussieht (Bild 4)

Mit *Einzigartig* legen Sie fest, dass der Wert in diesem Feld in allen eingegebenen Daten nur einmal vorkommen darf. Wenn Sie also beispielsweise einem Feld für die E-Mail-Adresse des Benutzers diese Eigenschaft geben, dann erreichen Sie damit, dass er effektiv nur ein einziges Mal eine Dateneingabe vornehmen kann. Jeder Versuch, das Formular mit dieser Adresse nochmals auszufüllen, wird dann von Formidable abgewiesen.

Über den Link *Konditionale Logik* öffnen Sie bei den Feldeigenschaften einen weiteren Unterbereich. Dort lässt sich festlegen, dass die Darstellung des Feldes an bestimmte Bedingungen geknüpft ist. So könnten Sie beispielsweise bei einem Formular zum Thema Gesundheit festlegen, dass das Feld *Wie viele Packungen rauchen Sie am Tag?* nur erscheint, wenn der Benutzer das Feld *Raucher* angekreuzt hat.

Feldern wie *Zahl*, die einen klar umrissenen Inhalt haben, können Sie über die *Eigenschaften* einen Wertebereich zuweisen. Eine vollkommen beliebige Validierung der Inhalte kann das *Eigenschaften*-Fenster aber nicht bieten. Dazu müssen Sie sich eigene Funktionen in PHP schreiben und über Hooks an die gewünschten Felder binden. Dafür gibt es einige Beispiele in der Knowledge Base.

Vorhandene Einträge auswerten

Wie Sie einzelne Benutzer-Einträge ansehen und als CSV exportieren, haben Sie ja bereits beim ersten Testlauf gesehen. Aber es ist noch mehr möglich. Klicken Sie zum Beispiel auf *Berichte*, so erhalten Sie verschiedene statistische Auswertungen, zum Beispiel eine Aufschlüsselung der Benutzeraktivitäten über einzelne Tage oder die Verteilung der Häufigkeit bei vordefinierten Auswahlfeldern, wie etwa einem Dropdown-Feld (Bild 4).

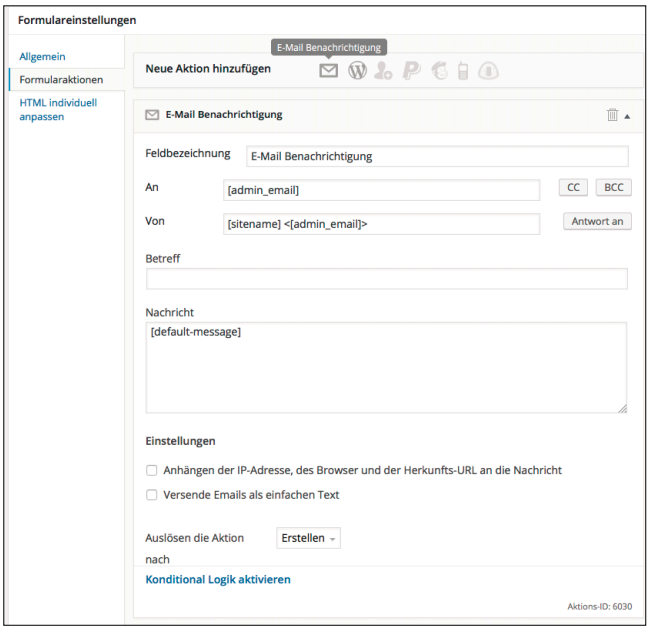
Die Ansichten bieten Ihnen die Möglichkeit, selbst definierte Auswertungen für die eingegebenen Daten zu bauen. Während die Liste der Einträge einfach stumpf alle Felder im

Kostenlose Alternative

Es gibt viele Plug-ins, die einige der Aufgaben von Formidable übernehmen und dabei umsonst zu haben sind. Allerdings decken diese immer nur Teilgebiete ab. So ist etwa Contact Form 7 ein Klassiker unter den Freeware-Plug-ins für die Kernaufgabe. Für den Export der Daten brauchen Sie aber wieder ein weiteres Plug-in.

Schnell summiert sich dann das persönliche Arsenal an Formular-Bausteinen auf fünf oder mehr Plug-ins zusammen. Das ist natürlich nervig, weil man schnell den Überblick verliert und das reibungslose Zusammenspiel aller Komponenten auch oft einiges an Einstellungen notwendig macht.

Da sind die 34 Dollar für Formidable Pro gut angelegt: Sie erhalten ein sofort lauffähiges und sehr gut ausgestattetes System. Taucht eine neue Anforderung im Bereich der Formulare auf, brauchen Sie meistens nicht im Internet nach Zusatzsoftware zu suchen, weil das Tool dies in der Regel schon beherrscht. Ein Blick auf die wohlgepflegte Knowlegde Base von Formidable (siehe Links) zeigt Ihnen meistens, wie Sie Ihre Wünsche umsetzen können.



Bei den Aktionen für ein Formular legen Sie fest, dass nach seinem Absenden E-Mail-Bestätigungen verschickt oder WordPress-Beiträge angelegt werden sollen (Bild 5)

Backend anzeigt, ist dieser Programmpunkt dazu da, ansprechende Listen mit frei definierbarem HTML zu produzieren, die dann im Frontend darstellbar sind.

Als Demo-Anwendung finden Sie dazu auf der Formidable-Seite ein Immobilien-Portal, bei dem die Nutzer selbstständig Angebote inklusive Bildern eingeben können, die dann in einer Liste optisch ansprechend dargeboten werden. Sie finden diese Anwendung auch in den automatisch mitinstallierten Vorlagen.

Aktionen für abgeschickte Formulare setzen

Standardmäßig bedankt sich ein Formidable-Formular beim Benutzer mit einer allgemein gehaltenen Meldung. Sie haben aber einige Möglichkeiten, dieses Verhalten anzupassen.

Dazu gehen Sie im Formular-Editor auf den Reiter *Einstellungen*. Auf der Unterseite *Allgemein* haben Sie schon einmal Zugriff auf die Benachrichtigungs-Optionen gegenüber dem User. Möchten Sie einfach nur den Meldungstext ändern, finden Sie ganz unten ein entsprechendes Feld, das den vorgegebenen Text enthält und von Ihnen geändert werden kann.

Mit den Einstellungen unter der Überschrift *Beim Absenden* legen Sie fest, ob statt der Meldung ein bestimmter URL angesprungen oder eine Seite von WordPress angezeigt werden soll. Wenn Sie also eine aufwendigere Dankesseite bauen möchten, ist dies die Option Ihrer Wahl.

Richtig interessant wird es aber, wenn Sie von der Einstellungs-Gruppe *Allgemein* auf *Formularaktionen* wechseln. Ohne weitere installierte Plug-ins können Sie hier aus der Symbolleiste die ersten beiden Icons wählen: *E-Mail Benachrichtigung* und *Beitrag erstellen*. Klicken Sie die erste Option an, und ein entsprechender Eintrag wandert in die Liste der Formularaktionen darunter. Klicken Sie den Eintrag an, um seine Optionen zu öffnen (Bild 5).

Schon mit den vordefinierten Einstellungen hat die E-Mail-Aktion einen praktischen Nutzen. Sie schickt dem Admin der WordPress-Site eine Nachricht, sobald ein neuer Formulareintrag erfolgt ist. Der durch den Shortcode *[default-message]* vorgegebene Inhalt der Nachricht ist eine nüchterne Aufstellung von Feldern und ihren individuellen Inhalten.

Möchten Sie zum Beispiel die E-Mail-Aktion dazu nutzen, um sich beim jeweiligen Benutzer zu bedanken, dann schreiben Sie bei *Nachricht* den gewünschten Text hinein. Dabei können Sie auch eingegebene Feldinhalte verwenden. Gibt es etwa im Formular ein Feld für den Namen, dann können Sie den Benutzer darüber ansprechen. Oder Sie teilen dem Einsender nochmals mit, was er in den Formularfeldern alles eingegeben hat. ▶

Interessante Feldtypen

Neben den altbekannten Arten von Eingabefeldern bietet das System auch einige Besonderheiten an.

Datum öffnet beim Anklicken des Feldes eine Pop-up-Kalenderauswahl, während **Zeit** eine Dropdown-Box mit stundenweisen Einträgen von 0:00 bis 23:00 anzeigt.

Datei-Upload gibt dem Benutzer die Möglichkeit, Bilder oder andere Dateien hochzuladen. Das kann er entweder per Drag and Drop erledigen oder einen klassischen Dateidialog nutzen. Sie als Admin bestimmen die dabei erlaubten Dateitypen und können eine maximale Dateigröße vorgeben. Die Dateien werden in der Medienbibliothek von WordPress in einem speziellen Unterordner für Formidable abgelegt. In den Einträgen des Formulars findet sich nur der Link auf die jeweilige Datei.

Ein wenig anders funktioniert **Bilder-URL**. Hier findet kein Upload statt, sondern Sie benennen lediglich eine komplette Adresse eines anderswo gespeicherten Bildes. In den Formulardaten sehen Sie dann ein Thumbnail des Bildes.

Mit **Abschnitt** fassen Sie mehrere Felder unter einer Überschrift zusammen. Neben der optischen Klammer bietet dieses Gruppierungsfeld auf Wunsch dem Benutzer auch die Möglichkeit an, den Abschnitt auf einen Mausklick hin zu verkleinern. Bei Formularen mit vielen Feldern ist das eine gute Möglichkeit, die Übersicht zu bewahren. Interessant für manche Anwendungen ist auch das Feature, dass Benutzer bei der Eingabe komplette Abschnitte vervielfältigen können. Stellen Sie sich dazu ein Formular vor, in dem man seine Kinder eintragen muss. Dazu könnte man natürlich im Formular Felder für die größte denkbare Anzahl von Kindern vorsehen. Eleganter geht es aber mit Abschnitt. Dort fassen Sie alle Daten für ein Kind zusammen, die Ihnen wichtig sind, und setzen in den Eigenschaften des Abschnitts das Häkchen bei *wiederholbar*. Dadurch erscheint im fertigen Formular am Ende des Abschnitts ein Knopf *Hinzufügen*, womit man die Felder für ein weiteres Kind anzeigen lässt.

reCAPTCHA baut ein CAPTCHA-Element in Ihr Formular ein, das den Missbrauch durch Skripts verhindern soll. Die Nutzung ist kinderleicht: Sie ziehen das Element einfach nur in Ihr Formular, und die Wahl der CAPTCHA-Variante, Darstellung und Auswertung der Eingabe erfolgen vollautomatisch vom System. Sie müssen nur ein kostenloses Konto bei reCAPTCHA anlegen und die Daten einmalig in *Globale Einstellungen* eintragen.

Rich Text verwendet wie WordPress selbst die JavaScript-Erweiterung TinyMCE, um dem User ein Eingabefeld mit Formatierungsmöglichkeiten anzubieten. Zusammen mit der Aktion *Beiträge erstellen* können Sie dadurch fertige WordPress-Artikel über ein Formular im Frontend anlegen lassen.

Über **Maß/Messbereich** schaffen Sie eine Skala in einem von Ihnen definierten Wertebereich in der Form von horizontalen Radio-Buttons. Die Eigenschaft *Zeige Optionen als Sterne* wandelt dann nicht bloß die Radioknöpfe zu Sternen um, sondern lässt das Feld genauso funktionieren wie die Bewertungssterne von Amazon und Co.

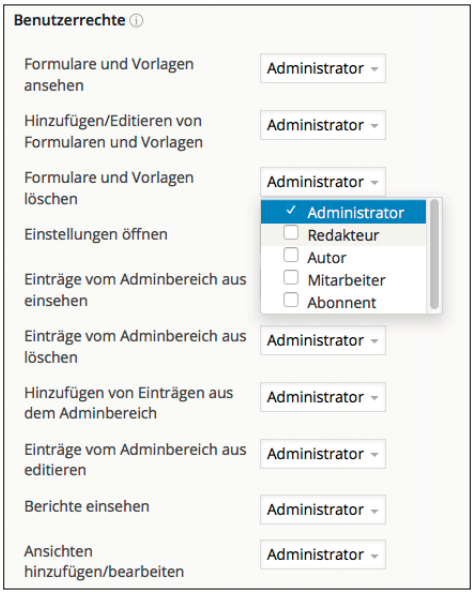
Hinter **HTML** verbirgt sich schließlich ein ganz einfacher Feldtyp, bei dem Sie individuellen HTML-Quelltext an der jeweiligen Stelle im Formular einfügen können, etwa für ausführliche Texthinweise zu bestimmten Feldern oder der Integration von Bildern.

Formidable listet Ihnen rechts alle Felder Ihres Formulars auf und Sie müssen nur daraufklicken, um sie an der aktuellen Cursorposition in die Nachricht einzufügen. Dabei verwendet das Tool standardmäßig die ID des Feldes, also lediglich eine Nummer. Damit wird die Nachricht etwas schwer zu entziffern.

Als Alternative bietet Ihnen das System darum das Umschalten auf Schlüssel an. Das ist ein Buchstaben- und Zahlencode, der in den Eigenschaften einen Feldes zu finden ist und den Sie selbst ändern können. Damit wird dann beispielsweise aus *Hallo [124] [125]* das für Benutzer deutlich verständlichere *Hallo [vorname] [nachname]*.

Sie dürfen auch mit HTML-Tags innerhalb des Nachrichtentextes arbeiten, um die E-Mails ansprechend zu gestalten. Wenn Sie dagegen lieber nüchterne Textnachrichten senden möchten, klicken Sie die Checkbox *Versende Emails als einfachen Text* an.

Wie auch bei der Formularbearbeitung selbst können Sie den E-Mail-Versand mit einer Entscheidungslogik versehen. Dazu klicken Sie ganz unten bei den Eigenschaften der E-Mail-Aktion auf den Link *Konditionale Logik*. Es öffnet sich daraufhin eine Sammlung von Auswahlboxen und Feldern. Damit stellen Sie sich Bedingungen zusammen wie zum Beispiel *E-Mail nur versenden, wenn Feld ‚Bestätigung erwünscht' ausgefüllt ist*.



Formidable bietet Ihnen eine sehr feine Abstufung bei der Vergabe von Rechten für bestimmte Aktionen mit dem Formularsystem (Bild 6)

Beiträge durch Formulare anlegen lassen

Eine andere Aktion bietet das Anlegen eines Beitrags, einer Seite oder eines Custom Posts an. Bei den Einstellungen können Sie recht genau festlegen, wie Titel, Auszug, Inhalt und weitere Elemente aus den Formularfeldern gebildet werden sollen.

So lässt sich der Inhalt aus einem einzigen Feld generieren, das zum Beispiel dank Rich Text bereits vorformatiert ist. Oder Sie nutzen ein Gerüst aus statischem Text mit eingestreuten Feldvariablen wie im Mail-Body bei der E-Mail-Aktion.

Damit nicht jeder beliebig Quatsch posten kann, der sofort auf der Website sichtbar ist, legen Sie zusätzlich den Status des Beitrags fest. Geben Sie zum Beispiel *Entwurf* vor, dann muss der Beitrag erst noch freigegeben werden, bevor er sichtbar wird.

Berechtigungen gezielt vergeben

Formidable verträgt auch Installationen, in denen eine große Menge an Formularen oder Einträgen pro Formular aufkommt. Bei solchen Fällen ist es dann vielleicht sinnvoll, die Arbeit daran auf mehrere Personen zu verteilen. Das Formularsystem erlaubt es dazu, Berechtigungen für die verschiedenen Aktionen an WordPress-Benutzerrollen zu vergeben. Wenn es beispielsweise bei Ihrer Anwendung oft um sensible persönliche Daten geht, könnten Sie einen Mitarbeiter zwar dazu berechtigen, neue Formulare zu erstellen, die Einsicht in die Userdaten aber abblocken.

Um Rechte zu vergeben, gehen Sie zum Menüpunkt *Globale Einstellungen*. Dort sehen Sie dann eine Liste von Aktionen und dahinter jeweils die Benutzerrollen, die dazu freigegeben sind. In der Grundversion ist das durchgängig nur die Rolle *Administrator*.

Eine Sonderstellung nimmt die Eigenschaft *Sichtbarkeit* bei den Feldern ein. Hier können Sie für einzelne Elemente eines Formulars festlegen, dass eine bestimmte Benutzerrolle notwendig ist, um das Feld im Frontend sehen zu können (Bild 6).

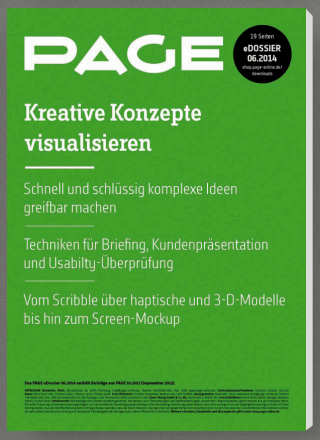
Wenn Sie jemandem das ganz oben in der Liste zu findende Recht verwehren, Formulare und Vorlagen einzusehen, ist er völlig aus Formidable ausgeschlossen. Er sieht dann nicht einmal das Menü *Formulare* im WordPress-Backend. ■

Links zum Thema

- Homepage des Formular-Systems Formidable
  www.formidablepro.com
- Zentrale Anlaufstelle für die Dokumentation
  www.formidablepro.com/knowledgebase
- Alle Feldtypen von Formidable, getrennt in die Standard-Felder und diejenigen, die der Pro-Version vorbehalten sind
  www.formidablepro.com/knowledgebase/field-types
- Homepage von WordPress
  www.wordpress.org
- WordPress-Tutorial
  http://wp-schulung.de/wordpress-handbuch
- WordPress-Plug-ins
  https://de.wordpress.org/plugins

Markus Schraudolph
ist Journalist und Programmierer. Er schreibt seit 16 Jahren Bücher und Artikel für Fachzeitschriften. Seine Schwerpunktgebiete als Programmierer sind die Webprogrammierung und Datenbanken.

Downloaden, aufschlauen!



PAGE eDossiers – Best-of-Kompilationen aus PAGE und WEAVE im Originallayout: PDFs einfach und jederzeit runterladen in unserem Online-Shop **shop.page-online.de/downloads**

PAGE
Das Magazin der Kreativbranche

Ein Magazin der Verlagsgruppe Ebner Ulm • www.ebnerverlag.de

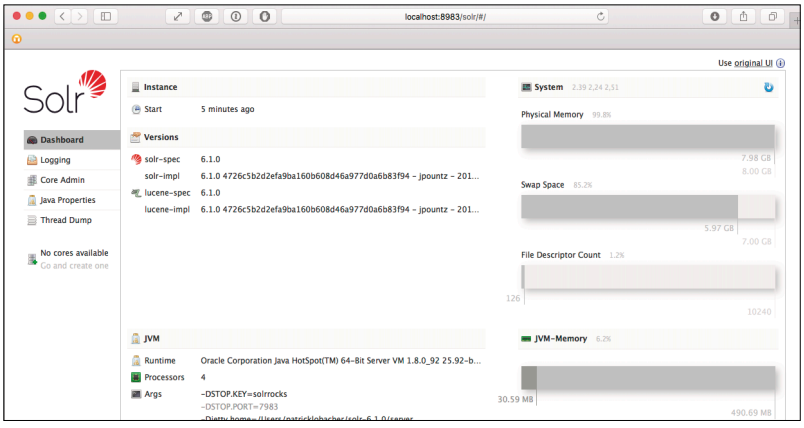
PHP

Solr mit PHP

Die Enterprise-Such-Engine Solr und PHP bilden eine harmonische Kombination.

Solr ist eine in Java geschrieben Enterprise-Search-Plattform auf Open-Source-Basis, die das Apache-Lucene-Projekt erweitert. Ihre Hauptmerkmale sind: Volltextsuche, Treffer-Highlighting, facettierte Suche, Indexing in Echtzeit, dynamisches Clustering, Datenbank-Integration, NoSQL-Unterstützung und Rich Document Handling (zum Beispiel Word, PDF).

Solr unterstützt verteilte Suche sowie Index-Replikation und ist für Hochskalierbarkeit und Fehlertoleranz konzipiert worden. Weltweit ist Solr nach Elasticsearch die beliebteste Suchmaschine.



So präsentiert sich das Admin-Panel von Solr (Bild 1)

Grundsätzlich läuft Solr als Standalone-Volltext-Such-Server. Dabei nutzt es die Lucene-Java-Search-Bibliothek als Kern für die Volltext-Indexierung und -Suche. Darüber hinaus besitzt Solr eine REST-Schnittstelle (basierend auf HTTP/XML) sowie ein JSON-API. Somit ist ein Zugriff von jeder populären Programmiersprache aus möglich. Die externe Konfiguration von Solr erlaubt die Anpassung an eine Vielzahl von Applikationen, ohne über Java-Kenntnisse zu verfügen. Zudem

gibt es eine Plug-in-Architektur, die über noch deutlich erweiterte Fähigkeiten verfügt.

Installation von Solr

Zunächst einmal benötigt Solr Java auf dem System. Um zu überprüfen, ob Java vorhanden ist, kann man die Konsole (beziehungsweise unter Windows die Kommandozeile) bemühen:

```
$ java -version

java version "1.6.0_65"
Java(TM) SE Runtime Environment
(build 1.6.0_65-b14-468-11M4833)
Java HotSpot(TM) 64-Bit Server VM
(build 20.65-b04-468, mixed mode)
```

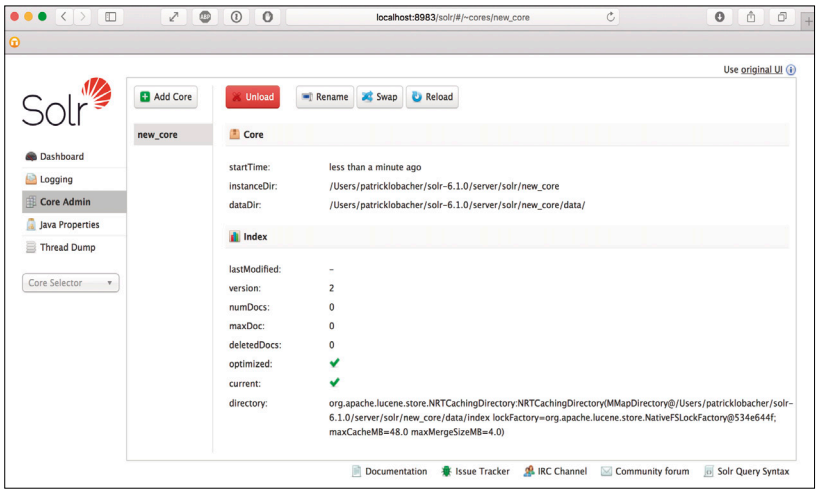
Wichtig ist vor allem auch, dass man die Java Runtime Environment (JRE) geladen und installiert hat. Diese kann man unter www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html laden. Danach laden wir Solr als ZIP-Datei von der Apache-Lucene-Website (http://lucene.apache.org/solr/downloads.html) herunter und entpacken das Archiv.

Für die Installation unter Windows geht man nun wie folgt vor: Rechtsklick auf das Archiv und Extrahieren in den Ordner *C:\solr-6.1.0*. Um Solr zu starten, wechselt man zum Windows-Kommando-Prompt mittels *Start* und *Run*. Hier tippt man *cmd* ein. In der Konsole erfolgt dann folgende Eingabe:

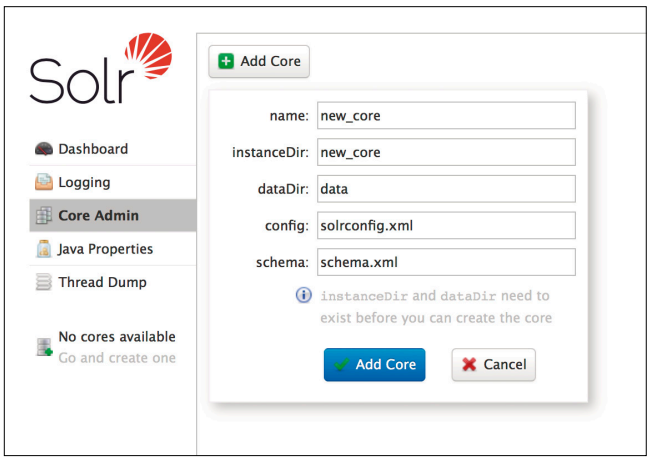
```
$ cd C:\solr-6.1.0
$ bin/solr start
```

Für die Installation unter Linux/Mac OS X kopiert man zuerst die Archiv-Datei ins Home-Verzeichnis und führt dann folgende Befehle auf der Kommandozeile aus:

```
$ unzip solr-6.1.0.zip
$ cd ~/solr-6.1.0
```



Die Option Core Admin präsentiert diesen Bildschirm (Bild 3)



Über diese Eingabemaske erfolgt das Anlegen eines Cores (Bild 2)

```
$ bin/solr start
```

Sofern man als Betriebssystem El Capitan einsetzt, ist es wichtig, die Java-Version wie folgt anzupassen: http://stackoverflow.com/questions/34201990/unsupported-major-minor-version-on-mac-os-x-el-capitan/34201991#34201991. Daraufhin läuft der Solr-Server unter dem Port 8983: http://localhost:8983/solr (Bild 1).

Anschließend brauchen wir noch einen Solr-Kern (Core). Dafür legen wir zunächst ein Verzeichnis *new_core* im Unterverzeichnis *server/solr* an. Nun kopieren wir das Verzeichnis *server/solr/configsets/basic_configs/conf* in das Verzeichnis *new_core*. In der Admin-Oberfläche können wir nun per Klick auf *Core Admin* und ein abschließendes *Add Core* den neuen Core anlegen (Bild 2, Bild 3).

Nun testen wir den Core im Browser und erhalten – im Erfolgsfall – eine JSON-Ausgabe:

```
// Im Browser
http://localhost:8983/solr/new_core/admin/ping/?wt=json

// Ergebnis
{"responseHeader":{"zkConnected":null,
"status":0,"QTime":37,"params":
{"q":"{!lucene}*:*","distrib":"false",
"df":"text","rows":"10","wt":"json",
"echoParams":"all"}},"status":"OK"}
```

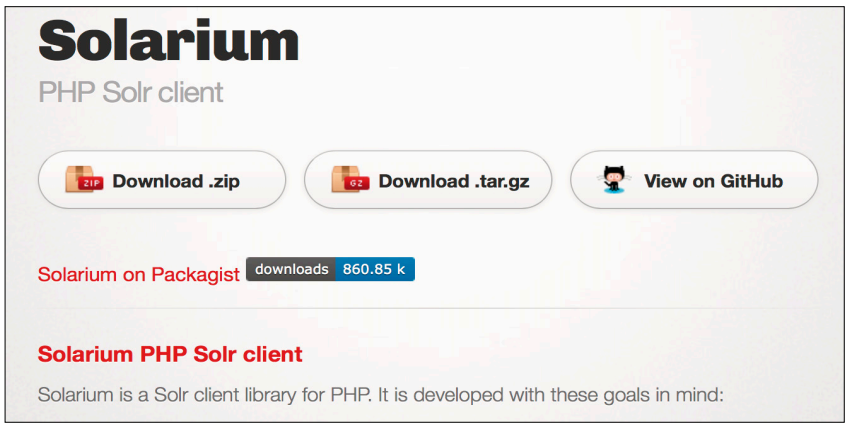
Man könnte nun direkt PHP für den Zugriff auf Solr verwenden – beispielsweise über folgendes Skript, in dem das API via *curl* abgefragt wird:

```
<?php
$curl = curl_init("http://localhost:8983/
solr/new_core/admin/ping/?wt=json");
curl_setopt($curl,
CURLOPT_RETURNTRANSFER, 1);
$output = curl_exec($curl);
$data = json_decode($output, true);
```

```
echo "Ping Status :
".$data["status"]."\n";
```

Deutlich bequemer erfolgt der Zugriff aber über eine PHP-Bibliothek. Hier gibt es eine große Anzahl an Bibliotheken, die jedoch teilweise nicht mehr regelmäßig aktualisiert werden.

Anders ist dies bei Solarium (www.solarium-project.org). Die Bibliothek wird ständig aktualisiert und ist sogar über Composer installierbar. Hierfür wird zuerst Composer benötigt (https://getcomposer.org). Anschließend kann Solarium über die Kommandozeile installiert werden:



Die Solarium-Website bietet verschiedene Download-Optionen (Bild 4)

```
$ composer create-project --no-dev solarium/solarium
```

Dies sorgt dafür, dass ein Unterverzeichnis *solarium* angelegt wird, das wir (wenn nicht ohnehin schon geschehen) in den Document-Root des Webservers verschieben (Bild 4).

Im nächsten Schritt legen wir eine PHP-Datei mit dem Namen *index.php* im Document-Root mit dem in Listing 1 gezeigten Inhalt an. Hier wird zunächst der Autoloader im Unterverzeichnis *solarium/vendor/* eingebunden. Anschließend wird die Konfiguration festgelegt. Dabei wird das Konzept der Endpoints verwendet, das je Core eine unterschiedliche Konfiguration zulässt.

Listing1: index.php

```
<?php

error_reporting(E_ALL);
ini_set(‚display_errors‘, true);

require __DIR__.‘/solarium/vendor/autoload.php‘;

$config = array(
  „endpoint“ => array(„localhost“ =>
  array(„host“=>“127.0.0.1“,
  „port“=>“8983“, „path“=>“/solr“,
  „core“=>“new_core“,)
) );

$client = new Solarium\Client($config);
$ping = $client->createPing();

try {
  $result = $client->ping($ping);
  echo ‚Ping query successful‘.chr(10);
  var_dump($result->getData());
} catch (Solarium\Exception $e) {
  echo ‚Ping query failed‘;
}
```

Nun erstellt man ein Client-Objekt und ruft dort die *Ping*-Methode auf, um ein Ping-Objekt zu erstellen. Dieses wird anschließend ausgeführt und der Inhalt (sofern vorhanden) zurückgegeben.

Sollte die Ausgabe nicht zufriedenstellend sein, kann man versuchen, über das Logfile herauszufinden, was nicht funktioniert (Listing 2).

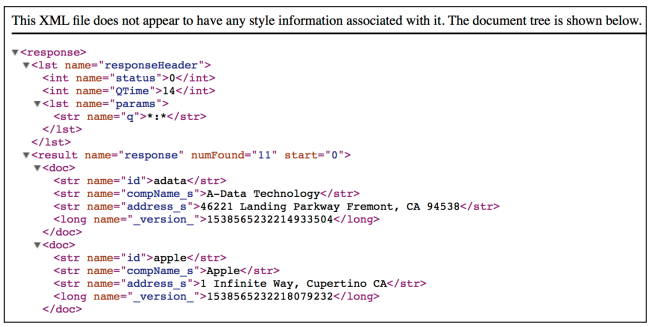
Demo-Daten erzeugen

Damit für den weiteren Verlauf des Artikels Daten im System vorhanden sind, erzeugen wir die Demo-Daten, die mit der Solr-Installation mitgeliefert wurden. Hierfür starten wir den Indexer für JSON. Dieser importiert die Daten aus dem – in der Installation enthaltenen – Demo-File in den Index:

```
$ bin/post -c new_core example/exampledocs/
manufacturers.xml
```

Listing 2: Logfile

```
$ cd <solr_home>/server/logs/
$ tail -f solr.log

2016-06-30 12:04:29.845 INFO (qtp475266352-18) [
x:new_core] o.a.s.c.S.Request [new_core]
webapp=/solr path=/admin/ping params={json.nl=
flat&omitHeader=true&wt=json} status=0 QTime=2
```



Demo-Daten werden im XML-Format angezeigt (Bild 5)

Listing 3: Schema erweitern

```
$ curl -X POST -H 'Content-type:application/json'
--data-binary '{
  "add-field":{
    "name":"name",
    "type":"string",
    "stored":true }
}' http://localhost:8983/solr/new_core/schema'

{
  "responseHeader":{
    "status":0,
    "QTime":36}}
```

Um zu überprüfen, wie viele Dokumente im Index sind, kann man eine Query absetzen:

```
http://localhost:8983/solr/new_core/select/?q=*:*
```

Das Feld *numFound* gibt dabei die Anzahl der gefundenen Dokumente zurück (Bild 5).

Das Solr-Schema

Das interne Solr-Schema besteht vorwiegend aus Feldern und Feld-Typen. Letztere definieren das Feld und die Verarbeitung der Daten. Es können nur Metadaten erfasst werden, die als Definition im Schema existieren. Per Default sind bereits einige Felder im Schema definiert.

Weitere Felder muss man selbst hinzufügen. Für einen Überblick über alle verfügbaren Felder kann man sich das Schema anzeigen lassen. Dafür nutzen wir das sogenannte Schema-API, das die bisherige Datei *schema.xml* ersetzt:

```
http://localhost:8983/solr/new_core/schema bzw. http://
localhost:8983/solr/#/new_core/schema
```

Das Schema-API wiederum legt seine Daten in der Datei *<solr_home>/server/solr/new_core/conf/managed-schema* ab. Diese Datei darf aber nicht editiert werden. ▶

```
▼<response>
 ▼<lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">0</int>
   ▼<lst name="params">
      <str name="q">name_s:*</str>
    </lst>
  </lst>
 ▼<result name="response" numFound="2" start="0">
  ▼<doc>
     <str name="id">1000</str>
     <str name="author_s">Rick Riordan</str>
    ▼<arr name="cat_ss">
       <str>book</str>
       <str>hardcover</str>
     </arr>
     <str name="name_s">The Lightning Thief</str>
     <long name="_version_">1538848873455288320</long>
   </doc>
  ▼<doc>
     <str name="id">1001</str>
     <str name="author_s">Rick Riordan</str>
    ▼<arr name="cat_ss">
       <str>book</str>
       <str>paperback</str>
     </arr>
     <str name="name_s">The Sea of Monsters</str>
     <long name="_version_">1538848873457385472</long>
   </doc>
  </result>
</response>
```

Auch die hinzugefügten Daten werden als XML präsentiert (Bild 6)

Listing 4: Daten mit PHP hinzufügen

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
require __DIR__.'/solarium/vendor/autoload.php';
$config = array(
  "endpoint" => array(
    "localhost" => array(
      "host"=>"127.0.0.1",
      "port"=>"8983",
      "path"=>"/solr",
      "core"=>"new_core",
    )
  )
);

$client = new Solarium\Client($config);
$updateQuery = $client->createUpdate();
$doc1 = $updateQuery->createDocument();
$doc1->id = 1000;
$doc1->author_s = 'Rick Riordan';
$doc1->cat_ss = ['book','hardcover'];
$doc1->name_s = 'The Lightning Thief';

$doc2 = $updateQuery->createDocument();
$doc2->id = 1001;
$doc2->author_s = 'Rick Riordan';
$doc2->cat_ss = ['book','paperback'];
$doc2->name_s = 'The Sea of Monsters';

$updateQuery->addDocuments(array($doc1, $doc2));
$updateQuery->addCommit();

$result = $client->update($updateQuery);

echo '<b>Update query executed</b><br/>';
echo 'Query status: ' . $result->getStatus(). '<br/>';
echo 'Query time: ' . $result->getQueryTime();
```

Wir wollen nun ein neues Feld mit dem Namen *name* zum Schema hinzufügen. Dafür können wir das Feld entweder direkt im GUI anlegen oder aber auf der Kommandozeile die Befehle laut Listing 3 absetzen. Hier wird festgelegt, dass der Name des Feldes *name* lauten soll und der Typ *string*. Eine entsprechende Abfrage bestätigt das korrekte Hinzufügen: http://localhost:8983/solr/new_core/schema/fields/name.

Neben den definierten Feldern gibt es sogenannte dynamische Felder, die per Convention over Configuration bereits vorab definiert sind. So kann man ein Feld *name_s* verwenden, weil durch die Endung *_s* definiert ist, dass der Typ *string* ist. Weitere mögliche Endungen sind: *_t* (für allgemeinen Text), *_i* (für Integer), *_f* (für Float), *_d* (für Double), *_dt* (für Date) oder *_p* (für eine Location). Will man mehrere Werte angeben (zum Beispiel ein Array aus Strings), so verwendet man *_ss*.

Daten hinzufügen

Im nächsten Schritt wollen wir einige Daten per PHP hinzufügen. Dazu verwenden wir die Datei *index.php* als Ausgangsbasis (Listing 4).

Zunächst wird hier wieder ein Solarium-Client initialisiert und darauf die *createUpdate()*-Methode angewendet, um später Daten hinzufügen zu können. Anschließend werden zwei Dokumente erstellt (über *createDocument())* und mit den entsprechenden Daten (die über dynamische Felder definiert wurden) versorgt. Über die Methode *addDocuments()* wurden diese zum Update hinzugefügt und per *addCommit()* so ausgewählt, dass diese beim nächsten Update (über *update())* auch wirklich ausgeführt werden. Eine kurze Abfrage zeigt, dass die Daten im Index angekommen sind: http://localhost:8983/solr/new_core/select/?q=name_s:* (Bild 6).

Im nächsten Schritt wollen wir Daten per PHP auch ändern. Dazu verwenden wir dieselbe Systematik wie beim Einfügen – mit der Ausnahme, dass wir die *addDocument()*-Methode mit dem Flag *overwrite* = *true* (der zweite Parameter) aufrufen. Im Beispiel fügen wir an den Titel *(name_s)* noch ein Ausrufezeichen hinzu (Listing 5).

Listing 5: Daten mit PHP ändern

```
...
$updateQuery = $client->createUpdate();

$updatedoc = $updateQuery->createDocument();
$updatedoc->id = 1000;
$updatedoc->author_s = 'Rick Riordan';
$updatedoc->cat_ss = ['book','hardcover'];
$updatedoc->name_s = 'The Lightning Thief!';

$updateQuery->addDocument($updatedoc, true);
$updateQuery->addCommit();

$result = $client->update($updateQuery);
...
```

Listing 6: Daten mit PHP selektieren

```
...
$query = $client->createSelect();
$query->setQuery('cat_ss:hardcover');
$query->setStart(0)->setRows(3);
$query->setFields(['id','author_s','name_s','cat_
ss']);
$query->addSort('name_s',$query::SORT_ASC);
$resultSet = $client->select($query);
foreach($resultSet as $doc)
  {
    echo PHP_EOL."-------".PHP_EOL;
    echo PHP_EOL."ID : ".$doc->id;
    echo PHP_EOL."Name : ".$doc->name_s;
    echo PHP_EOL."Author : ".$doc->author_s;
    echo PHP_EOL."Categories : ".implode($doc->
    cat_ss,',').PHP_EOL;
  }
echo PHP_EOL."Results found: ".$resultSet->
getNumFound();
```

Um Daten zu löschen, wird die Methode *addDeleteQuery()* aufgerufen, die zunächst eine Suche durchführt und die gefundenen Dokumente anschließend löscht:

```
$deleteQuery = $client->createUpdate();
$deleteQuery->addDeleteQuery('id:1000');
$deleteQuery->addCommit();
$result = $client->update($deleteQuery);
...
```

Als Argument verwendet man *key:value*. Die Methode nimmt auch ein Array an, wenn man mehrere Suchkriterien hat. Ist

Listing 7: Facetten ausgeben

```
...
$query = $client->createSelect();
$query->setQuery('author_s:"Rick Riordan"');
$facetset = $query->getFacetSet();
$facetset->createFacetField('cat')->
setField('cat_ss');
$facetset->setLimit(5);
$facetset->setMinCount(1);
$facetset->setMissing(true);

$resultSet = $client->select($query);
$facetData = $resultSet->getFacetSet()->
getFacet('cat');
foreach($facetData as $item => $count) {
  echo $item.": [".$count."] <br/>".PHP_EOL;
}
```

man im Besitz der ID, so kann man das Dokument auch direkt über die Methode *addDeleteById()* löschen.

Nun wollen wir auch Daten über PHP selektieren. Dafür nutzen wir die Methode *selectQuery()*, um grundsätzlich eine Selektion durchzuführen. Hier wird ebenfalls mit *key:value* gerarbeitet. Anschließend kann man mittels *setStart()* definieren, mit welchem Ergebnis gestartet werden soll (dabei entspricht *0* dem ersten Ergebnis), und mit *setRows()* wird angegeben, wie viele Suchergebnisse man abfragen möchte.

Über *setFields()* wird festgelegt, welche Felder im Ergebnisset enthalten sein sollen, und über *addSort()* wird die Sortierung festgelegt. Sobald man die Query mittels *select()* abgeschickt hat, kann man über *getNumFound()* die Anzahl der gefundenen Datensätze abfragen (Listing 6).

Facettierung

Schließlich wollen wir noch die Facettierung betrachten. In unserem Beispiel haben wir die Eigenschaft *cat* als Kategorie verwendet. Bei der Suche nach einem Autor wollen wir daher die Facette *Kategorie* entsprechend ausgeben (Listing 7).

Zunächst einmal fomulieren wir eine normale Query, die alle Datensätze selektiert, die *Rick Riordan* beinhalten. Dann definieren wir ein Facettierungs-Feld *cat*, das auf dem Datenfeld *cat_ss* basiert. Dann limitieren wir das Ergebnis, indem wir die Methode *setLimit()* anwenden. Die Methode *setMinCount()* definiert, dass wir als Ergebnis-Set all jene haben wollen, bei denen mindestens einmal eine Facette vorkommt. Würden wir dies auf den Wert *2* erhöhen, so wäre nur noch *book* enthalten (da dies zweimal vorkam).

Eine weitere Möglichkeit der Facettierung wäre es beispielsweise, Zahlen in Gruppen zu zerlegen. Wenn man zum Beispiel ein Preisfeld hat, dann könnte man dieses Feld in Abständen von zum Beispiel 10 Euro als Facette anzeigen lassen, also 0–10 Euro, 11–20 Euro, 21–30 Euro et cetera. Dafür legt man mit *setStart()* den Startpunkt und mit *setEnd()* den Endpunkt fest. Über *setGap()* wird dann der Abstand definiert.

Abschließend wollen wir uns noch das Solr-Feature Highlighting ansehen – also das Hervorheben des Suchergebnisses (Listing 8).

Hier wird mit *getHighlighting()* zunächst die Highlighting-Komponente geladen und entsprechend konfiguriert. Unter anderem wird festgelegt, dass das für das Feld *author_s* die Hervorhebung mittels Bold-Tag geschehen soll. Über *setSnippets()* werden dann noch die maximale Anzahl an Snippets im Suchergebnis und mittels *setFragSize()* die Größe des Fragments festgelegt, das hervorgehoben wird.

Links zum Thema

- Solr-Website
  http://lucene.apache.org/solr
- Solarium-Website
  www.solarium-project.org
- Solarium bei GitHub
  https://github.com/solariumphp/solarium
- Solarium-Dokumentation
  http://solarium.readthedocs.io/en/stable/s
- Solr-Tutorial
  http://yonik.com/solr-tutorial
- Apache-Lucene-Projekt
  https://lucene.apache.org
- Composer
  https://getcomposer.org
- Java Runtime Environment (JRE)
  https://www.java.com/de/download

Listing 8: Highlighting

```
...
$query = $client->createSelect();
$query->setQuery('author_s:Rick*');
$hl = $query->getHighlighting();
$hl->getField('author_s')->setSimplePrefix('<b>')->
setSimplePostfix('</b>');
$hl->setSnippets(2);
$hl->setFragSize(6);
$hl->setMergeContiguous(true);
$hl->setHighlightMultiTerm(true);

$resultSet = $client->select($query);
$hlresults = $resultSet->getHighlighting();

foreach($resultSet as $doc)
{
  $hldoc = $hlresults->getResult($doc->id);
  $hlauthor = implode(',',
  $hldoc->getField('author_s'));
  echo $hlauthor.PHP_EOL;
}
```

Fazit

Solr ist eine extrem leistungsfähige und flexible Enterprise-Suche, die mit PHP bequem angesprochen werden kann. So ist beispielsweise auch eine Integration in PHP-Applikationen wie CMS, E-Commerce oder CRM leicht möglich. ■

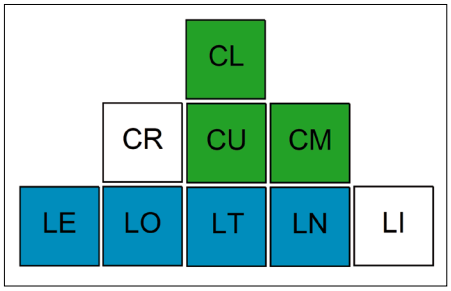
Patrick Lobacher
ist Digital Native, Entwickler, Berater, Coach und Autor zahlreicher Fachbücher und Fachartikel. Er ist Vorstandsvorsitzender der Pluswerk AG, die digitale Kommunikationslösungen konzipiert, umsetzt und betreut.
http://pluswerk.ag

RESTFUL HTTP

Etablierte Alternative

RESTful HTTP ist eine etablierte Alternative für SOAP-Webservices.

Eine wesentliche Anforderung des REST-Architekturstils ist der Einsatz von Hypermedia. Einfache Erweiterbarkeit, Selbstbeschreibung und geringe Kopplung zwischen Client und Server sind die wesentlichen Vorteile von REST und Hypermedia. Wie funktionieren diese Hypermedia-APIs und für welche Anwendungsfälle sind sie geeignet?



Hypermedia-Faktor von HTML nach Amundsen (Bild 1)

Wer REST implementieren und dabei Hypermedia richtig einsetzen will, muss zunächst einmal verstehen, was mit Hypermedia überhaupt gemeint ist. Während sich der Begriff Hypertext ausschließlich auf vernetzte Textformate bezieht, schließt dessen Oberbegriff Hypermedia auch andere Medien wie zum Beispiel Filme oder Bilder mit ein. Diese Medien könnte man konsequenterweise als Hyperfilme und Hyperbilder bezeichnen.

Ein bekanntes Beispiel für Hypertext beziehungsweise Hypermedia ist das im Web allgegenwärtige HTML. Dieser registrierte Medientyp hat ein strukturiertes Format und bildet Beziehungen zu anderen Objekten durch Links ab. Webbrowser wissen, wie sie diese Hypermedia-Steuerelemente nutzen können. Die Hypermedia-Steuerelemente sind ein wesentlicher Unterschied zwischen Hypermedia- und RPC-basierten Ansätzen.

Mike Amundsen schlug vor, die Hypermedia-Unterstützung eines Medientyps wie HTML durch den H-Faktor zu messen (http://amundsen.com/hypermedia/hfactor). Mit Hilfe des H-Faktors können Medientypen miteinander verglichen werden. Der Vergleich kann wiederum dabei helfen, einen oder mehrere passende Medientypen für die Implementierung eines API auszuwählen.

Das HTML-Snippet in Listing 1 zeigt verschiedene Hypermedia-Steuerelemente. Das HTML-Tag *<img/>* ist laut Amundsens Klassifikation ein Embedding Link (LE), der laut dem HTML-Standard mit *HTTP GET* genutzt wird. Ebenfalls mit *HTTP GET* wird der Outbound Link (LO) *<a/>* aufgerufen. Diese Links dienen zur Navigation zwischen verschiedenen Ressourcen. Das Form-Element *<form/>* kann entweder mit *HTTP GET* oder mit *HTTP POST* genutzt werden. Deswegen dient es sowohl für Templated Queries (LT) als auch für nicht idempotente Updates (LN). Der H-Faktor berücksichtigt neben den bereits genannten Links auch die Unterstützung von Steuerungsdaten.

Das Form-Element im HTML-Beispiel hat das Attribut *enctype*. Es spezifiziert, wie die Form-Daten für die Übertragung zum Server mit *HTTP POST* encodiert werden. Laut H-Faktor handelt es sich hierbei um Steuerungsdaten für Update Requests (CU). Ein Webbrowser weiß, wie diese Steuerungsdaten zu nutzen sind, und erzeugt entsprechende Content-Header (www.w3.org/TR/html5/forms.html).

Das Form-Element gibt mit dem Attribut *method* an, ob *HTTP POST* oder *HTTP GET* verwendet werden soll. Dies sind Steuerungsdaten für Interface Methods (CM). Schließlich sei noch auf die Angabe der Link-Relation *[www.iana.org/assignments/link-relations/link-relations.xhtml]* im *<a/>*-Element hingewiesen. Dies sind Steuerungsdaten für Links (CL).

Zusammenfassend ist die Hypermedia-Unterstützung von HTML in Bild 1 dargestellt. Nicht unterstützt werden Steuerungsdaten für Read Requests (CR) und Links für idempotente Updates (LI).

Hypermedia versus RPC

Wenn Entwickler gegen eine RPC-artige Schnittstelle programmieren sollen, brauchen sie eine Liste mit den von der Schnittstelle unterstützten Abfragen. Entsprechend dieser Informationen bauen sie dann ihre Applikation. Das Ergebnis

Listing 1: Hypermedia-Steuerelemente in HTML

```
<html>
  <head>
    <title>Hypermedia-Steuerelemente in HTML</title>
  </head>
  <body>
    <img src="..."/>
    <a rel="about" href="...">About</a>
    <form action="..." method="post"
    enctype="multipart/form-data">
      First name: <input type="text"
      name="fname"><br>
      Last name: <input type="text"
      name="lname"><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

ist eine Applikationen mit hartcodiertem Detailwissen über die Schnittstelle, sodass die Applikationen die Schnittstelle in der richtigen Reihenfolge mit den richtigen Parametern aufrufen kann.

Ein großer Nachteil dieses Ansatzes ist die vergleichsweise starke Kopplung zwischen Client und Server. Anschließend kann die Schnittstelle kaum noch verändert werden, es sei denn, die Clients werden ebenfalls angepasst.

Die Interaktion zwischen Client und Server funktioniert durch Hypermedia grundsätzlich anders. Typischerweise veröffentlicht ein Hypermedia-API nur wenige Ressourcen – oft kennen die Clients nur einen einzigen Start-URI. Mit URIs werden in RESTful HTTP eindeutig Ressourcen identifiziert und durch verschiedene Medientypen repräsentiert. Teil einer Repräsentation sind nicht nur die Daten der Ressource, sondern auch die zuvor beschriebenen Hypermedia-Steuerelemente, bestehend aus Links und Steuerungsdaten.

Listing 2: Siren-Beispiel

```
{
  "class": [ "user" ],
  "properties": {
    "id": 1,
    "name": "...",
    "status": "..."
  },
  "entities": [
    {
      "class": [ "items", "collection" ],
      "rel": [ "http://host:port/user-roles" ],
      "href": [ "http://api.host.port/users/u1/
      roles" ]
    }
  ],
  "links": [
    { "rel": ["self"], "href": "http://
    api.host:port/user/u1" }
  ],
  "actions": [
    {
      "name": "add-role",
      "title": "Add a user role",
      "method": "POST",
      "href": "http://api.host:port/user/u1/
      roles"
      "type": "application/
      x-www-form-urlencoded",
      "fields": [
        { "name": "roleName", class="role",
        "type": "text" }
      ]
    }
  ]
}
```

Mit diesen Links gibt der Server den Clients mögliche Navigationsoptionen vor. Das bedeutet, dass nicht der Client bestimmt, welche Operationen als Nächstes folgen können, sondern der Server macht dies in Form der Links. Die Ressourcen des API können selbst entscheiden, welche Folgezustände möglich und erlaubt sind. Auf diese Weise ist das API viel flexibler und die Kopplung zwischen Client und Server ist nicht so stark.

Hypermedia-Debatte

Der Einsatz von Hypermedia wird kontrovers diskutiert. Während Dr. Roy Fielding die Meinung vertritt, dass Hypermedia ein integraler Bestandteil des REST-Architekturstils sei, argumentieren Hypermedia-Skeptiker, dass Hypermedia APIs aufblasen und schwer nutzbar machen. Nichtsdestotrotz scheinen die Vorteile die Nachteile zu überwiegen, denn viele moderne Frameworks zur Entwicklung von APIs, wie beispielsweise Spring HATEOS und Jersey, unterstützen Hypermedia-Formate wie HAL, JSON-LD, JSON API, Siren oder Collection+JSON.

In der Tat müssen für Hypermedia mehr Daten übertragen werden, aber gemessen an der Gesamtgröße der Nachrichten ist dieser Overhead vernachlässigbar. Ein Hypermedia-API macht außerdem mehr Arbeit, denn die Links zwischen den Ressourcen müssen beim Design wohlüberlegt ausgewählt werden. Was kann man alles mit einer Collection-Ressource machen und was mit einem einzelnen Item? Welche Ressourcen arbeiten zusammen? Das sind allerdings alles Überlegungen, die sicherlich nicht zu einem schlechteren Entwurf führen. Es ist auch möglich, mit einer minimalen Verlinkung zu beginnen und inkrementell weitere hinzuzufügen.

Hypermedia eignet sich daher prinzipiell für alle Anwendungsfälle, die allgemein für RESTful HTTP geeignet sind. RESTful HTTP ist wiederum für nahezu jedes Web-API eine Option. Lediglich für verlässliche Nachrichtenübermittlung (SOAP WS-Reliable Messaging), zustandsbehaftete Dienste und APIs mit formalen Verträgen ist RESTful HTTP ungeeignet. Man kann jedoch darüber diskutieren, ob derartige Features tatsächlich notwendig sind (www.infoq.com/articles/no-reliable-messaging).

Siren – ein standardisierter Medientyp

Ein Web-API mit registrierten Medientypen kann leichter konsumiert werden, weil die Entwickler, die später das API benutzen werden, die Medientypen höchstwahrscheinlich schon kennen. In diesem Fall wird die Einarbeitung einfacher. Für bekannte Medientypen findet man in der Regel auch zahlreiche Erklärungen und Beispiele im Web (Bild 2).

Ein registrierter Medientyp speziell für Hypermedia-APIs ist Siren (https://github.com/kevinswiber/siren). Dieser von Kevin Swiber entwickelte Medientyp basiert auf dem Konzept der Entitäten. Eine Entität entspricht einer Ressource, die durch einen URI identifiziert wird. Die Entität kann Sub-Entitäten und Navigationslinks enthalten.

Das Siren-Beispiel in Listing 2 zeigt die Repräsentation einer User-Entität. Der Typ der Entität wird mit dem Array ▶

class beschrieben. Die Daten der Entität werden in einem flexiblen JSON-Objekt *properties* mit Schlüssel-Wert-Paaren angegeben.

Beziehungen zu anderen Entitäten werden mit Sub-Entitäten angegeben. Die Sub-Entitäten sind im Array *entities* aufgelistet. Die Sub-Entitäten werden entweder mit *href* referenziert oder vollständig eingebettet. Hier ist im Einzelfall ein Trade-off zwischen Nachrichtengröße und der Anzahl zusätzlicher Abfragen zu treffen.

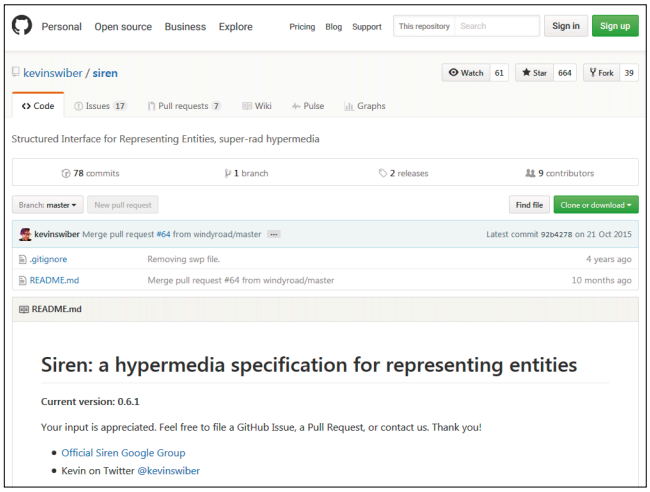
Mobile Applikationen haben häufig Verbindungen mit hoher Latenz. Jeder zusätzliche Roundtrip kann deshalb die Performance der Client-Applikation spürbar beeinträchtigen. In diesem Fall würde man daher häufig verwendete Ressourcen einbetten und nicht referenzieren. Ein ähnlicher Trade-off muss auch beim Lazy und Eager Loading von persistenten Entitäten aus Datenbanken getroffen werden. Typischerweise will man genau die Daten laden, die für eine Unit of Work gebraucht werden.

Welche Entitäten dazugehören, kann man auch nachträglich durch Analysen des HTTP-Logs herausfinden. Solche Details lassen sich später noch ändern. Clients müssen mit eingebetteten und mit referenzierten Sub-Entitäten umgehen können und dürfen keine speziellen Annahmen treffen.

Die Semantik der Beziehungen zwischen Entitäten und Sub-Entitäten wird durch das Array *rel* definiert.

Die darin enthaltenen URIs identifizieren die spezifischen Relationstypen. Falls möglich, sollte man jedoch auf registrierte Relationstypen wie *next, self* und *current* zurückgreifen, um möglichst wenig neue Konzepte definieren zu müssen. Mit den Relationstypen wird auch die Semantik der Navigationslinks beschrieben. Das allgemeine Web Linking ist in RFC 5988 definiert. Auch der Medientyp der verlinkten Sub-Entitäten kann optional mit dem Attribut *type* angegeben werden.

Eine Besonderheit von Siren sind die Actions. Diese Templates beschreiben die Operationen, die die Ressource dem Client anbietet. Die Action in Siren-Beispiel-Listing zeigt, dass mit *HTTP POST*, dem angegebenen URI und dem Parameter *roleName* jeweils eine Rolle hinzugefügt wird.



Siren ist ein registrierter Medientyp speziell für Hypermedia-APIs (Bild 2)

Die Relationstypen sollte man nicht mit den Entitätstypen, die mit dem Array *class* angegeben werden, verwechseln. Laut der Siren-Spezifikation sind diese Werte abhängig von der Implementierung und müssen daher dokumentiert werden. Leider gibt die Spezifikation nicht vor, wie dies zu erfolgen hat.

Die unterstützten Rollen könnten beispielsweise in einem JSON-Schema als Enumeration definiert werden. Alternativ könnte man ein Profil auf Basis von ALPS zur Angabe der applikationsspezifischen Semantik nutzen. ALPS wird beispielsweise von Spring Data REST unterstützt.

Fazit

Hypermedia-APIs sind eine vielversprechende Alternative für RPC-artige Web-APIs. Die Selbstbeschreibung der APIs kann deren Benutzung stark vereinfachen. Außerdem können mit Hypermedia-Links Beziehungen zwischen Ressourcen und Navigationspfade dynamisch repräsentiert werden. Ein großer Vorteil ist die geringere Kopplung zwischen Client und Server.

Hypermedia stellt jedoch höhere Anforderungen an Clients. Clients erhalten keine Liste mit erlaubten Abfragen, die in festgelegter Reihenfolge aufgerufen werden können. Stattdessen müssen die Clients viel intelligenter sein und flexibler auf das reagieren, was ihnen die Ressourcen anbieten. ■

Links zum Thema

- Hypermedia-Typen
  http://amundsen.com/hypermedia/hfactor
- A vocabulary and associated APIs for HTML and XHTML
  www.w3.org/TR/html5/forms.html
- Nobody Needs Reliable Messaging
  www.infoq.com/articles/no-reliable-messaging
- Siren: a hypermedia specification for representing entities
  https://github.com/kevinswiber/siren
- Unit of Work
  http://martinfowler.com/eaaCatalog/unitOfWork.html
- Web Linking
  http://tools.ietf.org/html/rfc5988

Kai Spichale
ist Software-Architekt bei der Adesso AG mit den Themenschwerpunkten Software-Architektur, API-Design und NoSQL. Er ist Autor zahlreicher Fachartikel und regelmäßiger Sprecher auf Konferenzen.
@kspichale

Foto: Fotolia / bakhtiarzein

INFRASTRUKTUR AUS DER PUBLIC CLOUD

Flexibel und skalierbar

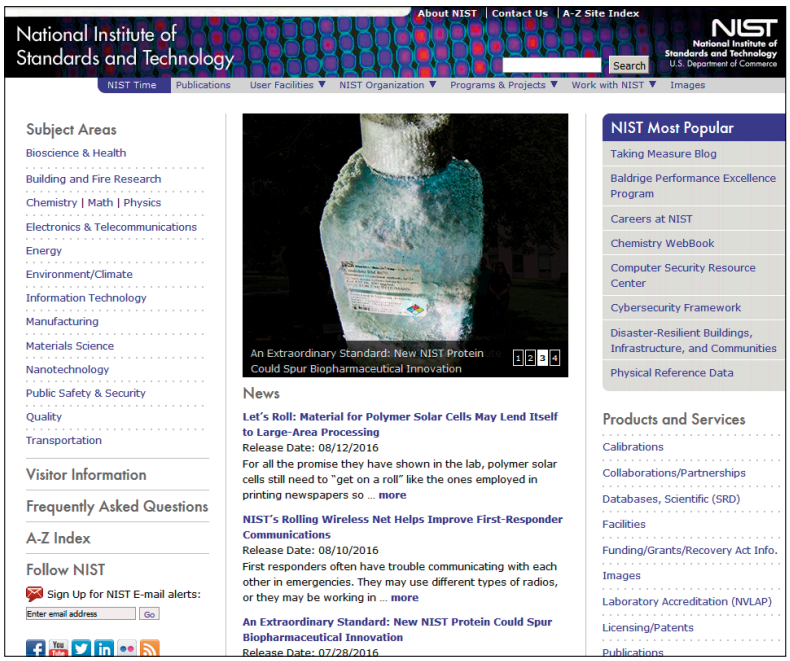
Infrastructure as a Service (IaaS) bildet die Basis aller Cloud-Computing-Modelle.

Ohne Rechenleistung, Arbeitsspeicher, Storage und Netzwerkkomponenten lässt sich schließlich auch im Zeitalter von Virtualisierung und Industrie 4.0 keine Software betreiben. Obwohl der Einstieg in die Cloud meist über Software as a Service (SaaS) erfolgt, investieren Unternehmen mittlerweile auch vermehrt in Cloud-Infrastruktur. Laut den Analysten von IDC haben sie im vergangenen Jahr weltweit 32,6 Milliarden Dollar und damit rund ein Drittel des gesamten Hardware-Budgets für IaaS ausgegeben. Bis 2019 soll fast die Hälfte der IT-Infrastrukturausgaben in die Cloud fließen. Vor allem Public-Cloud-Angebote legen zu. Sie sind laut IDC um gut 30 Prozent auf mehr als 20 Milliarden Dollar gewachsen, während die Private-Cloud-Investitionen nur um 15 Prozent anstiegen.

Der Trend zu IaaS hat inzwischen auch die deutsche Wirtschaft erreicht. Sie wird nach Angaben des Marktforschungsunternehmens Crisp Research 2018 etwa 2 Milliarden Euro allein in Public-Cloud-Infrastruktur-Ressourcen investieren. Diesen Trend bestätigt der Cloud Monitor des Branchenverbands Bitkom. Hatten 2012 gerade einmal 14 Prozent der Befragten IaaS aus einer Public Cloud im Einsatz, waren es 2014 bereits 53 Prozent. Im Bereich Private Cloud betrug der Anteil 2014 dagegen erst 34 Prozent.

Wozu eigentlich IaaS?

Belastbare Aussagen über den IaaS-Markt und die hier agierenden Anbieter sind nur möglich, wenn man sich darüber einigen kann, was Infrastructure as a Service überhaupt ist.



Vom National Institute of Standards and Technology (NIST) gibt es eine Definition, was IaaS ist (Bild 1)

Dies erscheint nur auf den ersten Blick einfach. Nach der allgemein akzeptierten Definition des National Institute of Standards and Technology (NIST) stellt IaaS dem Nutzer wesentliche Rechenressourcen zur Verfügung, auf denen dieser beliebige Betriebssysteme und Programme ausführen kann (Bild 1). Der Anwender hat dabei keine Kontrolle über die zugrunde liegende Cloud-Infrastruktur, sondern nur über Betriebssystem, Speicher, Applikationen und gegebenenfalls Netzwerkkomponenten wie Firewalls.

Ein gehosteter Server, bei dem der Kunde die Hardware selbst verwaltet, wäre also kein IaaS. Übernimmt dagegen der Hoster das Server-Management oder stellt er virtuelle Maschinen zur Verfügung, müsste man auch solche Plattformen nach der NIST-Definition als Infrastructure as a Service bezeichnen. Manche Hoster tun dies auch. Schon vor dem Cloud-Computing-Zeitalter habe man den Begriff IaaS benutzt, sagt beispielsweise Benjamin Schönfeld, Managing Director bei LeaseWeb Deutschland: »In der damaligen Zeit bezeichnete man mit IaaS schlichtweg das klassische Server-Hosting, also letztlich eine Hardware-Infrastruktur, die als Service zur Verfügung gestellt wird.«

Heute liegt die Trennlinie zwischen Server-Hosting und IaaS weniger in der Technik als im Bezahlmodell, auch wenn dieses gar kein Bestandteil der NIST-Definition ist. Server vom Hoster – egal ob Bare Metal, also echte Hardware, oder virtuelle Maschine – sind in der Regel langfristig zu mieten (Bild 2). Die Verträge laufen oft über zwölf oder 24 Monate und werden auf monatlicher Basis abgerechnet. IaaS im engeren Sinn setzt dagegen auf maximale Flexibilität. »Rechen-, Speicher- und Netzwerkkapazitäten werden nur bei Bedarf (…) bereitgestellt und nutzungsabhängig abgerechnet, teilweise in sehr kleinen Einheiten, etwa pro Stunde«, sagt Sven Klindworth, der als Head of Advise Compute bei BT das Beratungs-Team für IT-Services und Cloud-Lösungen leitet.

»Damit werden IT-Leistungen so flexibel nutzbar wie Wasser und Strom, und ganz neue Geschäftsmodelle werden möglich, bei denen sich allerdings immer der Nutzer um die Anwendung auf der bereitgestellten Infrastruktur kümmern muss«, so Klindworth weiter. »Die Ressourcen werden nur für die gerade benötigte Zeit gemietet, im Idealfall gibt es keine vertragliche Bindung«, ergänzt Matthias Schorer, Head of Strategy Consulting CEMEA bei VMware.

Die Flexibilität lassen sich die Provider allerdings meist mit einem Aufschlag bezahlen. Sie wollen Planungssicherheit und gewähren deshalb deutliche Nachlässe, wenn sich der Kunde zu einer monatlichen oder gar jährlichen Abnahme bestimmter Kontingente verpflichtet. Ein weiteres Merkmal, das IaaS zumindest vom ganz klassischen Server-Hosting unterscheidet: Es gibt keine direkte Zuordnung der Rechenleistung zu dedizierter Hardware. »Infrastructure as a Service bietet zwar letztlich auch Zugang zu Server-Hardware, aber nur durch eine Schnittstelle, ein API«, sagt Giri Fox, Director of Customer Technology Services bei Rackspace.

Die Grenzen zwischen Server-Hosting und IaaS sind fließend (Bild 2)

Den richtigen Anbieter wählen

Diese Abstraktion von der Hardware macht Vergleiche zwischen den Plattformen schwierig. Jeder Anbieter kocht sein eigenes Süppchen, was Leistungsstufen, Preismodelle, Rabatte und Pakete angeht. Um das Preis-Leistungs-Verhältnis verschiedener IaaS-Angebote aus der Public Cloud zu evaluieren, bleibt dem Anwender nichts anderes übrig, als Testläufe mit einer typischen Anwen- ▶

dungsumgebung zu fahren. Er muss zudem eine ziemlich klare Vorstellung davon haben, welche Leistung er wie lange benötigt.

Erschwerend kommt hinzu, dass die Performance nicht konstant ist, sondern unter anderem vom Auslastungsgrad der Hardware abhängt. Die meisten Provider überbuchen ihre Server, um die Auslastung zu optimieren. Wollen plötzlich doch alle Kunde gleichzeitig auf die überbuchte Maschine zugreifen, dann kommt es zu Leistungseinbrüchen. Selbst ein Benchmarking-Test mit einer definierten Anwendungsumgebung, wie ihn beispielsweise Crisp Research vorschlägt und anhand eines Beispiels auch durchexerziert, kann daher nur eine Momentaufnahme darstellen.

Das Preis-Leistungs-Verhältnis ist ohnehin nur ein – wenn auch wichtiger – Aspekt bei der Wahl eines IaaS-Providers. Meist soll die Rechenarbeit in der Cloud nicht isoliert vonstatten gehen, sondern eine nahtlose Ergänzung anderer Ressourcen darstellen, etwa klassischer Rechenzentrums-Infrastrukturen oder einer Private Cloud. Auf diese Anforderung weisen naturgemäß besonders diejenigen Hersteller gern hin, die wie VMware über viele Kundeninstallationen verfügen: »Kompatibilität mit der eigenen Infrastruktur sollte klar im Vordergrund stehen, sodass Services, die auf einer IaaS gehostet werden, jederzeit zwischen dem IaaS-Anbieter und dem eigenen Rechenzentrum verschoben werden können«, sagt VMware-Manager Schorer.

Noch ist der Safe-Harbor-Nachfolger Privacy Shield nicht in Kraft getreten (Bild 3)

Ein Unternehmen dürfe bei der Wahl einer Lösung aber nicht nur auf die Technologieanforderungen schauen, hält Rackspace-Manager Giri Fox dagegen: »Es sollte auch überlegen, wie sie entworfen wird, wer sie verwalten soll, wie sie mit neuen Funktionen aktualisiert wird und wer sich um Ausfälle kümmert, die um Mitternacht an einem Wochenende geschehen.«

Natürlich spielen die Themen Datensicherheit und Verfügbarkeit eine große Rolle bei der Wahl eines Providers. Allerdings schenken sich die großen Anbieter hier wenig. Die meisten halten zertifizierte Rechenzentren bereit, eine redundante Anbindung und Datenhaltung sowie Service Level Agreements (SLAs) mit mindestens 99,95 Prozent Uptime.

Wie weit man den SLAs der Provider trauen sollte, ist allerdings Ansichtssache. Es handle sich mehr um Service Level Announcements als Agreements, sagt René Büst, Senior Analyst und Cloud Practice Lead bei Crisp Research. Wer IaaS wie ein klassisches Outsourcing betrachte, sitze sowieso einem großen Missverständnis auf, schreibt er in einem Blog-Beitrag: »In der Public Cloud geht es um die Aufteilung von Verantwortlichkeiten – auch als Shared Responsibility bezeichnet. Anbieter und Kunde teilen sich die Aufgabenbereiche untereinander auf. Die Eigenverantwortung des Kunden spielt dabei eine zentrale Rolle.«

Eine Frage des Standorts

Nicht erst seit dem Safe-Harbor-Urteil im vergangenen Herbst spielt das Thema Datenschutz und Compliance ebenfalls eine bedeutende Rolle, sofern personenbezogene Daten auf den Servern in der Public Cloud gelagert und verarbeitet werden. Solange der Safe-Harbor-Nachfolger Privacy Shield nicht in Kraft getreten ist, muss auf jeden Fall eine Vereinba-

Leistungstest: Public-Cloud-IaaS

Das Marktforschungsunternehmen Crisp Research hat das Preis-Leistungs-Verhältnis von vier Public-Cloud-IaaS-Angeboten (Amazon Web Services, Google Cloud Platform, Microsoft Azure, ProfitBricks) verglichen. Als Modellumgebung diente ein typischer Webshop auf Basis der E-Commerce-Software Magento.

Mit Hilfe von BlazeMeter, einem SaaS-Angebot für Lasttests, simulierten die Analysten 50 parallel auf den Shop zugreifende Nutzer, die verschiedene Seiten aufriefen und Einkäufe tätigten. Als Maß für die Leistung diente die Zeit, bis die Webseiten sich aufgebaut hatten (Antwortzeit). Die Tests wurden in unregelmäßigen Abständen wiederholt, um ein möglichst realistisches Bild von der durchschnittlichen Leistungsfähigkeit des jeweiligen Angebots zu erhalten.

Die Antwortzeiten waren bei Google mit rund 6000 Millisekunden am geringsten, bei AWS mit fast 11.000 Millisekunden am höchsten. Google und Microsoft wiesen zudem wenig Schwankungen auf, während bei ProfitBricks und vor allem bei AWS die Antwortzeiten stark variierten.

Beim Preis gab es allerdings eine Überraschung: Nicht AWS war bei der gewählten Konfiguration am günstigsten, sondern ProfitBricks. Beim deutschen Provider kostete die Betriebsstunde nur 4 Cent, während es bei Google 6,2 Cent, bei AWS 6,8 Cent und bei Azure über 7 Cent waren. Mit einer minutengenauen Abrechnung bot ProfitBricks auch das fairste Preismodell. Dank der geringeren Kosten hatte ProfitBricks auch im Preis-Leistungs-Vergleich die Nase vorn, vor Google, Amazon und Microsoft.

rung zur Auftragsdatenverarbeitung nach den EU-Standardvertragsklauseln mit dem Cloud-Provider vorliegen (Bild 3). Die Daten sollten den EU-Raum nicht verlassen oder wenigstens in Rechenzentren verarbeitet werden, die in einem Land mit angemessenem Datenschutzniveau stehen, etwa in Kanada, der Schweiz oder in Argentinien.

Datenverteilung beschränken

Die meisten IaaS-Provider bieten entsprechende Regionen oder Zonen an, auf die der Kunde die Datenverteilung beschränken kann. Besonders weit geht dabei BT. Der britische Konzern betreibt zwanzig Cloud-Rechenzentren rund um den Globus, wobei auch eher exotische Standorte wie Südafrika, Saudi-Arabien, Kolumbien oder Mexiko zur Verfügung stehen.

Die Deutsche Telekom nutzt bei ihren IaaS-Angeboten in Sachen Datenschutz den Standortvorteil Deutschland (Bild 4)

Der Kunde kann individuell bestimmen, welche Daten in welchem Land lagern. »Ein Unternehmen kann also zum Beispiel festlegen, dass bestimmte Daten in Deutschland gespeichert werden sollen, andere hingegen in China, Indien, den USA oder Argentinien«, sagt BT-Manager Klindworth.

Auch bei Rechenzentrumsstandorten in Deutschland oder in anderen datenschutzrechtlich unbedenklichen Ländern bleibt bei US-amerikanischen Providern ein Restrisiko, da sie zur Herausgabe von Daten auch dann verpflichtet sind, wenn diese sich gar nicht in den USA befinden. Dies ist zumindest die Auffassung von US-Regierung und Geheimdiensten, gegen die sich zum Beispiel Microsoft gerade gerichtlich zu wehren versucht. »Microsoft legt beim Angebot von Public-Cloud-Diensten sehr hohen Wert auf Datensicherheit und Datenschutz«, betont Jürgen Dick, Cloud Platform Lead bei Microsoft Deutschland.

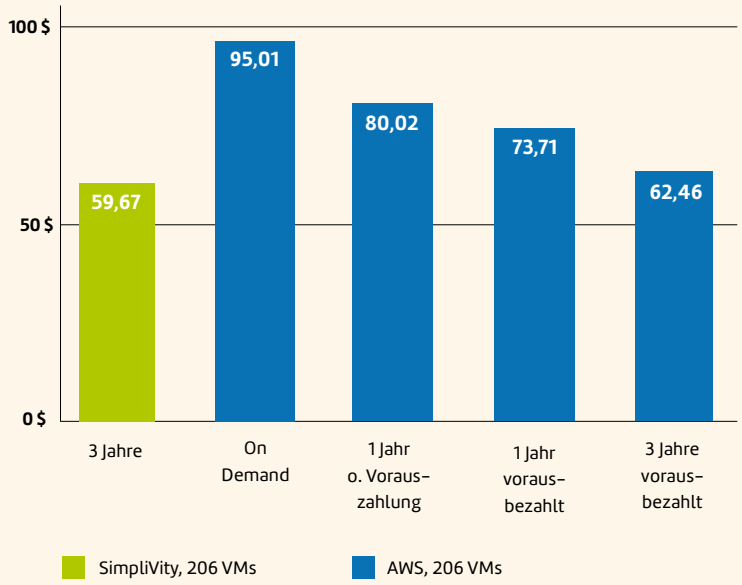
Mit der Microsoft Cloud Deutschland, bei der die Deutsche Telekom als Treuhänder den Zugang zu den Kundendaten kontrolliert, will das Unternehmen deutschen Anwendern Rechtssicherheit bieten und die Gefahr bannen, dass US-Behörden ohne Weiteres Zugriff auf personenbezogene Daten europäischer Kunden erhalten. Die Deutsche Telekom selbst nutzt mit ihrer Tochter T-Systems den Standortvorteil Deutschland gleich für mehrere IaaS-Angebote (Bild 4).

Mit der DSI vCloud bietet sie IaaS für die klassische VMware-Klientel, die DSI Intercloud soll vor allem als Cloud-Marktplatz dienen. Sie basiert wie die zur CeBIT gelaunchte Open Telekom Cloud auf der Open-Source-Cloud-Computing-Plattform OpenStack.

Wann sich IaaS (nicht) lohnt

Cloud-Services gelten als unschlagbar günstig, wenn es um die Betrachtung der Gesamtkosten (Total Cost of Ownership, TCO) geht. Nach einer Untersuchung, die die Experton Group im Auftrag von Claranet durchgeführt hat, liegen die Einsparungen über alle Einsatzszenarien hinweg bei durchschnittlich 25 Prozent gegenüber dem Betrieb im eigenen Rechenzentrum. ▶

Beispielrechnung



Kosten: Nach Berechnungen der Evaluator Group ist eine hyperkonvergente Infrastruktur im eigenen Rechenzentrum – beispielsweise von SimpliVity – günstiger als IaaS aus der Amazon-Cloud

web & mobile developer 10/2016 Quelle: Evaluator Group

Interview

»Wirklich erfolgreiche Unternehmen bieten Plattformen an«

René Büst, Senior Analyst und Cloud Practice Lead bei Crisp Research, erklärt im Gespräch mit **web & mobile developer**, was der eigentliche Mehrwert von Infrastructure as a Service ist und warum man in der Public Cloud durchaus auch geschäftskritische Anwendungen betreiben kann.

René Büst
Cloud Practice Lead
www.crisp-research.com

web & mobile developer: *Welche Rolle spielt Infrastructure as a Service aktuell im Cloud-Markt, und wie wird sich die Bedeutung von IaaS in Zukunft verändern?*
René Büst: Der Markt zieht an, das sieht man beispielsweise ganz deutlich an dem enormen Wachstum der Amazon Web Services. Auch von Microsoft wissen wir, dass die Nachfrage global gesehen immer mehr zunimmt. Ich habe erst vor Kurzem mit Dienstleistern gesprochen, die in der Technical Preview für das deutsche Azure Data Center sind. Das läuft anscheinend ganz gut an.

»Amazon hat vergangenes Jahr 722 neue Services und Funktionen bei AWS eingeführt. (...) Zeigen Sie mir ein Unternehmen oder einen Private-Cloud-Hoster, der dies leisten kann.«

web & mobile developer: *Wie kann man sich als Anbieter solcher Basisdienste anders von den Mitbewerbern unterscheiden als über den Preis?*
Büst: Anbieter, die sich auf reine Infrastrukturleistungen beschränken, werden vom Markt verschwinden. Nicht umsonst konzentrieren sich die wirklich erfolgreichen Unternehmen wie Amazon und Microsoft nicht mehr nur auf Server, Storage und Network, sondern bieten Plattformen an.
Amazon macht zwar noch recht viel Umsatz mit virtuellen Maschinen auf EC2 und dem Speicherdienst S3, um das Infrastrukturangebot herum ist aber im Lauf der Zeit ein riesiges Portfolio an Services entstanden. Das ist der eigentliche Mehrwert für die Anwender. Nicht umsonst tun sich Unternehmen mit Angeboten schwer, die diesen Mehrwert nicht bieten, etwa VMware mit vCloud Air oder auch IBM mit SoftLayer.

web & mobile developer: *Infrastruktur aus der Cloud ist nicht notwendigerweise kostengünstiger, als Server im eigenen Rechenzentrum zu betreiben, das haben mehrere Analystenhäuser errechnet. Gibt es auch Ihrer Ansicht nach Fälle, in denen eine eigene Infrastruktur sinnvoller und günstiger ist als IaaS?*
Büst: IaaS lohnt sich immer dann, wenn sich meine Auslastung stark verändert, sei es durch unvorhersehbare Ereignisse, sei es durch bekannte Peaks wie etwa im Weihnachtsgeschäft. Wenn ich dagegen konstante Workloads habe, dann ist diese Skalierbarkeit unwichtig und unnötig.
Man darf aber nicht vergessen, dass man Kapital braucht und erst einmal in die Infrastruktur investieren muss, wenn man zum Beispiel eine Private Cloud aufbauen möchte. Um Wartung und Erneuerung der Infrastruktur muss man sich ebenfalls selbst kümmern. Bei IaaS fällt das alles weg. Der Provider stellt die physikalischen Grundlagen ebenso zur Verfügung wie die Software, die man zur Bereitstellung der Dienste benötigt.
Vergessen Sie auch nicht die Innovationen, die aus der Public Cloud kommen. Die Anbieter tauschen die Hardware darunter permanent aus, das bekommt man als Anwender nur gar nicht mit. Das müsste man als Private-Cloud-Betreiber alles selbst machen, und das fehlt in diesen Diskussionen einfach.

web & mobile developer: *Wäre nicht eine Managed oder Hosted Private Cloud der goldene Mittelweg zwischen der kompletten Eigenverantwortung und der Abhängigkeit von einem Public-Cloud-Betreiber?*

»Die Rechenleistung kommt zu den Daten und nicht umgekehrt.«

Büst: Amazon hat im vergangenen Jahr 722 neue Services und Funktionen bei AWS eingeführt. Microsoft Azure spricht von ähnlichen Zahlen. Zeigen Sie mir ein Unternehmen oder einen Private-Cloud-Hoster, der dies leisten kann. In einer Managed Private Cloud ist das einfach nicht möglich.

web & mobile developer: *Neben den großen Anbietern gibt es immer wieder Versuche, Marktplätze für IaaS zu etablieren, allerdings mit wenig Erfolg. Erst kürzlich ist die Deutsche Börse mit ihrer Cloud-Broker-Plattform »Deutsche Börse Cloud Exchange« (DBCE) gescheitert. Woran liegt das?*
Büst: Es war für die Marktphase viel zu früh und die Leute haben nicht verstanden, was sie damit machen sollen. Die Plattform hat außerdem nicht die kritische Masse an attraktiven An-

bietern auf der einen und genügend Kunden auf der anderen Seite erreicht, um sie wirtschaftlich betreiben zu können. Schließlich ist das Modell an sich problematisch. Die Deutsche Börse wollte ja das tun, was sie am besten kann: makeln. Der Kunde schließt also so etwas wie eine Wette auf die Infrastruktur ab. Wer will das schon, wenn er geschäftskritische Anwendungen darauf betreiben will?

web & mobile developer: *Da ist zu fragen, welcher Anwender denn überhaupt seine geschäftskritischen Anwendungen in eine Public Cloud gibt.*
Büst: Es gibt genügend Beispiele. Netflix etwa nutzt ausschließlich AWS für seine Dienste und da ist das System nun wirklich geschäftskritisch. Das Unternehmen hat auf einer Public-Cloud-Infrastruktur ein massiv skalierbares und hochverfügbares System aufgebaut. Der Service läuft sogar weiter recht stabil, wenn eine ganze AWS-Region ausfällt.
Das ist es, was wir auch immer wieder versuchen, den Kunden klarzumachen: Auf einer Public Cloud sind sie zu 100 Prozent für die Daten und die Applikationen selbst verantwortlich. Man muss dafür sorgen, dass die Software-Architektur die Eigenschaften der jeweiligen Public Cloud versteht und auch beherrscht.

web & mobile developer: *Könnte man also jede Applikation in der Public Cloud betreiben?*
Büst: Ja, technisch ist das absolut möglich. Man muss sich einfach nur im Klaren sein, dass man sehr viel Eigenverantwortung hat. Oder man überträgt diese Verantwortung an einen Dienstleister.

web & mobile developer: *Fehlt in Deutschland nicht auch das Vertrauen in Public-Cloud-Angebote, vor allem wenn sie von amerikanischen Anbietern wie Amazon oder Microsoft kommen?*
Büst: Versetzen Sie sich doch einmal in die Lage eines IT-Entscheiders. Warum wählt er eine bestimmte Lösung? Doch nicht, weil sie Vertrauen einflößt, sondern weil sie ihm einen Mehrwert bietet.
Vertrauen ist die Basis und kein Verkaufsargument. Aus diesem Grund halte ich auch nicht besonders viel von Zertifikaten wie Trusted Cloud, schon gar nicht, wenn sich die Anbieter selbst zertifizieren.

web & mobile developer: *Rechtlich befindet man sich allerdings in einer Grauzone, wenn man personenbezogene Daten bei einem amerikanischen Public-Cloud-Betreiber lagert. Sollte man vorsorglich alles clientseitig verschlüsseln?*
Büst: Verschlüsselung ist absolut notwendig! Es gibt aber noch weitere interessante Ansätze. NetApp beispielsweise bietet mit »Private Storage für Amazon Web Services« die Möglichkeit, über eine Direktverbindung AWS für die Berechnungen zu nutzen, ohne dass die Daten in die Public Cloud wandern. Die Rechenleistung kommt zu den Daten und nicht umgekehrt.

Betrachtet man nur die Infrastruktur, beträgt das Einsparpotenzial sogar 30 Prozent. Dieses gern gezeichnete Bild von der kostengünstigen Wolke hat in letzter Zeit allerdings Risse bekommen. So ergab eine Studie des Marktforschungs- und Beratungsunternehmens Information Services Group (ISG), dass sich IaaS aus der Public Cloud nur dann rechnet, wenn man die Instanzen wenig nutzt. Bei einem Auslastungsgrad von 55 Prozent und mehr ist die interne IT günstiger.

Die Evaluation Group hat im Auftrag des Hyperconverged-Infrastructure-Anbieters SimpliVity den Vergleich »Cloud versus eigene (hyperkonvergente) Infrastruktur« einmal an einem konkreten Beispiel durchgerechnet. 206 virtuelle Maschinen bei Amazon Web Services mussten gegen dieselbe Anzahl VMs auf einem hyperkonvergenten OmniCube-System von SimpliVity antreten. Die Inhouse-Lösung schlug das AWS-Angebot in der TCO-Betrachtung über drei Jahre deutlich, selbst wenn die von Amazon bei Langzeitverträgen gewährten Rabatte miteingerechnet wurden. Der Unterschied wurde um so größer, je mehr Maschinen in die Berechnung einflossen.

Natürlich hinken solche Vergleiche. Über drei Jahre hinweg konstante Workloads sind sicher nicht der Haupteinsatzzweck von IaaS in der Public Cloud. Das Kernmerkmal ist ja gerade deren Flexibilität und Skalierbarkeit. Sie sind ideal, um Lastspitzen abzufangen oder schnell eine Testumgebung aufzusetzen. Ohnehin stelle sich die Frage des Entweder-oder immer seltener, sagt Benjamin Schönfeld von LeaseWeb: »Viele Unternehmen setzen auf hybride Ansätze, bei denen sowohl die klassische Hardware-Nutzung für Bestandsapplikationen fortgeführt wird als auch für neue Anwendungen eine IaaS- oder sogar eine SaaS-Lösung gefunden wird.«

Das findet auch Microsoft-Manager Dick: »In vielen Fällen empfiehlt es sich, eine hybride Infrastruktur zu implementieren, um so die Integration der Services im eigenen Rechenzentrum mit Services aus der Public Cloud oder Hosted Private Cloud zu gewährleisten.« Das Unternehmen kündigte erst vor Kurzem mit Azure Stack eine Lösung an, die den konsistenten Aufbau einer hybriden Infrastruktur erlauben soll.

IaaS – ein Auslaufmodell?

Anbieter von Infrastructure as a Service müssen sich nicht nur gegeneinander, sondern auch noch gegen die Konkurrenz aus den unternehmenseigenen Rechenzentren positionieren – angesichts kaum vorhandener Unterscheidungsmöglichkeiten ein fast unmögliches Unterfangen. »Differenzierungsmerkmale wie zum Beispiel Bereitstellungszeit, Ressourcenverfügbarkeit, Standardisierung verfügbarer Speichermethoden oder Auswahl an unterstützten Betriebssystemen haben sich bei den großen Anbietern stark angeglichen«, erklärt Jürgen Dick von Microsoft.

Es verwundert deshalb auch nicht, dass die IaaS-Provider längst mehr als reine Infrastrukturdienstleistungen bereitstellen. Bestes Beispiel ist AWS. Neben der Compute-Plattform Elastic Cloud (EC2) und dem Speicherdienst S3 gibt es eine Unmenge von mehr oder weniger infrastrukturnahen Diensten wie Datenbanken, Bausteine für mobile Applikationen und Internet of Things, Services für Business Intelligence ▶

und Big-Data-Analysen, Dokumentenverwaltung, virtuelle Desktops und E-Mail. Dazu kommen mehr als 2500 Applikationen von über 800 ISV-Partnern aus dem AWS Marketplace. »Wir glauben, dass es für unsere Kunden wenig Sinn ergibt, Cloud Computing in Kategorien wie IaaS, PaaS oder SaaS zu unterscheiden, da die meisten Kunden vielmehr daran interessiert sind, schnell und flexibel neue IT-Architekturen aus verschiedenen Komponenten aufzubauen, die unterschiedliche Aspekte von IT abdecken können, oder ihre bestehenden Systeme in die Cloud zu verlagern«, sagt Constantin Gonzalez, Principal Solutions Architect bei Amazon Web Services Germany Google. Microsoft oder IBM mit Bluemix setzen ebenfalls auf das Marktplatzkonzept, während Rackspace Service und Support in den Vordergrund stellt: »Die Spezialisten bei Rackspace helfen unseren Kunden bei der Nutzung der optimalen Technologien, ohne dass unsere Kunden wissen müssen, wie sie diese selbst betreiben können«, sagt Giri Fox, »IaaS-Angebote von Rackspace, AWS, Microsoft oder VMware bilden die darunterliegende Technologie.«

Fazit

Infrastructure as a Service (IaaS) aus der Public Cloud ist immer verfügbar, schnell eingerichtet und sehr flexibel. Ob sie kostengünstiger ist als die eigene Infrastruktur oder nicht, hängt vom Einsatzzweck sowie den Rahmenbedingungen ab – und auch davon, welche Kosten man wie in die Kalkulation einbezieht. Ein Preisvergleich hinkt ohnehin immer, denn mit der Rechenleistung in der Public Cloud kauft man sich in ein ganzes Ökosystem ein. Mehr oder weniger infrastrukturnahe Zusatz-Services lassen die Grenzen zwischen Infrastructure as a Service und Platform as a Service (PaaS) immer mehr verschwinden. Unter der Haube und oft vom Kunden unbemerkt arbeiten die Anbieter zudem beständig an der Erneuerung von Hard- und Software. Das muss man im eigenen Rechenzentrum alles selbst leisten.

Dennoch bedeutet Infrastructure as a Service in der Public Cloud nicht, dass man die Verantwortung komplett an den Cloud-Provider abgeben kann – im Gegenteil: Nutzung und Integration erfordern ausgeklügelte Verfügbarkeitskonzepte. »Die Migration jedes IT-Prozesses in die Cloud erfordert Erfahrung in Cloud-Entwicklung und Architekturunterstützung«, sagt Rackspace-Manager Giri Fox zu Recht. Wer selbst nicht über diese Erfahrung verfügt, sollte auf den Rat eines Managed Service Providers hören oder auf Cloud-Integration spezialisierte Systemhäuser zurate ziehen. ■

Anbieter von IaaS

- Amazon Web Services (AWS)
  www.aws.amazon.com/de
- BT Cloud Compute
  https://cloud.bt.com
- Google Cloud Platform
  https://cloud.google.com/compute
- HPE Helion Managed Cloud Services
  www.hpe.com/de/de/services/consulting/cloud.html
- IBM Cloud (SoftLayer)
  www.ibm.com/cloud-computing/de/de/infrastructure
- LeaseWeb
  www.leaseweb.com/de/cloud
- Microsoft Azure
  https://azure.microsoft.com/de-de
- ProfitBricks
  www.profitbricks.de
- Rackspace
  www.rackspace.com/de/cloud/servers
- Telekom DSI vCloud / DSI InterCloud / Open Telekom Cloud
  https://cloud.telekom.de/infrastruktur
- VMware vCloud Air
  http://vcloud.vmware.com

IaaS: Kriterien für die Anbieter-Auswahl

Für Infrastrukturdienstleistungen sind folgende Kriterien besonders wichtig:

- Wie schnell lassen sich Rechenleistung, Storage und Netzwerk einrichten, verwalten und skalieren?
- Kann ich das Angebot kostenlos testen?
- Passt das Abrechnungsmodell zu meinen Anforderungen?
- Wie ist die Anbindung? Gibt es verschiedene Möglichkeiten (Internet, MPLS, CDN, Direct Connect)?
- Wie nahtlos lässt sich das Cloud-Angebot in meine interne IT-Landschaft integrieren?
- Wo liegen die Daten? Wie sind die Rechenzentren gesichert?
- Für wie zuverlässig halte ich den Provider? Wird er langfristig am Markt bestehen?
- Welche Möglichkeiten habe ich, meine Daten und virtuellen Maschinen zurück ins eigene Rechenzentrum zu holen oder zu einem anderen Provider zu migrieren?
- Halte ich bei der Verwendung des IaaS-Angebots alle gesetzlichen Bestimmungen und Compliance-Richtlinien ein?
- Welche zusätzlichen Services bietet der Provider? Benötige ich diese oder muss ich sie nur mitbezahlen, ohne sie wirklich zu nutzen?
- Welche SLAs bietet der Provider? Welche Reaktionszeiten garantiert er?

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

dotnetpro Newsletter

Top-Informationen für den .NET-Entwickler.
Klicken. Lesen. Mitreden.

Jetzt kostenlos anmelden:

dotnetpro.de

twitter.com/dotnetpro_mag

facebook.de/dotnetpro

gplus.to/dotnetpro

RAUMPLANER-APPS UND -WEBSITES

Möbelrücken

Raumplaner-Apps und -Websites sollten vor allem funktional sein.

Beim Umzug in die neue Wohnung oder das neue Haus ist es sinnvoll, sich bereits vorab zu überlegen, wie vorhandene Möbel am besten in den neuen Räumen platziert werden können. Danach lässt sich recht gut einschätzen, ob bestimmte Einrichtungsgegenstände fehlen oder bisherige Möbel im neuen Heim überflüssig werden.

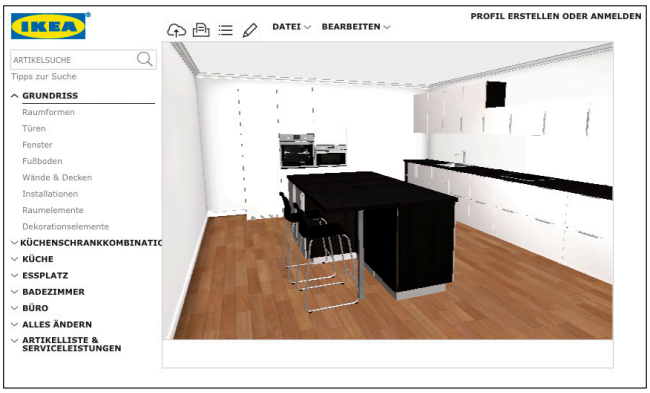
Mussten Umzügler früher noch kleine Papierquadrate auf dem handgezeichneten, mehr oder weniger maßstabsgetreuen Plan umherschieben, helfen heute verschiedene Websites. Hilfreich ist ebenfalls eine entsprechende App, mit der der Anwender auch vor Ort schnell das ein oder andere Möbelstück an den passenden Platz setzen kann.

Ein guter Raumplaner muss dabei verschiedene Anforderungen erfüllen: Er sollte leicht zu bedienen und optisch ansprechend sein. Dafür sollte er eine Fülle an verschiedenen Elementen bieten. Sinnvoll sind zudem unterschiedliche Darstellungen, etwa eine zweidimensionale zum Erstellen des Grundrisses und zum Verschieben der Möbel, sowie die 3D-Ansicht, um das Resultat zu überprüfen.

Bedienbarkeit

Nicht jeder, der umzieht, ist ein ausgefuchster Technikfreak. Wichtig ist also eine intuitive Bedienbarkeit – unabhängig davon, ob es sich um eine App oder eine Website handelt.

Wer plant, sich das nötige Equipment fürs neue Heim bei Ikea zu holen, sollte sich die verschiedenen Planer auf der Hersteller-Website ansehen. Vorteil: Die digitalen Möbel entsprechen in den Maßen und dem Aussehen dem erwarteten Ergebnis. Hier hatte sich bereits vor mehreren Jahren der Ikea Küchenplaner etabliert, heute gibt es weitere Planer, etwa für die unterschiedlichen Schranksysteme.

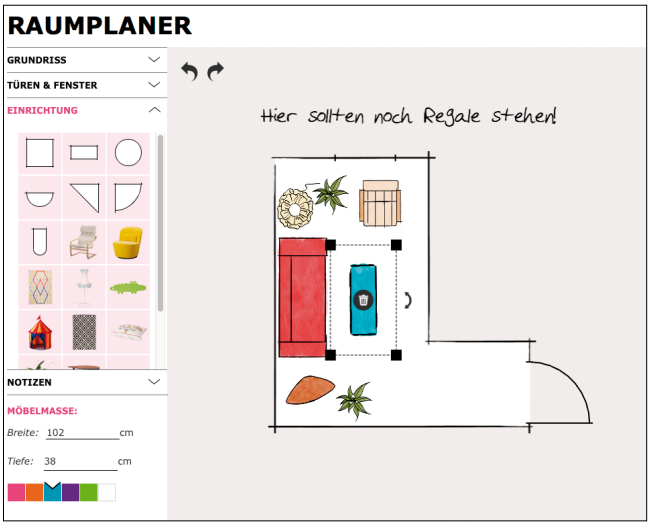


Nicht zum einfachen Loslegen geeignet: Der Ikea Küchenplaner zeigt sich etwas sperrig (Bild 1)

Zwar ist die Nutzung kostenlos, die Installation eines Plugins ist jedoch erforderlich. Diese erweist sich zwar für den Geübten als relativ problemlos, danach muss der geduldige Anwender jedoch noch ein Profil erstellen. Nun erscheint eine zweidimensionale Arbeitsfläche, auf der Grundriss, Türen und Fenster erstellt und die einzelnen Elemente aus dem Ikea-Katalog platziert werden. Alle Einrichtungsgegenstände landen nach einem Klick auf die jeweilige Vorschau direkt im Raum und müssen dann an ihren Platz geschoben werden – viel Spaß macht das nicht, da die Bedienung nicht gerade besonders flüssig läuft. Hier sollte seitens des Herstellers eine Überarbeitung angedacht werden (Bild 1).

Besser funktioniert der Hey Planer, eine modernere Variante von Ikea. Ganz unverbindlich, ohne die Notwendigkeit einer Anmeldung, startet sofort die Oberfläche zum Planen. Hier wird zunächst der Grundriss einer Zimmers festgelegt, mit Fenstern und Türen bestückt und dann möbliert. Anders als der Ikea Küchenplaner erweist sich diese Anwendung als intuitiv und schnell. Es entsteht eine zweidimensionale Ansicht mit bunten Möbeln, die sich in der Größe und Farbe ändern lassen. Dabei steht jedoch leider bei Weitem nicht das komplette Ikea-Sortiment zur Verfügung (Bild 2).

Dass die Design-Möglichkeiten eines Raumplaners nicht unter einer einfachen Bedienung leiden müssen, zeigt der Roomstyler von Autodesk – sobald auf der Site zunächst der entsprechende Link zum Werkzeug gefunden wurde. Dieser versteckt sich nämlich im dezent gehaltenen Menü *Tools* in einer Leiste am oberen Rand. Ist diese Hürde genommen, erwartet den Anwender ein gut ausgestattetes Sortiment an



Der Hey Planer ist ebenfalls von Ikea, jedoch sehr einfach in der Bedienung (Bild 2)

Grundrissen, Türen, Fenstern zusammen mit den passenden Materialien. Ebenfalls ein recht gut ausgerüsteter Katalog an Möbeln und Accessoires steht bereit.

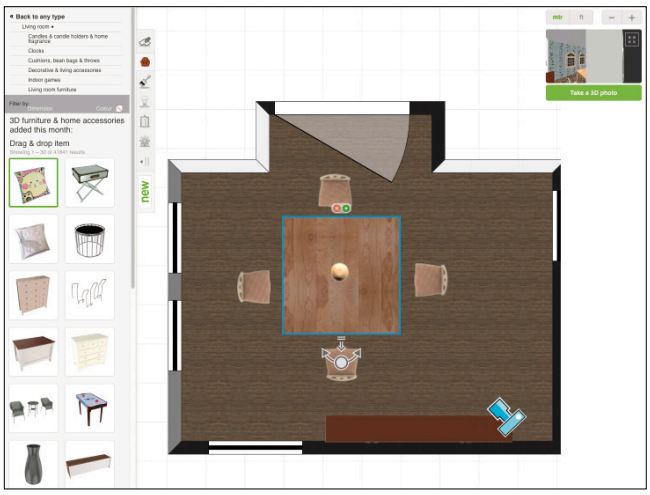
Zwar gibt es keine deutsche Sprachversion, die Bedienung ist jedoch trotz der Fülle ähnlich einfach einfach wie bei dem Hey Planer; Wände lassen sich intuitiv auf der Arbeitsfläche verschieben, die Raummaße werden dabei angezeigt. Eine recht gut bestückte Bibliothek liefert das Mobiliar für die einzelnen Räume. Platzierte Elemente können in der Größe an die individuellen Bedürfnisse angepasst werden, und unterschiedliche Materialien lassen sich mit einem Pinselklick auf Boden und Wände übertragen (Bild 3).

Bedienbarkeit und mobile Geräte

Die intuitive Bedienung eines Raumplaners für mobile Geräte schließt weitere Aspekte ein, etwa die Integration von Touch-Gesten. Hier punktet die App RoomSketcher: Nach dem Download der App steht eine bereits fertig eingerichtete Demo-Wohnung zum virtuellen Rundgang bereit. Hier gelangt der Anwender per Klick auf das Auge-Symbol in die Räume, in denen er sich über Wischgesten frei bewegen kann. Eine weitere Schaltfläche erlaubt eine animierte Rundumsicht von oben. Nette Spielerei: Wahlweise ein Mann oder eine Frau lassen sich durch die Räume führen (Bild 4). RoomSketcher gibt es für iOS- und Android-Geräte. Auch über das Internet lässt sich der Raumplaner nutzen.

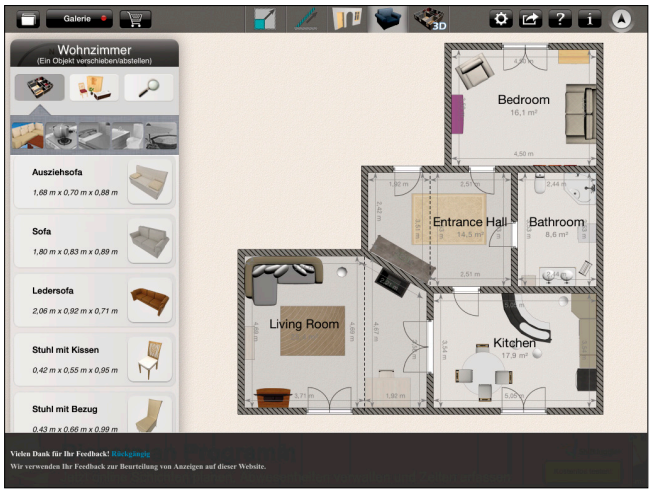
Die freie Version von Home Design 3D startet mit der 3D-Ansicht eines Raumes in Fotoqualität. Diese macht dann aber relativ schnell dem Angebot kostenpflichtiger Varianten der App Platz. Weiter geht es dann mit einer mehrseitigen Einführung: Hier wird das doch recht komplexe Programm gut bebildert erklärt. Wer die App zunächst kostenlos testen möchte, bekommt verschiedene Grundrisse zur Verfügung gestellt. Alle Wohnungen und Häuser sind bereits möbliert und liefern einen ersten Eindruck von den Möglichkeiten der App. Der Anwender kann einzelne Möbel löschen und neue in die Räume ziehen. Um einen eigenen Grundriss zu erstellen, ist es notwendig, zuvor alle Räume eines bereits vorhandenen Beispiels einzeln zu entfernen. Trotz der umfangreichen Funktionen ist die Anwendung recht gut zu bedienen. Eigene Grundrisse sowie eine genaue Platzierung und Skalierung der einzelnen Gegenstände gelingen im 2D-Modus, eine Vorstellung vom Ergebnis vermittelt die dreidimensionale Ansicht. Hier funktionieren auch Touch-Gesten. Zum Navigieren durch die einzelnen Räume stehen Pfeiltasten bereit. Die Anwendung gibt es sowohl im Google Play Store als auch im Apple Store (Bild 5).

Sehr hilfreich und dank der durchdachten Erklärungen gut bedienbar ist die App MagicPlan. Neben ganzen Wohnanlagen oder Geschäftsgebäuden kann auch die eigene Wohnung zunächst vermessen und dann eingerichtet werden. Hierfür greift die Anwendung auf die integrierte Kamera des Smartphones oder Tablets zu, um nacheinander die Maße der einzelnen Räume zu digitalisieren. Ein Video zeigt Schritt für Schritt, wie es geht: Zuerst wird mit Hilfe mehrerer Fotos die Raumhöhe vermessen, dann die Grundfläche. Zum Schluss ▶



Wände, Türen und Fenster sowie die Einrichtung können im Roomstyler intuitiv auf die Arbeitsfläche gezogen und angepasst werden (Bild 3)

Die App RoomSketcher unterstützt Touch-Gesten in der 3D-Darstellung der zuvor modellierten Räume (Bild 4)



Eine gut sortierte Demo-Wohnung liefert dem Interessenten einen ersten Einblick in die App Home Design 3D (Bild 5)

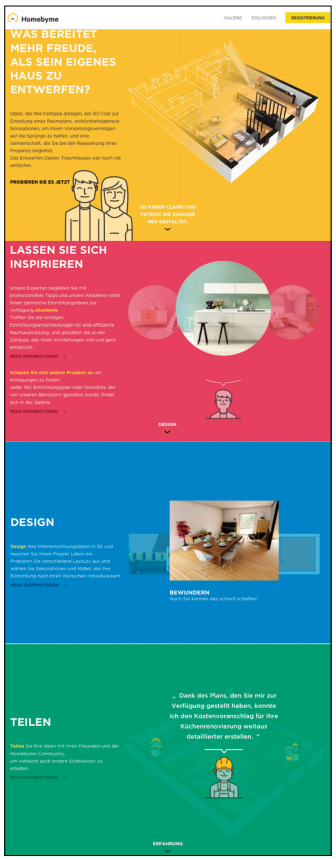
legt der Anwender durch weitere Aufnahmen die Position der Türen und Fenster fest. So entsteht, Raum für Raum, ein genauer Plan der Immobilie, die dann mit dem gewünschten Mobiliar gefüllt werden kann. Auch diese App steht für Android- und Apple-Geräte zur Verfügung (Bild 6).

Die App MagicPlan hilft dabei, einen exakten Grundriss des neuen Heims zu erstellen (Bild 6)

Grafische Umsetzung

Bei der grafischen Umsetzung von Apps und Webseiten zur Raumplanung gilt es zweierlei Dinge zu beachten. So sollte die Oberfläche mit den Bedienelementen wie Schaltflächen und Menüs möglichst einfach gehalten werden. Sinnvoll ist es, wenn Elemente zum Er- oder Einstellen der Räume und des Equipments gut sichtbar an einem Platz gesammelt sind. Meist liegen diese Elemente am linken Rand des Bildschirms – was eine funktionale und gute Bedienbarkeit durchaus unterstützt, da viele Benutzer diese Anordnung, gerade im Webbereich, gewohnt sind. Entscheidend ist zunächst das Layout des Startscreens. Dieser sollte den Nutzer in die Anwendung hineinführen, ihm sozusagen einen Vorgeschmack der angebotenen Möglichkeiten liefern und ihn anregen, ebenfalls genau mit dieser Lösung zu arbeiten.

Ein gutes Beispiel liefert die Webseite Homebyme. Hier landet der Interessierte auf einem übersichtlich gestalteten Onepager: Zuoberst steht auf gelben Grund die 3D-Ansicht einer Wohnung, bei der die Konstruktionsskizze einer Demo-Wohnung in eine Ansicht mit Texturen überblendet ist. Weiter geht es auf anderen Farbflächen mit inspirierenden Beispielen, die auf einer runden Bühne angeordnet sind, eine weitere Fläche zeigt Bilder der einzelnen Arbeitsschritte. Nun folgt der Hinweis auf die Homebyme-Community, auf der eigene Exponate mit anderen Anwendern geteilt werden können (Bild 7).

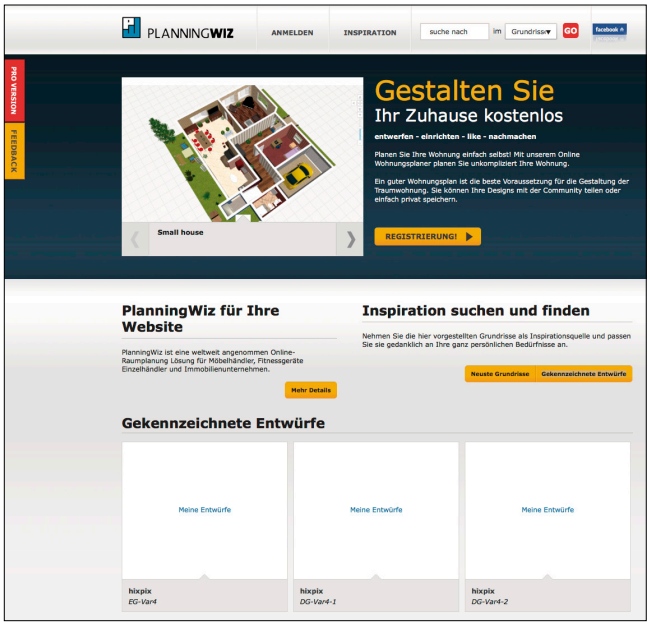


Das ansprechenden Layout der Startseite von Homebyme liefert alle notwendigen Informationen zum Online-Raumplaner (Bild 7)

Der PlanningWiz Raumplaner hingegen startet mit einer eher nüchtern anmutenden Seite. So ist im oberen Bereich eine sehr kleine und dadurch etwas nichtssagende 3D-Ansicht aus der Vogelperspektive zu sehen, rechts daneben befindet sich der Button zum Registrieren. Weiter fallen drei leere Flächen im unteren Drittel der Site ins Auge, auf denen der registrierte Nutzer die eigenen Entwürfe finden kann. Hier zeigt der Hersteller falsches Understatement: Der Online-Raumplaner besticht ansonsten durch seine Fülle an Funktionen, Vorlagen und Einrichtungsgegenständen (Bild 8).

Je größer die Auswahl an Mobiliar ist, desto individueller kann der Anwender seine neuen vier Wände gestalten. Dazu gehört auch, dass sich der digitale Grundriss optimal an die vorhandenen Gegebenheiten anpassen lässt.

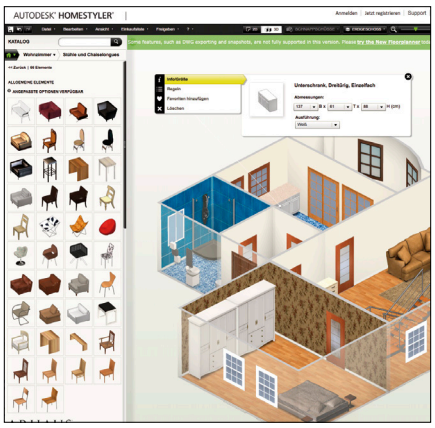
Die eigene Wohnung kann mit der Online-Anwendung Autodesk Homestyler entweder frei gezeichnet oder anhand eines zuvor geladenen Bildes des fertigen Grundrisses angelegt werden. Als Besonderheit sind hier auch gekrümmte Wände zu finden. Leider scheint es jedoch nicht möglich zu sein, einen Balkon anzulegen; dazu fehlen die notwendigen Elemente wie Balkongeländer. Ist der Grundriss fertig, folgen Fenster und Türen, danach die Wandgestaltung – hier wird man sehr positiv von einer recht großen Auswahl an Wandfarben, Kacheln und Fliesen überrascht; das Angebot an Tapeten und andere Materialien zur Gestaltung der Wände und Böden hingegen könnte etwa umfangreicher sein. Hilfreich



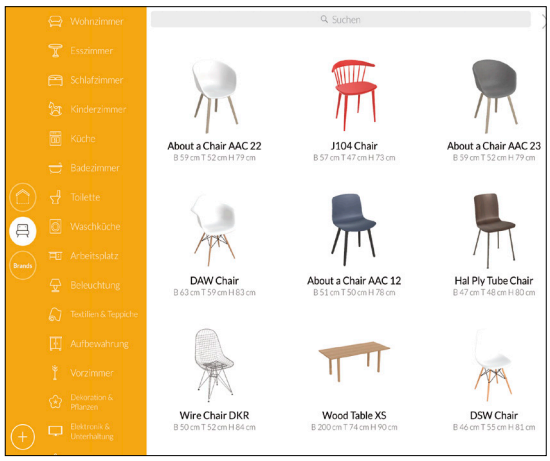
PlanningWiz ist eine hervorragende Lösung zum Planen der Räume, was die Startseite leider nicht ganz vermittelt (Bild 8)

beim Platzieren der Möbel ist die Möglichkeit, zwischen 2D- und 3D-Ansicht hin- und herzuschalten.

Der Homestyler zeigt sich funktional und liefert auch im Stil sehr unterschiedliche Einrichtungsgegenstände. So kann jeder in etwa das Mobiliar finden, das seinem individuellen Einrichtungsstil entspricht. Gerade für fotorealistische Renderings zeigt sich der Planer als echter Meister: Neben der detailreichen Inneneinrichtung kann ein Garten angelegt werden, der bei einem Blick durch das Fenster die Ansicht abrundet und so eine realistische Vorschau des neuen Heims liefert. Eine Einkaufsliste fasst zudem alle platzierten Gegenstände zusammen; sie ist jedoch auf den US-amerikanischen Markt abgestimmt und findet somit in Deutschland wohl lediglich als Gedankenstütze Verwendung (Bild 9).



Der Online-Raumplaner Homestyler liefert viele Materialien und erlaubt ausgezeichnete Renderings (Bild 9)



Die App Roomle stellt auch moderne Möbel bereit und liefert dazu Vorschläge für den Online-Kauf (Bild 10)

Ebenfalls sehr ansprechend in der Gestaltung ist die App Roomle für das iPhone/iPad: Eine erste Oberfläche zeigt mehrere kurze Einführungsvideos und verlangt lediglich eine einfache Registrierung. Danach kann – ohne auf eine Bestätigungs-E-Mail warten zu müssen – der erste Grundriss direkt mit dem Finger auf den Screen gezeichnet werden; die Maße werden eingeblendet. Ist der Raum geschlossen, weist eine kleine animierte Schaltfläche den Weg zu den Möbeln. Diese liegen in Ordnern am linken Rand der Fläche sortiert und zeigen sich, im Vergleich zum teils etwas angestaubten Mobiliar der Konkurrenten, erfrischend modern. Zudem können die einzelnen Stücke individuell gefärbt werden. Ist ein neues Stück gefunden, liefert die App auf Wunsch ähnliche Möbel verschiedener Hersteller mit Link und dem Preis.

Sind Tisch und Stuhl im Raum platziert, geht es ans Auspacken: Hier hat der Hersteller eine nette Spielerei eingebaut; per Mausklick gelangt der digitale Raumplaner beziehungsweise Käufer in die 3D-Ansicht, in der alle neuen Gegenstände zunächst in Umzugskisten stehen, die dann wie von Geisterhand geöffnet und entsorgt werden (Bild 10).

Fazit

Ob als Website oder App – digitale Raumplaner können hilfreich sein und zugleich Spaß machen. Die Lösungen sind sehr unterschiedlich und auf ebenso vielfältige Bedürfnisse zugeschnitten. So benötigt sicher nicht jeder ein fotorealistisches Rendering seiner zukünftigen Wohnung, einige hingegen werden gerade an solchen Angeboten viel Freude haben. ■

Links zum Thema

- Ikea
  www.ikea.com/ms/de_DE/campaigns/services/planer_und_ratgeber.html
- Hey Planer
  https://www.hej.de/raumplaner/neu
- Roomstyler
  https://roomstyler.com/3dplanner
- RoomSketcher
  www.roomsketcher.de
- Home Design 3D
  www.homedesign3d.net/EN
- MagicPlan
  www.sensopia.com/english
- Homebyme
  https://home.by.me/de
- PlanningWiz
  http://planningwiz.com/de/home.html
- Autodesk Homestyler
  http://de.homestyler.com/designer
- Roomle
  www.roomle.com
- Room Arranger
  www.roomarranger.com
  www.roomarranger.com/objects.html

Katharina Sckommodau
arbeitet als freiberufliche Autorin, Grafikerin und Dozentin unter anderem für die Akademie der Bayerischen Presse und für Macromedia. Sie veröffentlicht regelmäßig Beiträge in renommierten Fachzeitschriften und verwirklichte mehrere Buchprojekte.

SOFTWARE-DEFINED NETWORKING (SDN)

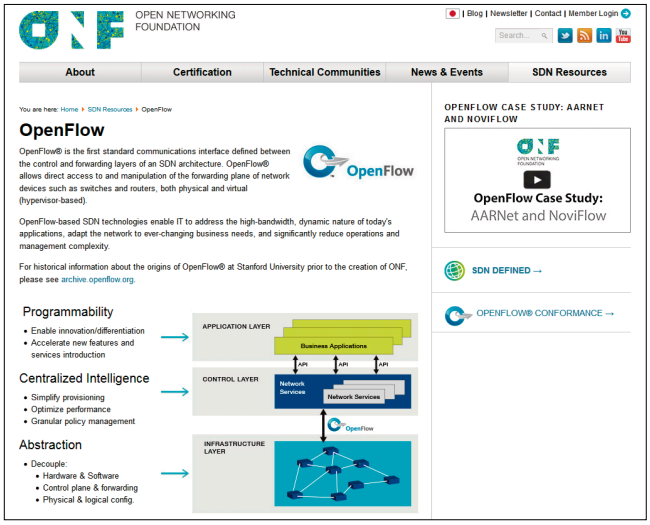
Intelligente Netzwerk-Topologien per Software

Klassische Netze lassen sich durch SDN-Technik flexibler, robuster und sicherer machen.

Kaum ein Bereich der IT hat eine derart unternehmenskritische Bedeutung wie die sichere und performante Vernetzung.

Es nützt nichts, den schnellsten Massenspeicher, die leistungsstärksten Server und die modernsten Anwendungen einzusetzen, wenn die Konnektivität mit dem wachsenden Datenvolumen kaum noch Schritt hält, auf Bedarfsfluktuationen nicht reagieren kann und erst recht nicht in der Lage ist, die Netzwerktopologie dynamisch zu verändern oder verteilte Denial-of-Service-Attacken (DDoS) aufzuspüren und abzuwehren, bevor die Firewall zusammenbricht.

Doch genau das ist die Realität der immer noch vielerorts vorherrschenden Netzwerktechnik. Bei der geringsten Betriebsstörung müssen Administratoren etliche Räume durchqueren, Racks aufschließen, Kabel manuell umstöpseln, Netzwerk-Hardware rebooten, IP-Adressen oder Subnetzmasken von Hand eingeben, denn ohne ein robustes Netzwerk kommt das Geschäft zum Stillstand. SDN soll hier Abhilfe schaffen.



OpenFlow ist ein Kommunikationsprotokoll, das Zugriff auf die Hardwarekomponenten eines Switches oder Routers liefert (Bild 1)

SDNs: Verschiedene Ansätze

SDN steht für Software-Defined Networking – zu Deutsch also softwaredefiniertes Netzwerk.

Wer sich mit der SDN-Praxis tiefergehend auseinandersetzt, stellt allerdings schnell fest, dass SDN lediglich ein Oberbegriff ist, der für eine Vielzahl von Ansätzen und Technologien verwendet wird – aber alle mit dem Zweck, das althergebrachte klassische Netzwerk umfassend zu modernisieren.

Die Implementierung eines SDN setzt den Einsatz geeigneter Steuerungsprotokolle voraus, die die Abstraktion des physischen Netzwerks mit einer fein abgestuften Kontrolle ermöglichen können. Die folgenden drei abgestuften Ansätze gibt es:

- **Netzwerkvirtualisierung (NV):** Dient der effizienteren Vernetzung virtueller Maschinen (VMs) durch das Tunneln von Datenströmen zwischen logischen Domains durch virtuelle Netzwerke. Das Neuverlegen von Kabeln entfällt dank der logischen Partitionierung des physischen Netzwerks. Anpassungen der Netzwerktopologie lassen sich leicht umsetzen,
- **Virtualisierung der Netzwerkfunktionen (NFV):** Abstrahiert die Funktionalität der Ebenen 4 bis 7 des OSI-Modells, darunter Firewalls oder IDS/IPS (Angriffserkennungs- und Abwehrsystem, Intrusion Detection and Prevention System) und Lastverteiler (Controller der Anwendungsbereitstellung) auf bestimmten Tunneln für ausgewählte Datenströme. Zu den wichtigsten NFV-Anbietern zählen unter anderem PLUMgrid und Embrane.
- **Softwaredefiniertes Netzwerk (SDN):** Trennt die Funktionen der Netzwerkkontrolle (auf der sogenannten Controller-Ebene) und die Datenweiterleitung (auf der Datenebene) voneinander, um eine saubere Steuerung der gesamten Netzwerkinfrastruktur per Software zu ermöglichen.

Marktüberblick

Die Netzwerkanbieter, von Alcatel-Lucent über Arista Networks, Brocade, Citrix, F5 bis hin zu Juniper und Riverbed, sind allesamt auf den SDN-Zug aufgesprungen. Cisco führt SDN-Lösungen im Rahmen der Produktreihen ONE, API und OpFlex. Auch Hersteller von Datencenter-Hardware – von HP über IBM bis hin zu Dell – wollen bei der Umrüstung von Netzen keinesfalls außen vor bleiben. Diese Anbieter integrieren in ihre sogenannten Black-Box-Switches erweiterte SDN-Funktionalität – meist auf der Basis proprietärer Software.

Generische White-Box-Switches sind das Mittel der Wahl, um die Kostenspirale in den Boden zu drücken. White-Box-Switches von Anbietern wie Accton Technology, Celestica und Quanta Computer basieren auf Betriebssystemen aus dem Hause Cumulus, Vello, Big Switch oder Pica8 und auf erschwinglicher Hardware mit Broadcom- oder Intel-Bauteilen. Sie ermöglichen den Aufbau von SDN-Overlays zu vergleichsweise geringen Anschaffungskosten und die Netzwerkorchestrierung mit minimaler Einarbeitung, bieten jedoch andererseits nicht die erweiterte Funktionalität von Marken-Hardware.

VMware ermöglicht mit der NSX-Netzwerk-Virtualisierungssoftware das Aufsetzen von Overlay-SDNs auf fast beliebiger Hardware.

Quelloffene Lösungen

Quelloffene Lösungen respektabler Standard-Gremien ließen nicht lange auf sich warten. In diese Kategorie fallen unter anderem OpenFlow und OpenStack (Bild 1). OpenDaylight täuscht diese Offenheit lediglich vor, denn hinter dieser Initiative stecken vor allem Cisco und IBM, einige ihrer Partner und auch ein paar neugierige Mitbewerber, die wissen wollen, was sich auf dem Gebiet tut.

Zu guter Letzt wollen auch innovative Start-ups wie ADARA, Big Switch, Embrane, Midokura, Plumgrid, Pluribus und andere im SDN-Markt Fuß fassen.

Mit der lobenswerten Ausnahme von OpenFlow existieren bisher keine nennenswerten offenen SDN-Standards. Zwar gibt es mit dem OpenDaylight-SDN-Konsortium einen scheinbaren Anwärter, aber die zwei Hauptakteure hinter der Initiative, Cisco und IBM, stellen die Unabhängigkeit ernsthaft infrage. Auch ein uniformer SDN-Ansatz existiert nicht. Jeder Anbieter kocht vorerst mehr oder weniger sein eigenes Süppchen.

Betroffene Interessenten sehen sich somit bei der Wahl einer Lösung einem unübersichtlichen und dynamischen Markt voller Tücken gegenüber.

»Es wäre gar nicht schwierig, sich auf einen Standard zu einigen«, bemerkt Robert Sherwood, CTO von Big Switch Networks und Vorsitzender der Architecture and Framework Working Group der Open Networking Foundation, die für die Entwicklung der Northbound-APIs verantwortlich zeichnet. »Wir könnten einfach sagen: Gut, dieser oder jener ist jetzt unser Standard, und schon sind wir fertig. Meine Sorge ist aber, dass sich ein Standard später entweder als komplett falsch oder als unvollständig entpuppen könnte. Am Ende kann das viel mehr Schaden anrichten, als vorerst überhaupt keine Wahl zu treffen.«

Zumindest anhand der verwendeten APIs lassen sich die SDN-Lösungen dennoch klassifizieren (Tabelle 1).

North- und Southbound-APIs

Ist von Software-Defined Networking die Rede, dann spielen immer zwei Fachbegriffe eine Rolle: Northbound-API und Southbound-API.

Unter einem Northbound-API versteht man eine Schnittstelle, die mit einer Komponente einer höheren Ebene des OSI-Modells (dem Southbound-API) kommunizieren kann. Zu den Funktionen des Northbound-API einer Netzwerk-Hardware-Komponente wie einem Router oder einem Switch zählt die Kommunikation mit Managementlösungen für die Automatisierung, die Orchestrierung und den Austausch von Daten zwischen verschiedenen Netzwerken.

Der Begriff Southbound-API bezeichnet eine Schnittstelle, die mit einer Komponente einer niedrigeren Ebene des OSI-Modells (dem Northbound-API) kommunizieren kann. Zu den Funktionen des Southbound-API einer Komponente wie einem Router oder einem Switch zählen die Kommunika- ▶

Tabelle 1: Netzwerktypen im Vergleich

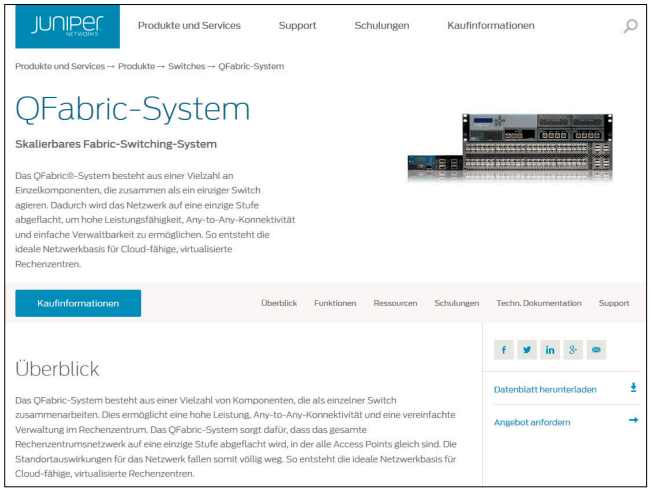
	Existierende Legacy-Netzwerke	Overlay-SDN	Underlay-SDN
Netzwerksicht	hardwaredominiert	softwaredominiert	hardwarezentrisch, softwaredominiert
Kontrolle über die Konfigurationsmodalitäten durch den Anwender	nicht gegeben (beschränkt durch den Anbieter der Hardware-Komponente)	gegeben (im Rahmen des bereitgestellten Funktionsumfangs)	gegeben (im Rahmen des bereitgestellten Funktionsumfangs)
Offenheit der Technologie	nicht gegeben (es sind nur geschlossene, starre Netzwerktopologien möglich)	vollständig	beschränkt (offen im Rahmen unterstützter Hardware)
Kohärenz des Netzes	nicht gegeben (die Kommunikation der Netzwerkgeräte miteinander erfolgt über unabhängige Protokolle)	vollständig (die Kommunikation der Netzwerkgeräte miteinander erfolgt über standardisierte Protokolle)	vollständig (die Kommunikation der Netzwerkgeräte miteinander erfolgt über standardisierte Protokolle)
Administrative Effizienz	niedrig	hoch	hoch
Kosten der Administration	hoch	niedrig	niedrig
Beispiele	Altlasten-Hardware	VMware NSX	Cisco ACI

tion mit dem Netzgewebe (Network Fabric) und die Funktionalität rund um die Handhabung von Netzwerkvisualisierungs-Protokollen und die Integration in ein verteiltes Netzwerk.

Genau genommen können die Begriffe Northbound und Southbound praktisch jede Art von Netzwerk oder Computersystem bezeichnen, sofern die betreffende Komponente Datenflüsse befördert. Dennoch kamen Northbound und Southbound in den vergangenen Jahren besonders im Zusammenhang mit APIs und SDNs in Mode. In Netzwerkdiagrammen hat sich die Konvention etabliert, einen Northbound-Datenfluss als von der betreffenden Netzwerkeinheit nach oben verlaufend und analog dazu einen Southbound-Datenfluss nach unten verlaufend zu visualisieren.

SDN-Varianten

Etablierte Anbieter von Netzwerk-Hardware fokussieren sich tendenziell auf Southbound-APIs, um eine enge Integration mit den eigenen Produkten zu gewährleisten. Bei solchen SDN-Lösungen spricht man von Underlay-SDNs. In diese Kategorie fallen unter anderem Cisco mit FabricPath und Juniper mit QFabric (Bild 2).

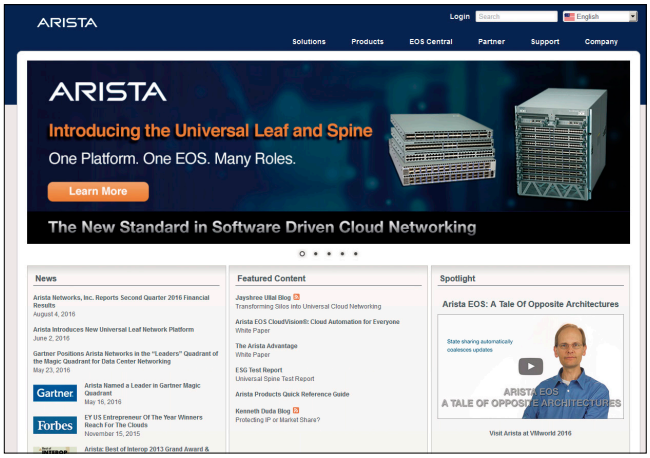


Juniper mit QFabric ist ein Vertreter der sogenannten Underlay-SDNs (Bild 2)

Stärken von Underlays

Zu den unverkennbaren Stärken von Underlays zählen die enge Integration und ihre hohe Performance. Diese Eigenschaften erkaufen sich ihre Anwender durch vergleichsweise hohe Anschaffungskosten und eine reduzierte Interoperabilität mit anderen Herstellern.

Ein Underlay-SDN birgt für den Anwender ein hohes Risiko des Vendor-Lock-ins, ließe sich allerdings durchaus auch hersteller- und controllerunabhängig implementieren – etwa bei der hybriden Arista-7000-Serie. Diese Implementierung kann jedoch bei der Integration des Software-Defined Network mit physischen Netzen auf der Basis von VXLAN Probleme bereiten. Der Virtualisierungspionier VMware legt mit



Arista Networks verfolgt mit dem Software-Driven-Networking-Konzept einen hybriden Ansatz bei SDN (Bild 3)

der eigenen Overlay-Technologie NSX den Schwerpunkt auf Northbound-APIs. Gleiches trifft auf Microsofts System Center, Junipers Contrail und die SDN-Technologie von Nuage Networks zu. Der wichtigste Vorteil von Overlays besteht in ihrer vergleichsweise hohen Interoperabilität mit vorhandener Hardware nahezu beliebiger Anbieter. Dieser weitgehend hardwareunabhängige Ansatz kann die Migration bestehender Netzwerke wesentlich erleichtern.

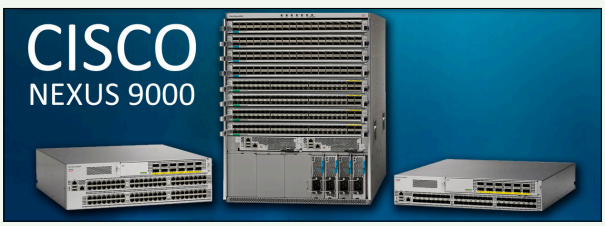
Reduzierte Leistung

Die Flexibilität dieser Lösungen fordert jedoch ihren Preis in Form einer etwas reduzierten Leistung aufgrund des Overheads und eingeschränkten Möglichkeiten der Diagnose von Hardware-Problemen.

Während die Verfechter der Underlay- und Overlay-SDNs ihren jeweils bevorzugten Ansatz für unschlagbar halten, hat

Das steckt hinter SDN

Bei Software-Defined Networking (SDN), einem softwaredefinierten Netzwerk, handelt es sich um einen Ansatz der Netzwerkvirtualisierung und Netzwerkautomatisierung, der die Steuerung der Netzwerkkonfiguration von den Aktivitäten rund um das Weiterleiten der Datenpakete trennt, um auf diese Weise flexible, anpassungsfähige und intelligente Netzwerktopologien durch Software zu ermöglichen.

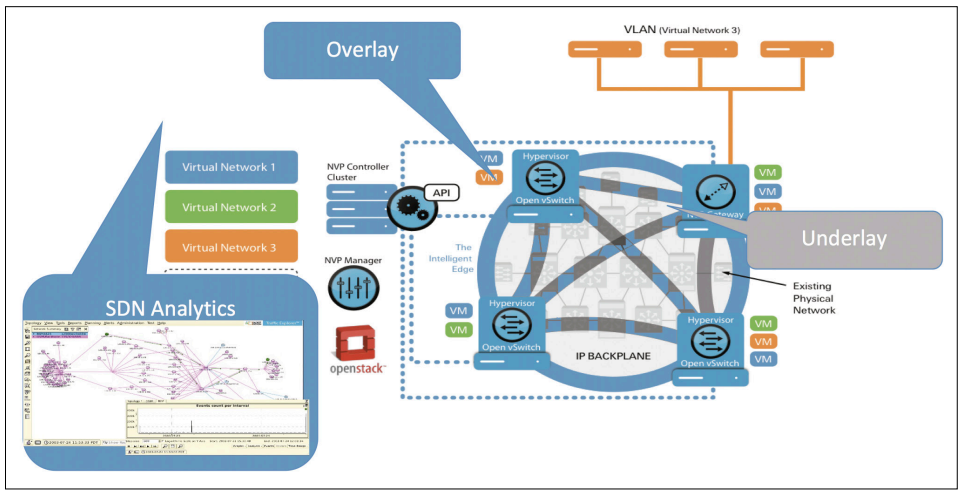


Standbein Hardware: Cisco ACI setzt spezialisierte Switches der Nexus-9000-Produktfamilie voraus, arbeitet aber im Legacy-Modus auch mit anderen Netzwerkgeräten zusammen

sich ein dritter, hybrider SDN-Ansatz auf der Basis von Northbound- und Southbound-APIs entwickelt, der für sich in Anspruch nimmt, die Vorteile von Underlay- und Overlay-SDNs zu verbinden.

Hybrider Ansatz

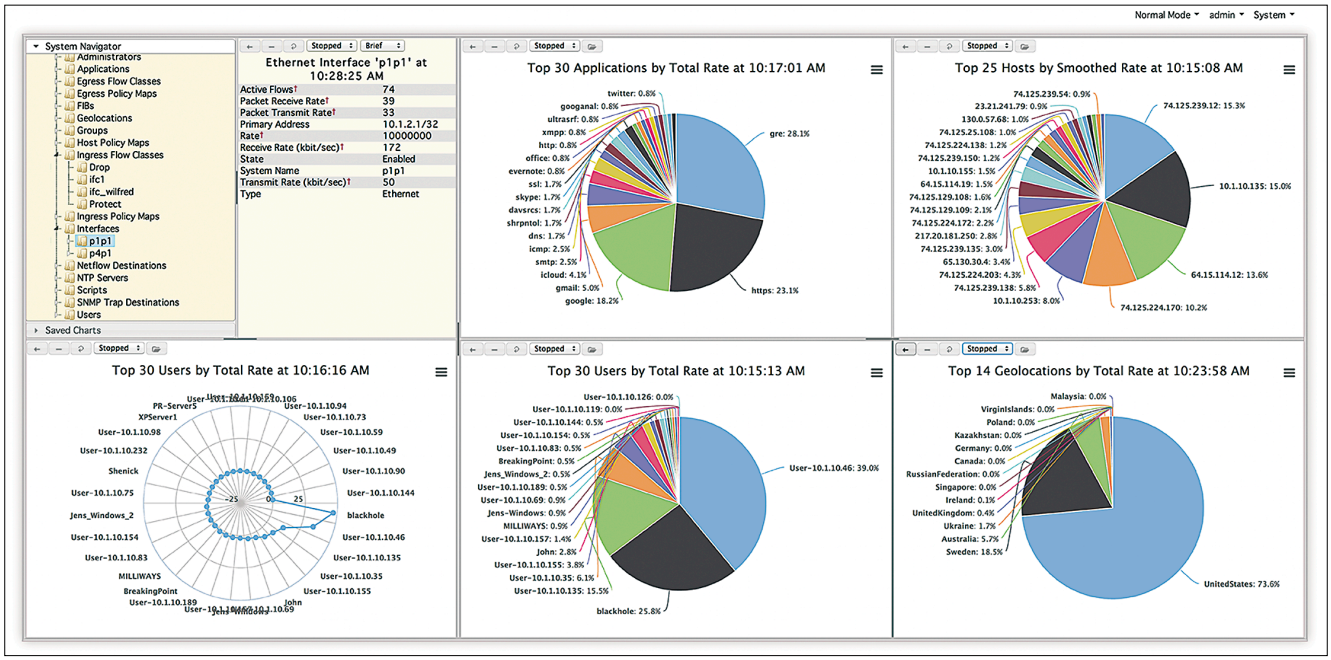
Einen hybriden Ansatz verfolgt unter anderem Arista Networks mit dem Software-Driven-Networking-Konzept (Bild 3). SDN-Lösungen von Arista Networks kombinieren die hohe Geschwindigkeit der vergleichsweise komplexen OSI-Ebenen L2/L3/L4 der Underlay-SDNs mit den relativ unkompliziert einzubindenden Overlay-SDN-Daten, die durch SDN-Controller gesteuert und unter Verwendung von OpenFlow-, Wireless- und anderen Protokollen übertragen werden. Arista Networks empfiehlt sich generell dann, wenn die maßgeschneiderte Optimierung des Netzwerkgewebes (Fabric) und des Overlays im Hybrid-SDN zu merklichen Performance-Gewinnen führen soll.



Monitoring: Mit Hilfe von SDN-Analytics-Werkzeugen lässt sich der Einfluss von technischen Störungen im Underlay-Netzwerk auf die Performance des Overlays in Erfahrung bringen (Bild 4)

Es gibt also insgesamt drei Arten von SDNs:

- **Underlays:** Underlay-SDNs liegen Southbound-APIs zugrunde. Ein Underlay-SDN ist darauf ausgelegt, die Netzwerk-Performance der Datenebene zu maximieren, indem es die Kontrollfunktionen in integrierter Hardware bereitstellt. Beim Underlay-SDN sind die Hardware- und Softwarebestandteile der Netzwerkarchitektur eng integriert und kommunizieren miteinander unter Verwendung von dynamischen Routing-Protokollen wie OSPFv2 (Open Shortest Path First) ohne den Einsatz externer SDN-Controller (Bild 4).
- **Overlays:** Die Umsetzung von Overlay-SDNs wird von Northbound-APIs ermöglicht. Bei einem Overlay-SDN werden Teile der Netzwerkfunktionalität in eine virtualisierte Ebene abstrahiert, sodass sie sich vollständig in Software implementieren lassen. Den physikalischen Unterbau bildet IP-basierte Netzwerk-Hardware. Die SDN-Controller kommunizieren mit Routing-Hardware via APIs unter Verwendung von Protokollen wie OpenFlow oder NETCONF. Routing-Hardware in einem reinen Overlay-SDN verfügt über keine Netzwerkmanagementfähigkeiten, und sie bezieht ihre Anweisungen von spezialisierten SDN-Controllern (Bild 5). ▶

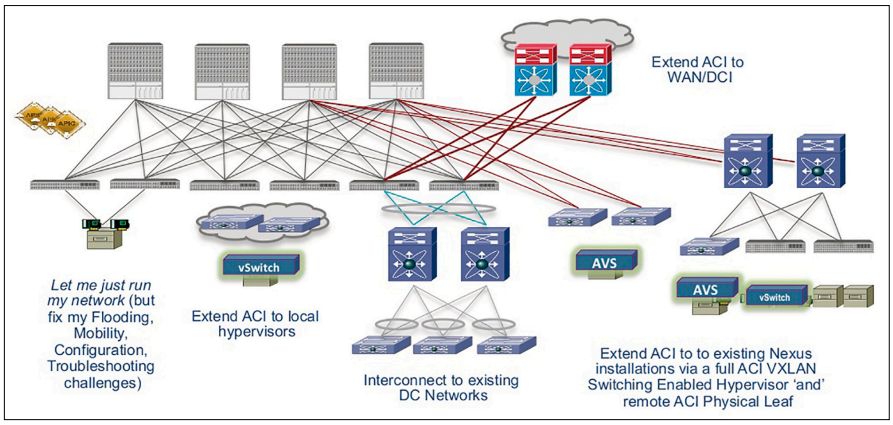


Salsei Flow Command: Das Werkzeug wertet die Datenflüsse in einem Overlay-SDN aus und setzt Sicherheitsrichtlinien um (Bild 5)

- **Hybride SDNs:** Hybride SDNs unterstützen sowohl Northbound- als auch Southbound-APIs. Sie verbinden die Stärken beider Ansätze, ohne allerdings deren Schwächen komplett aufzuheben.

Wenn über das Pro und Contra von SDN-Lösungen gestritten wird, dann werden als Beispiele sehr häufig das Overlay-SDN VMware NSX und das Underlay-SDN Cisco ACI herangezogen (Bild 6).

VMwares Ansatz basiert auf verteilten virtuellen Switches (vSwitches), die innerhalb des Hypervisors arbeiten, die Konnektivität virtueller Maschinen kontrollieren und mit Hilfe eines Overlay-SDNs verbunden sind. Jeder vSwitch bildet eine Bridge zwischen den virtuellen Maschinen des Hosts und dem Netzwerk außerhalb.



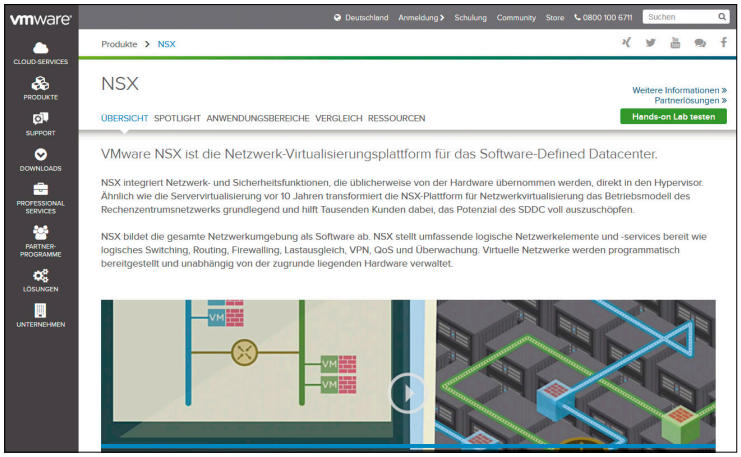
Alt und neu verbunden: Beispiel für die Einbindung von Netzwerk-Altlasten in ein Cisco-ACI-basiertes SDN (Bild 7)

NSX-Controller

Darüber hinaus kann jeder einzelne vSwitch die Aufgaben eines Routers und die einer (verteilten) Firewall und eines Lastverteilers übernehmen. Zwischen den Anwendungen und dem Netzwerk vermittelt der NSX-Controller. Die Southbound-Schnittstelle des Controllers spricht OpenFlow mit der Netzwerk-Hardware.

Das Overlay kommuniziert durch Tunneling der Datenpakete via VXLAN (Virtual Extensible LAN), STT (Stateless Transport Tunneling) und/oder GRE (Generic Routing Encapsulation), um die Kompatibilität mit mehreren Hypervisoren zu gewährleisten. Auf ihrem Weg nach außen müssen diese ein NSX-Gateway passieren.

NSX ist eine erweiterbare Netzwerkplattform, die eine Vielzahl von Lösungen von Anbietern wie Arista, Brocade, Cumulus, Palo Alto Networks, Citrix, F5, Symantec und anderen integriert.



Bei VMware NSX handelt es sich um eine Netzwerkvirtualisierungs-Plattform für das softwaredefinierte Datencenter (SDDC) (Bild 6)

Die wichtigsten Kritikpunkte sind die eingeschränkte Erkennung von Hardware-Versagen und mangelnde Transparenz des Netzwerkgewebes gegenüber Anwendungen. VMware NSX empfiehlt sich vor allem dann, wenn die vorhandene Hardware weiterhin im Einsatz bleiben soll.

Bei Cisco ACI (Application Centric Infrastructure) handelt es sich um eine SDN-Lösung, bei der sich alles um die Anforderungen der betreffenden Anwendungen als Endverbraucher der Netzwerkdienstleistung dreht. Cisco steht auf dem Standpunkt, dass sich die IT-Landschaft dauernd ändert und dass Infrastructure as a Service (IaaS) mehr und mehr dem AaaS-Ansatz (Applications as a Service) weicht (Bild 7). Potenzielle Anwender der ACI-Technologie müssen nicht nur auf Netzwerk-Hardware anderer Hersteller, sondern selbst auf bestehende, aber nicht unterstützte Cisco-Produkte verzichten und neue Hardware anschaffen. Alte Lösungen von Cisco laufen zwar mit ACI, aber nur mit reduzierter Performance.

Kostspielige Hardware-Anforderungen

Der Einsatz von Cisco ACI erfordert Switches aus der Nexus-9000-Produktlinie und mindestens einen APIC-Hardware-Controller (Application Policy Infrastructure Controller). Diese vielfach als übertrieben kostspielig empfundenen Hardware-Anforderungen hat Ciscos ACI-Technologie den Spitznamen Hardware-definierte Netzwerke eingebracht.

Mit dem Application Virtual Switch bietet Cisco auch einen eigenen virtuellen Switch. Zu den unterstützten Hypervisoren zählen die von Microsoft, VMware, Red Hat und Citrix. Für die Konnektivität zwischen den Nexus-9000-Switches (in der Leaf- und Spine-Zuordnung) kommt VXLAN zum Einsatz.

Eine ideale Lösung für sämtliche Nutzungsszenarien gibt es nicht. Cisco ACI bietet sich in erster Linie für ein Cisco-zentrisches Unternehmen an, wenn es darum geht, die Fabric zu modernisieren und zu beschleunigen. Cisco konzentriert sich beim Support ohnehin auf die ACI-Technologie und ist dabei, die eigenen Legacy-Technologien schrittweise abzuschaffen.

Mit dem verstärkten Einsatz von SDNs rücken Sicherheitsaspekte in den Vordergrund.

Ein durchdacht implementiertes SDN kann gegenüber konventioneller Netzwerktechnik eine deutlich höhere Sicherheit bieten. So lassen sich etwa OpenFlow-basierte Access-Switches nutzen, um vorab alle Datenpakete zu filtern, die ins Netzwerk gelangen sollen. Außerdem lässt sich die Ressourcenzuordnung auf der Basis von Rollen einrichten und via SDN-Controller überwachen. SDN-Controller können zudem DDoS-Attacken abwehren. Sicherheitsdienstleister, die sich auf die Verteidigung von Datencentern gegen DDoS-Attacken spezialisiert haben, setzen ebenfalls auf SDNs.

Mit SDNs lassen sich zwar sicherere Netzwerke einrichten, aber bisher sind bei Weitem nicht alle Lösungen automatisch abgesichert. SDN-Administratoren kommen daher nicht darum herum, sich mit dem Thema Sicherheit von SDNs auseinanderzusetzen.

Intelligente Controller

Mittlerweile gibt es Implementierungen von Overlay-SDNs, die mit mehreren intelligenteren Controllern aufwarten können. Dennoch ist diese Konfiguration längst nicht der Standardfall.

Ein einzelner zentralisierter SDN-Controller eines Overlay-SDN kann als ein potenzieller Single Point of Attack schnell zum ersten Opfer werden. Sollte der SDN-Controller im Fall eines Angriffs temporär offline gehen, muss er in der Lage sein, sich bei der Wiederverfügbarkeit umgehend um die Synchronisierung der Daten zu kümmern.

Das Southbound-Interface zwischen dem SDN-Controller und den darunter befindlichen Netzwerkgeräten ist potenziellen Angriffen ausgesetzt. Es sind Schutzmechanismen vonnöten, um die Verfügbarkeit, die Leistungsbereitschaft und die Integrität des Netzes zu gewährleisten.

Vorteile eines SDN

Softwaredefinierte Netzwerke bieten Unternehmen zahlreiche Vorteile, darunter:

- die Fähigkeit zur schnellen, bedarfsgerechten Provisionierung von Netzwerkressourcen,
- eine höhere Netzwerktransparenz dank verbessertem Monitoring durch ein umfassendes Netzwerkmanagementsystem,
- richtliniengesteuerte Kontrolle der Datenflüsse,
- Fähigkeiten zur automatisierten Bereitstellung durch Netzwerkorchestrierung,
- selbstheilende Netzwerktopologien bei Hardware-Fehlern mit automatischer Eingliederung nach erneuter Bereitstellung,
- geringere operative Kosten,
- verbesserte Auslastung bestehender Netzwerk-Hardware,
- Verwaltung von Datenströmen anhand von QoS-Kriterien (zum Beispiel pro Anwendung oder pro Benutzer),
- verbesserte Netzwerksicherheit unter anderem durch intelligente DDoS-Abwehr (sofern unterstützt).

Links zum Thema

- Anbieter hybrider Switches für Underlay- und Overlay-SDNs **www.arista.com/en**
- Unterstützt SDN auf der Basis von Open-Source-Projekten **www.bigswitch.com**
- SDN-Lösung zur Analyse des Datenflusses in Echtzeit **http://saisei.com**
- Konsortium, das einen SDN-Standard etablieren will **www.opendaylight.org**

Die Hardware in einem SDN muss in der Lage sein, trotz des Ausfalls des (letzten) SDN-Controllers gelegentliche Angriffe zu überstehen und trotzdem neue Netzwerkdatenflüsse umgehend synchronisieren können, sobald die betroffenen SDN-Controller ihre Betriebsbereitschaft wiederherstellen.

Um Schwachstellen zu minimieren, gilt es, das System des SDN-Controllers und des Host-Betriebssystems beim Einsatz von virtuellen Maschinen zu härten. Darüber hinaus sollten Authentifizierungs- und Autorisierungs-Prozeduren implementiert werden, die den Zugriff auf SDN-Controller beschränken.

Fazit

Die SDN-Ära bringt einen Paradigmenwechsel mit sich: Anstatt die bloße Übertragung von Datenpaketen von Schnittstelle zu Schnittstelle zu verwalten, wird verteilte Software eingesetzt, die komplette Datenflüsse intelligent optimiert.

Im Markt für SDN-Lösungen dominieren derzeit drei Ansätze: Overlay-SDN (Northbound-APIs), Underlay-SDNs (Southbound-APIs) und hybride SDNs (Northbound- und Southbound-APIs).

Unter den IT-Entscheidungsträgern setzt sich schrittweise die Überzeugung durch, dass SDN-Technologie nicht nur die Provisionierung und Auslastung physikalischer Infrastruktur verbessert, sondern auch die Netzwerksicherheit erhöhen kann. Eine fachgerecht umgesetzte SDN-Architektur zählt unter anderem zu den bewährten Lösungen, wenn es darum geht, verteilte DoS-Attacken abzuwehren, die neue Plage aus der Cloud. ■

Filipe Pereira Martins und Anna Kobylinska sind international anerkannte IT-Berater mit Schwerpunkt auf Cloud-Lösungen. Sie stehen den Lesern der **web & mobile developer** gern per Twitter via **@D1gitalPro** und **@D1gitalInfo** zur Verfügung.

AUSWAHLKRITERIEN FÜR EINEN ONLINE-SHOP

Optimale Auswahl

Aus technischer Sicht ist das Betreiben eines eigenen Online-Shops eher unproblematisch.

Die optimale Auswahl einer passenden E-Commerce-Plattform hängt vor allem von den eigenen Geschäftsprozessen ab. Daher ist es wichtig, dass man sich als Betreiber eines Online-Shops nicht nur darüber im Klaren ist, welche Produkte man vertreiben möchte. Eine intensive Betrachtung des gesamten Geschäftsmodells ist für ein solches Vorhaben unabdingbar. Händler, die besonderen Wert auf Unabhängigkeit legen, werden sich kaum mit den starken Restriktionen von Ebay, Amazon und Co. anfreunden wollen.

Das hat allerdings zur Folge, dass eine tiefgehende Beschäftigung mit der Thematik E-Commerce notwendig wird, um so auch langfristigen Erfolg sicherstellen zu können. Schließlich ist die eingesetzte Software nur ein Werkzeug, das Sie in Ihrem Vorhaben unterstützt. Den richtigen Umgang damit kann man mit überschaubarem Aufwand ohne viele Schwierigkeiten schnell erlernen. Zum leichteren Einstieg in die Thematik klären wir gleich vorab einige Begrifflichkeiten, deren Verständnis äußerst hilfreich ist.

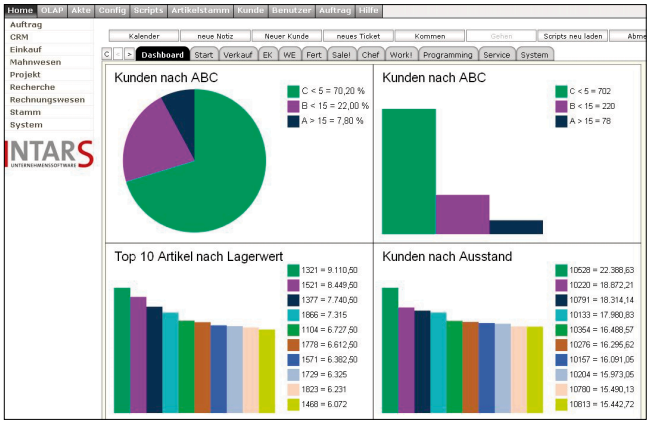
Fachchinesisch oder doch nur Kauderwelsch?

Einige Mitmenschen neigen dazu, ihre Unkenntnis in einer Thematik damit zu überdecken, dass sie allgemein bekannte Fachbegriffe anbringen. Deren wirkliche Bedeutung aber ist in den meisten Fällen nicht geläufig. Eine solche Strategie mag zwar verlockend sein und schindet in Kreisen von Nichteingeweihten oftmals ordentlich Eindruck, ist der eigenen Sache aber nicht dienlich.

Beginnen wir mit der Frage, worin sich eine E-Commerce-Plattform zu einem Online-Shop unterscheidet.

Ein Online-Shop ist eine Software, die eine Liste an Artikeln präsentiert, aus der ein Kunde nach seinen Wünschen auswählen kann. Die Aufstellung der individuellen Auswahl von Artikeln ist bis zum Abschluss des Bestellvorgangs durch den Kunden veränderbar und wird Warenkorb genannt. Der Warenkorb zeigt übersichtlich die Stückpreise und die zugehörigen Gesamtpreise an. Sind alle Artikel ausgewählt, kann der Kunde den Bestellprozess einleiten, in dem die folgenden Schritte abgearbeitet werden:

- Überprüfen der Artikelverfügbarkeiten,
- Auswählen der Bezahloption,
- Festlegen der Versandart,
- Ermitteln und Anzeigen der Versandkosten,
- Aufnehmen der Kundendetails wie Rechnungsadresse und Lieferanschrift,
- Einblenden und Akzeptieren der Allgemeinen Geschäftsbedingungen (AGB),
- Abschließen der Bestellung und Versenden der Bestellbestätigung via E-Mail.



Intars stellt eine freie Alternative zu kommerziellen ERP-Systemen dar (Bild 1)

Bei einer E-Commerce-Plattform ist der Shop nur eine Komponente von vielen, es kommen beispielsweise noch Warenwirtschaftssystem und Marketingwerkzeuge hinzu.

Optimierung von Geschäftsprozessen

Oft wird von Geschäftsprozessen und deren Optimierung gesprochen, aber was steht hinter diesem mystischen Vorhaben und welche Resultate kann man davon erwarten?

Ein Prozess im Allgemeinen ist das Verarbeiten einer Eingabe zu einer Ausgabe. Dabei können Prozesse sehr klein sein oder aus anderen Prozessen zusammengesetzt werden. Bisher haben wir einen zusammengesetzten und komplexen Prozess kennengelernt, den Bestellprozess. Dazu ein anschauliches Beispiel: Nehmen wir an, Sie vertreiben bereits erfolgreich im eigenen Online-Shop Ihre Produkte. Dank glücklicher Umstände verdoppeln sich innerhalb kurzer Zeit die Bestellungen. Nun stehen Sie vor dem Problem, dass Sie mit der Abwicklung der Bestellflut kaum hinterherkommen.

Die Zeit ist für neues Personal noch nicht reif. Zudem müssen diese Leute auch erst eingearbeitet werden, was die Bearbeitung der Bestellungen weiter verzögert. Lange Lieferzeiten sind kein Erfolgsrezept für zufriedene Kunden. Was also ist eine gute und praktikable Lösung für dieses Problem? Die Antwort ist recht einfach.

Zuerst werden alle Schritte, die für den Versand notwendig sind, näher betrachtet und die arbeitsintensiven Bereiche herausgelöst. Dies sind die Kandidaten, bei denen Verbesserungen sofort ins Gewicht fallen. In der Praxis sind dies oft die Bereiche der automatischen Rechnungsgenerierung, das Drucken der Adressaufkleber und das Prüfen der Zahlungseingänge.

Damit unnötige Nacharbeiten, wie zum Beispiel die entstehende Kommunikation bei nicht verfügbaren Artikeln, unterbleiben können, ist es mehr als empfehlenswert, ein einfaches Warenwirtschaftssystem (Enterprise Resource Planning – ERP) zu verwenden (Bild 1). Viele Systeme bieten dies von Haus aus an. Schließlich erwartet Ihre Kundschaft eine zügige Lieferung der vollständigen Bestellung. Ein voll ausgestattetes ERP erlaubt zudem eine detaillierte Analyse des Produktportfolios. So lassen sich die Ladenhüter von den Selbstläufern schnell unterscheiden, um geeignete Marketingmaßnahmen zu ergreifen.

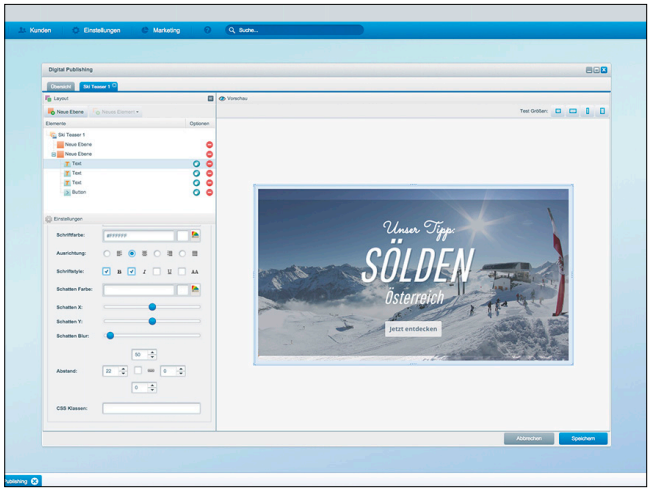
Produktpräsentation und Inhaltserstellung

Die Inhalte von Online-Shops sind naturgemäß die einzelnen Produkte, die in Kategorien zusammengefasst werden. Üblicherweise ergibt sich aus der Verschachtelung der Kategorien die spätere Navigation.

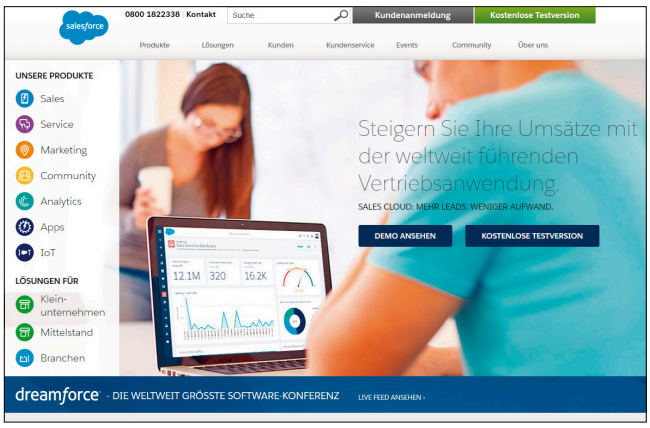
Grundsätzlich können Kategorien beliebig tief verschachtelt werden. Um Ihren Besuchern aber die Orientierung zu erleichtern, sollten Sie nicht mehr als drei Verschachtelungsebenen anlegen. Die meisten Shops erlauben es, einem Artikel mehrere Kategorien zuzuordnen. So ist es möglich, ein Produkt in unterschiedlichen Übersichtsseiten einzubinden.

Wenn Sie beispielsweise T-Shirts vertreiben, möchten Sie garantiert nicht für identische Modelle, die sich eventuell nur in der Größe unterscheiden, separate Darstellungen in der Produktübersicht. Für diesen Zweck gibt es Produktgruppen, die anhand von definierten Eigenschaften (Attribute) die konkreten Artikel unterscheiden und die Artikelbeschreibung miteinander teilen. Die Güte eines Webshops lässt sich daran bestimmen, wie komfortabel neue Produkte in das System eingepflegt werden können.

Andere Inhaltsformen sind die Allgemeinen Geschäftsbedingungen (AGB), Widerrufsbelehrung und Impressum, um nur einige zu nennen. Um solche Texte zu verwalten, besitzen die meisten Shops ein Content-Management-System (CMS), mit dem man beliebig viele individuell gestaltete Inhaltsseiten verwalten kann (Bild 2).



Shopsysteme wie beispielsweise Shopware haben ausgefeilte CMS integriert (Bild 2)



Salesforce zählt zu den größten Anbietern von CRM-Systemen (Bild 3)

Für Händler, die ihre Waren in Deutschland vertreiben und sich nicht in Abhängigkeit von Amazon und Ebay begeben wollen, gilt es darauf zu achten, dass der Online-Shop auch den geforderten gesetzlichen Bestimmungen genügt.

Viele Plattformen, die für den internationalen Markt entwickelt wurden, verfügen oft über ein zusätzliches Paket, welches das System fit für die deutschen Gesetze macht. Dabei ist auf Aktualität zu achten, sodass auch Updates verfügbar sind, die die aktuellsten Gesetzesnovellen berücksichtigen und entsprechend umsetzen. Bietet ein Shop-Hersteller weder Aktualisierungen noch Support bei Problemen, sollte man besser Abstand von einer Verwendung nehmen. Schnell summieren sich dann die Kosten für Spezialisten, die die notwendigen Änderungen für Sie vornehmen.

In den nächsten Jahren erwartet die E-Commerce-Branche noch einige neue Vorgaben aus Brüssel, die den länderübergreifenden Online-Handel vereinfachen sollen. Wichtige Punkte in diesem Zusammenhang sind die korrekte Auszeichnung der Steuer und eine korrekte Preisaufstellung mit den Positionen und den anfallenden Versandgebühren.

Es gelten auch unterschiedliche Regeln für Geschäftsbeziehungen mit anderen Händlern oder zu Endverbrauchern. Daher wird zwischen Business to Business (B2B) und Business to Customer (B2C) unterschieden. Das äußert sich vor allem bei den Mindestbestellmengen, den Preisen und den Zahlungsmodalitäten.

Guter Kunde – böser Kunde

Dass jeder Mensch stets nur die besten Absichten hegt, ist leider nicht immer die Tagesordnung. So gilt es auch als Händler, die Spreu vom Weizen zu trennen. Schließlich ist es kein geringes Geschäftsrisiko, wenn höhere Beträge ausstehend bleiben. Um die eigenen Kunden besser einschätzen zu können, hilft die Verwendung einer Customer-Relationship-Management-Software (CRM) (Bild 3).

Einerseits unterstützt ein solches Werkzeug Sie bei der Planung und der anschließenden Auswertung unterschiedlicher Marketingmaßnahmen. Andererseits können Sie damit zahlungsunwillige Vertragspartner klassifizieren und geeignet agieren. Auch ist es nicht förderlich, an Kunden auszulie- ▶

fern, die eine hohe Rücksendungsquote haben. Eine hohe Zahl an Rückläufern verursacht einen hohen Arbeitsaufwand zum Beispiel durch Prüfen der Ware auf Mängel, Gutschriften und so weiter.

Dies erzeugt entsprechend hohe Kosten und sollte weitgehend vermieden werden. Möglicherweise sind Gründe für Rücksendungen eine unzureichende Produktbeschreibung, mangelhafte Qualität durch den Hersteller, oder die Präsentation durch die Fotos ist nicht aussagekräftig genug oder gar irreführend. Sobald die konkreten Ursachen für diese Problemkategorie bekannt sind, kann dem effizient entgegengewirkt werden. Eine Lösung könnte dahin gehen, dass ein anderer Hersteller ausgewählt wird, um eine bessere Qualität ausliefern zu können.

Marketingaktionen können verschiedene Ausprägungen haben. Von Multi-Channel-Marketing ist die Rede, wenn Anzeigen beispielsweise in einschlägigen Magazinen geschaltet werden. Es kann auch für bestimmte Suchbegriffe bei Google Werbung eingeblendet werden. In diesem Fall spricht man von Suchmaschinen-Marketing (SEM).

Auch Newsletter per E-Mail mit Angeboten für Bestandskunden sind weit verbreitet. Sie sollten jedoch bei E-Mail penibel darauf achten, dass nur an Empfänger verschickt wird, die Ihre Newsletter auch erhalten möchten. Auch die Häufigkeit eines solchen Mailings ist zu bedenken. Ein monatlicher Newsletter ist oft mehr als ausreichend.

Für sämtliche Bemühungen ist es natürlich unabdingbar, den Nutzen den Kosten gegenüberzustellen und so den Erfolgsfaktor zu bestimmen. Da für viele Menschen die Zeit ein knappes Gut ist, sollen unrentable Aktivitäten möglichst vermieden werden. Eine Option, um den Erfolg einer Marketing-Kampagne zu messen, können Gutscheine sein. Anhand der vergebenen Nummernkreise des Gutschein-Codes lässt sich später die Quelle zurückverfolgen.

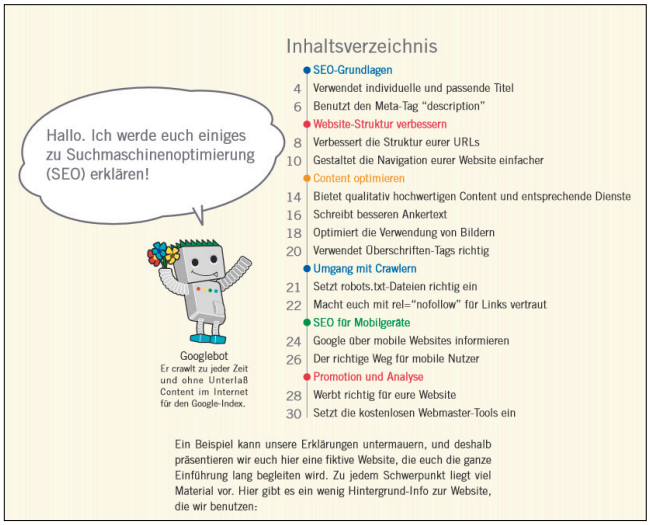
Akquise von Neukundschaft

Ein anderer wichtiger Bereich ist die Akquise von Neukundschaft, um so langfristig Wachstum zu generieren. Wie auch bei Ladengeschäften gilt auch hier: Lage, Lage, Lage.

Unter dem Begriff Search Engine Optimization (SEO) werden verschiedene Maßnahmen zusammengefasst, die den eigenen Internetauftritt bei Google unter den ersten Treffern auflisten (Bild 4). Um sogenannten qualifizierten Traffic zu erhalten, ist es unabdingbar, die korrekten Suchbegriffe zu ermitteln. Es hat wenig Nutzen, Besucher zu generieren, die kein Interesse an Ihrem Angebot haben.

Um sinnvolle Suchbegriffe ermitteln zu können, stehen zur Keyword-Recherche verschiedene Tools zu Verfügung. Mittlerweile kann man auch eine der unzähligen Agenturen für SEO beauftragen. Es existiert auch eine enorme Auswahl an guter Literatur für diese Thematik. Bei der Auswahl einer geeigneten Shop-Software sollte man darauf achten, dass es Möglichkeiten zur individuellen Anpassung des eigenen SEO-Konzepts gibt.

Nicht ganz unkritisch ist der Fall, wenn die vorhandene Plattform gegen eine neue ausgetauscht werden muss. Die Gründe dafür sind sehr unterschiedlich, aber die Durchführung sollte nur mit erfahrenen Teams angegangen werden. Das Risiko, dass Daten verloren gehen, ist sehr hoch. Auch wenn Import- und Export-Funktionen vorhanden sind, erfüllen sie oft nicht die notwendigen Anforderungen.



SEO: Vom Suchmaschinenbetreiber Google gibt es eine umfangreiche SEO-Anleitung (Bild 4)

Es gibt unterschiedliche Arten von Daten, die nicht alle dieselbe Priorität haben. So kann die technische Konfiguration des Shops oft vernachlässigt werden. Der erfasste Kundenstamm und die zugehörigen Bestellungen haben dagegen einen hohen Stellenwert. Statistiken der besuchten Seiten et cetera werden oft aus Kostengründen nicht übernommen. Hier erzeugt man einen abschließenden Report und die entsprechende Auswertung für mögliche Optimierungen.

Schnell wird klar, dass ein solcher Plattformwechsel langfristig geplant werden sollte, um unliebsame Überraschungen zu vermeiden. Schließlich bedeutet jede Stunde, die der Shop nicht live ist, finanziellen Verlust. Deshalb darf man die Entscheidung für einen Wechsel nicht bis zum letzten Augenblick hinauszögern. Reserven zu haben ist also unverzichtbar.

Fazit

Die Thematik des E-Commerce bietet weitaus mehr Facetten, als dieser Artikel behandeln kann. So wurde etwa bewusst auf die Fragestellung der Internationalisierung verzichtet, und auch der Aspekt »Mandantenfähige Systeme« wurde nicht behandelt. Diese speziellen Lösungen sind mit erheblichen Lizenz- und Implementierungskosten verbunden. ■

Marco Schulz
studierte an der HS Merseburg Informatik. Sein persönlicher Schwerpunkt liegt in der Automatisierung von Build-Prozessen und dem Softwarekonfigurationsmanagement. Seit über zehn Jahren entwickelt er auf unterschiedlichen Plattformen Webapplikationen.

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. Günther Götz

Chefredakteur
Max Bold
– verantwortlich für den redaktionellen Teil –
E-Mail: redaktion@webundmobile.de

Schlussredaktion
Ernst Altmannshofer

Redaktionelle Mitarbeit
Philip Ackermann, Philippe Bénard, Christian Bleske, Patrick Fend, Ekkehard Gentz, Jens Geyer, Thomas Hafen, Tam Hanna, Anna Kobylinska, Bernhard Lauer, Patrick Lobacher, Filipe Martins, Florence Maurice, Frank Pientka, Michael Rohrlich, Markus Schraudolph, Marco Schulz, Katharina Sckommodau, Christoph Spannagel, Kai Spichale

Art Directorin
Maria-Luise Sailer

Grafik & Bildredaktion
Alfred Agatz, Dagmar Breitenbauch, Verena Greimel, Hedi Hefele, Manuela Keller, Simone Köhnke, Cornelia Pflanzer, Karoly Pokuta, Petra Reichenspurner, Ilka Rüther, Sebastian Scharnagl, 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

Vertrieb
Axel Springer Vertriebsservice GmbH
Objektvertriebsleitung Lothar Kosbü
Süderstraße 77
20097 Hamburg
Telefon: (040) 34724857

Bezugspreise
web & mobile developer ist das Profi-Magazin für Web- und Mobile-Entwickler und erscheint zwölfmal im Jahr. Der Bezugszeitraum für Abonnenten ist jeweils ein Jahr. Der Bezugspreis im Abonnement beträgt 76,20 Euro inklusive Versand und Mehrwertsteuer im Halbjahr, der Preis für ein Einzelheft 14,95 Euro. Der Jahresbezugspreis beträgt damit 152,40 Euro.

In Österreich sowie im übrigen Ausland kostet das Abonnement 83,70 Euro im Halbjahr. Der Jahresbezugspreis beträgt somit 167,40 Euro. In der Schweiz kostet das Abonnement 152,00 Franken im Halbjahr. Der Jahresbezugspreis in der Schweiz beträgt 304,00 Franken.

Das Abonnement verlängert sich automatisch um ein Jahr, wenn es nicht sechs Wochen vor Ablauf der Bezugszeit schriftlich beim Verlag gekündigt wird. Studenten erhalten bei Vorlage eines Nachweises einen Rabatt von 50 Prozent.

ISSN: 2194-4105

LOGFILES MIT ELASTIC STACK ANALYSIEREN

Schatz und Problem

Der Elastic Stack bietet eine einfache Lösung für die sinnvolle Nutzung von Logfiles.

Vor zehn Jahren war das Archivieren und Durchsuchen von Logdateien noch ein vergleichsweise übersichtliches Thema: In den Zeiten vor Smartphone und Co. war die Datenmenge zwar höher als jemals zuvor, doch immer noch relativ überschaubar. Das hat sich fundamental geändert: Die ständig wachsende Menge an Logfiles ist ein Schatz und ein Problem zugleich.

Fakt ist: Das Internet gehört zum Alltag. Vom Smartphone mal eben die Mails, WhatsApp oder Facebook checken, per App die nächste Zugverbindung suchen und buchen – die Liste der Möglichkeiten ließe sich endlos fortsetzen. Die mit diesen Prozessen verbundenen Datenmengen sind in den letzten Jahren explosionsartig angewachsen. Monitoring, Ressourcenplanung und Fehlersuche stellen IT-Experten zunehmend vor Herausforderungen. Die Menge der durch Web- und Applikationsserver anfallenden zu protokollierenden Logdateien war noch vor zehn Jahren recht überschaubar. Zum Sammeln von Logfiles reichten Tools wie Rsyslog völlig aus, die Daten wurden meist auf einem einzelnen Server gespeichert. Doch in Anbetracht der heutigen Datenmenge sieht das anders aus: Der Speicher ist irgendwann voll und Speicherplatz teuer. Außerdem nutzt die Cloud-Technologie Server, die flexibel entstehen, aber auch wieder verschwinden. Und dann sind die Logdateien weg.

Eine Lösung, um Logdateien viel sinnvoller zu nutzen, bietet das auch mit dem Begriff Elastic Stack bezeichnete Zusammenspiel der Tools Elasticsearch, Logstash, Kibana und Beats (Bild 1). Die vier Open-Source-Tools sind für jeden kostenfrei zugänglich und aufgrund der einfachen Handhabung sowie der guten Skalierbarkeit in den letzten Jahren recht bekannt geworden. Sie ermöglichen die einfache Speicherung, Auswertung und Visualisierung von Logdateien. Deshalb finden diese Werkzeuge vor allem im Monitoring von Zugriffen und bei der Fehlersuche ihre Anwendung.

Logfiles durchsuchen mit Elasticsearch

Elasticsearch ist eine ursprünglich aus Lucene entstandene Suchmaschine, die die Ergebnisse in einer NoSQL-Datenbank speichert. Sie ermöglicht ein einfaches und komfortables Durchsuchen von Logdateien und macht Zusammenhänge transparent.

Das Tool ist explizit darauf ausgelegt, preiswert in die Breite zu skalieren: Statt immer größere und teurere Server einzusetzen, macht Elasticsearch es möglich, die Datenspeicherung auf mehrere kleinere Server zu verteilen. Daraus ergibt sich ein weiterer Vorteil: Die Vorhaltezeit für die Logs ist nahezu unbegrenzt. Daten können bei Bedarf monate- oder jahrelang gespeichert werden.

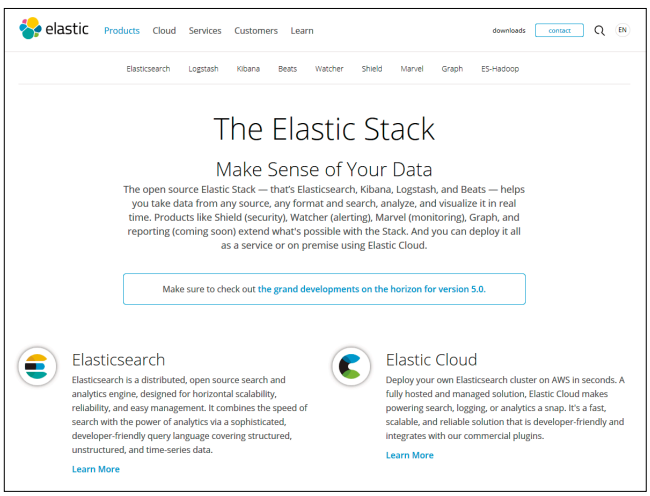
Als Logserver mit eingebauten Analyse-Tools kann Logstash die Protokolle vieler Server zentral archivieren. Gleichzeitig macht das Tool die Daten quasi nebenbei auch noch durchsuchbar: Es sammelt die Logdateien, bereitet sie auf, schickt sie an einen zentralen Logstash-Server und schreibt sie in Elasticsearch. Das heißt, Logstash fasst jeden Logeintrag, der eine oder mehrere Zeilen umfasst, zu einer Entität zusammen und schreibt diese dann in die Datenbank. Zusammengehörende Einträge liegen zusammengefügt statt verteilt vor. Die Anzahl, Art und Weise der Datenaufbereitung bestimmt der Nutzer. Alternativ kann der zentrale Logstash-Server auch Logs von Tools wie Rsyslog annehmen und wie beschrieben aufbereiten.

Logstash stellt damit eine Sammlung von Fehlerquellen zur Verfügung. Diese ermöglicht es, fehlerhafte oder problematische Server leicht zu identifizieren. Doch nicht nur das: Aufgrund der vollständigen Erfassung von Webserver- und Datenbank-Server-Logs können Probleme sehr gut miteinander in Verbindung gesetzt werden.

Komfortables Visualisieren von Daten: Kibana

Mit Kibana steht schließlich ein Web-Frontend zur Verfügung, um die gesammelten Daten zu visualisieren: Das Tool kann die Daten lesen, nach Belieben filtern und je nach Sinn und Ziel ansprechende Visualisierungen daraus erstellen.

Die Anwendungsszenarien sind vielfältig: IT-Leiter können beispielsweise eine einfachere Übersicht über die Auslastung ihrer Webserver erhalten. Oder es können Aufrufe einer Website aus verschiedenen Ländern verglichen werden. Da-



Die vier Open-Source-Tools Elasticsearch, Logstash, Kibana und Beats sind für jeden kostenfrei zugänglich (Bild 1)

Kibana bietet ein Web-Frontend zur Visualisierung der gesammelten Daten (Bild 2)

rüber hinaus lässt sich der Verlauf der Zugriffe über einen sehr langen Zeitraum in der Historie anzeigen, um daraus gegebenenfalls Aussagen für die Zukunft zu treffen. Ein anderes Beispiel ist das Betreiben mehrerer Webseiten auf einem Server. Reagiert ein Server langsam, ist recht schnell klar, ob die Ursache im immensen Traffic liegt.

Mit Kibana lässt sich leicht herausfinden, welche Website den starken Traffic verursacht – Ressourcen können daraufhin gegebenenfalls neu geplant werden: Liegt die stark besuchte Website eventuell auf einem anderen Server, werden die übrigen Seiten nicht gestört. Das Gleiche gilt für Bilder und Videos, die häufig aufgerufen werden. Die Beispiele zeigen: Die Visualisierung der Daten mit Kibana vereinfacht wesentlich die Kampagnen- und Ressourcenplanung (Bild 2).

Das Maximum aus den Daten herausholen: Beats

Beats ist das neueste Tool im Elastic Stack. Es bietet mit operativen Analysen Einblicke in Konfigurationen, Kapazität, Fehler und Events der Server- und Netzwerkinfrastruktur in Rechenzentren oder Cloud-Infrastrukturen. Die Beats-Plattform sendet die Events sicher und geschützt an Logstash und Elasticsearch. Derzeit sind drei Beats verfügbar:

- **Topbeat** dient der Zusammenstellung von CPU, Speicherplatz und anderen Prozessen. Der auf die Server verteilte Top-Befehl unterstützt Linux, Mac OS X und Windows und sendet die Metriken regelmäßig an Logstash oder Elasticsearch. Systemweite Daten wie zur System- und Speicherbeziehungsweise Festplattenauslastung sowie prozessspezifische Metriken können zentral organisiert werden.
- **Packetbeat** stellt Echtzeitanalysen für das Internet, Datenbanken und andere Netzwerkprotokolle bereit. Dafür setzt es Anfrage und Antwort in einer Transaktion zueinander in Beziehung und fügt die Daten über jede Transaktion in Elasticsearch ein. Es überwacht somit die Kommunikation zwischen den Servern im Netzwerk und stellt Einblicke in die Anwendungen zur Verfügung.
- **Filebeat** übernimmt schließlich das Weiterleiten der Logdateien. Die Konfiguration von Filebeat erfolgt ganz einfach nach der Installation: Auf den Servern werden nur die Pfade konfiguriert, die Filebeat absuchen soll.

Zahlreiche weitere Beats sind bereits in Planung. Basis der Beats-Plattform ist die Libbeat-Bibliothek, die in der relativ neuen Open-Source-Programmiersprache Go geschrieben ist und auch als golang bezeichnet wird. Bei Google entwickelt, hat sie das Ziel, für große Systeme skalierbar zu sein. Sie dient darüber hinaus Networking- und Multiprocessing-Zwecken. Typische Anwendungsgebiete sind Analysen von Netzwerken oder das Prüfen der Systemauslastung.

Das Zusammenspiel von Elasticsearch, Logstash, Kibana und Beats vereinfacht das Monitoring von Web- und Applikationsservern: Schwer zu konsolidierende Daten werden auswertbar – und damit transparent und nutzbar, etwa als Entscheidungsgrundlagen. Und das zu sehr geringen Kosten. Die Open-Source-Tools sind einfach in der Anwendung – und das für jede Applikation, unabhängig von deren Größe. Außerdem sind sie für jeden einfach zugänglich. Das macht den Elastic Stack ziemlich einmalig: Alternativen wie Splunk und Greylog sind kostenpflichtig und in Deutschland kaum verbreitet.

Wichtig ist auch hier, genau zu überlegen, was das Ziel der Datensammlung und -auswertung sein soll. Das wird häufig unterschätzt: Die gesammelten Daten haben an sich noch keinen Nutzen. Das Sammeln von 300 Millionen Einträgen von einer Applikation ist leicht gemacht, doch sinnlos, wenn nicht klar ist, was damit passieren soll. Die Tools sind im Sammeln und Aufbereiten der Daten nicht wählerisch. Es empfiehlt sich deshalb, frühzeitig darüber nachzudenken, was am Ende mit den Daten erreicht werden soll, und dann nur die dafür notwendigen Daten zu sammeln. Für Einsteiger hat der Elastic Stack den Vorteil, dass sie erst mal mit kleinen Applikationen beginnen und sich ausprobieren können.

Fazit

Das Zusammenspiel der Tools Elasticsearch, Logstash, Kibana und Beats ermöglicht das Konsolidieren, Durchsuchen, Speichern und Auswerten von Logdateien. Letzteres eröffnet mit operativen Analysen umfassende Einblicke in Konfigurationen, Kapazität, Fehler und Events der Server- und Netzwerkinfrastruktur-Elemente in Rechenzentren oder Cloud-Infrastrukturen. Momentan stehen Topbeat, Packetbeat und Filebeat zur Verfügung. Es sind aber weitere Beats geplant. Während Logstash die Daten zusammenführt, durchsuchbar macht und sie auf einem zentralen Server ablegt, übernimmt die Suchmaschine Elasticsearch die Speicherung – bei Bedarf auf mehreren kleineren Servern. Das Web-Frontend Kibana dient schließlich der Visualisierung der Suchergebnisse: Verschiedene ansprechende Diagramme und Graphen stehen je nach Sinn und Zweck der Auswertung zur Verfügung. ■

Patrick Fend
ist einer der drei Unternehmensgründer der Adacor Hosting GmbH und der Technikexperte im Unternehmen.
www.adacor.com

AKTUELLE REGELN UND GESETZE FÜR DEN ONLINE-HANDEL

Neue Regeln

Die neuesten Gesetze, Verordnungen und Urteile für E-Commerce-Betreiber.

Ein bisschen Datenschutz und IT-Sicherheit, ein wenig E-Mail-Marketing und noch ein Schuss E-Commerce – das ist ein kurzer Überblick über die Themenvielfalt der aktuellen Wochen und Monate.

IT-Sicherheit

Das Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz) ist zum 25. Juli 2015 in Kraft getreten. Unmittelbare Auswirkungen für Website- und Webshop-Betreiber brachte es in Gestalt des § 13 Abs. 7 des Telemediengesetzes (TMG) mit (Wortlaut siehe Kasten).

Ungeachtet der teilweise vorkommenden unbestimmten Begrifflichkeiten, wie etwa »Stand der Technik«, »technisch möglich« oder »wirtschaftlich zumutbar«, lässt sich hieraus die Pflicht aller Betreiber von nicht nur rein privat ausgerichteten Internetpräsenzen zur Vorhaltung von bestimmten Mindeststandards ableiten. Das Betriebssystem des Webservers ist ebenso aktuell zu halten wie beispielsweise die PHP-Version, das Content-Management-System oder die eingesetzte Shop-Software.

Dies gilt gleichermaßen natürlich auch für andere EDV-Systeme. Ganz banal gesagt müssen regelmäßig an allen genutzten Rechnern etwa Windows-Patches eingespielt, Java- oder Flash-Updates durchgeführt und die Antivirus-Lösung gepflegt werden.

Was aber im Detail realisiert werden soll, um den Vorgaben des Gesetzgebers zu genügen, verrät § 13 Abs. 7 TMG leider nicht. Um diesem Umstand zu begegnen, gibt es inzwischen von verschiedenen Seiten Hilfe, insbesondere in Form einer EU-Richtlinie oder Anleitungen vom Bundesamt für Sicherheit in der Informationstechnik (BSI).

So wird die Richtlinie zur Netz- und Informationssicherheit (sogenannte NIS-Richtlinie) zum 8. August 2016 in Kraft treten und muss bis spätestens Mai 2018 in nationales Recht umgesetzt werden. Für Betreiber wesentlicher Dienste und Anbieter digitaler Dienste werden insbesondere neue IT-Sicherheitsanforderungen sowie Meldepflichten von Bedeutung werden. In die erste Kategorie fallen Betreiber kritischer Infrastrukturen, wie unter anderem Energieversorger. Unter den Begriff »digitale Dienste« fallen Online-Marktplätze, Suchmaschinen und Cloud-Computing-Dienste.

Für die Betreffenden gelten zwar nicht ganz so hohe Anforderungen wie für die wesentlichen Dienste, aber dennoch wird es um notwendige Anpassungsprozesse geben. So müssen auch sie angemessene Sicherheitsvorsorge treffen und Sicherheitsvorfälle melden.

Die EU-Elektroaltgeräterichtlinie WEEE regelt die fachgerechte Wiederverwertung von Altgeräten und deren Finanzierung durch Hersteller (Bild 1)

Das BSI hat mittlerweile ein Diskussionspapier »Absicherung von Telemediendiensten« herausgegeben. Dieses enthält zahlreiche Erläuterungen und Tipps, wie man das eigene Webangebot sinnvoll absichern kann. Dazu wird zwischen Content-, Host- und Access-Providern unterschieden. Je nach Art der Tätigkeit werden unterschiedliche Maßnahmen vorgeschlagen. Dazu zählen unter anderem die Wahl von sicheren Passwörtern, das zeitnahe Einspielen von Updates und Patches oder auch der Einsatz von Antivirus- und Firewall-Software. Hosting-Anbieter haben hier im Vergleich zu Website-Betreibern naturgemäß weitergehende Pflichten.

Elektroschrott

Schon zum 24. Oktober 2015 ist die Novelle des Elektro- und Elektronikgerätegesetzes (ElektroG) in Kraft getreten. Dies geschah in Umsetzung der EU-Elektroaltgeräterichtlinie »Waste Electrical and Electronic Equipment« (kurz: WEEE). Darin wird die fachgerechte (Wieder-)Verwertung von Altgeräten und deren Finanzierung durch Hersteller beziehungsweise Vertreiber in den EU-Mitgliedstaaten normiert (Bild 1).

Die im Gesetz vorgesehene Übergangsfrist ist zum 24. Juli 2016 abgelaufen, alle betroffenen Online-Händler müssen die gesetzlichen Vorgaben inzwischen also beachten und umsetzen. Es ist jedoch eine 400-Quadratmeter-Grenze vorgesehen, sodass nicht jeder kleine Einzelhändler in einem

§ 13 Abs. 7 des Telemediengesetzes (TMG)

Diensteanbieter haben, soweit dies technisch möglich und wirtschaftlich zumutbar ist, im Rahmen ihrer jeweiligen Verantwortlichkeit für geschäftsmäßig angebotene Telemedien durch technische und organisatorische Vorkehrungen sicherzustellen, dass
1. kein unerlaubter Zugriff auf die für ihre Telemedienangebote genutzten technischen Einrichtungen möglich ist und
2. diese
a) gegen Verletzungen des Schutzes personenbezogener Daten und
b) gegen Störungen, auch soweit sie durch äußere Angriffe bedingt sind,
gesichert sind.

Vorkehrungen nach Satz 1 müssen den Stand der Technik berücksichtigen. Eine Maßnahme nach Satz 1 ist insbesondere die Anwendung eines als sicher anerkannten Verschlüsselungsverfahrens.

Wust von ausgedienten Elektrogeräten ersticken muss. Nur für Hersteller und Vertreiber mit einer Lager- und Versandfläche beziehungsweise einer (stationären) Verkaufsfläche von über 400 qm besteht die Pflicht zur kostenfreien Rücknahme von Elektroaltgeräten; zudem existiert eine entsprechende Informationspflicht.

Alle Hersteller beziehungsweise Vertreiber von derartigen Produkten müssen Verbraucher über Folgendes informieren:

- Alte Elektrogeräte dürfen nicht im Hausmüll entsorgt werden, sie sind einer separaten Erfassung zuzuführen. Vorher sind gegebenenfalls Altbatterien beziehungsweise -Akkus zu entfernen und getrennt zu entsorgen (diesbezüglich besteht eine weitere Informationspflicht laut Batteriegesetz).
- Kunden müssen erfahren, dass und wie sie Elektroaltgeräte an den Händler zurückgeben können.
- Endnutzer sind selbst für das Löschen der auf den Geräten vorhandenen personenbezogenen Daten verantwortlich.
- Das Symbol der durchgestrichenen Mülltonne muss abgebildet und erläutert werden.

Diese Informationen sollten gut zugänglich und verständlich auf einer eigenen Unter-Seite platziert werden, auf die man per eigenständigem Menüpunkt geleitet wird. Ein sprechender Link auf diese Info-Seite sollte zudem in den Bestellablauf integriert werden, also auf jeder einzelner Produkt-Seite und auf der abschließenden Checkout-Seite. Zusätzlich müssen Hersteller ihre WEEE-Nummer im Impressum angeben.

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

E-Mail-Marketing

Eine Entscheidung des Oberlandesgerichts (OLG) in München (Urteil vom 27. September 2012, Az. 29 U 1682/12) hatte vor einiger Zeit für Aufsehen gesorgt, sprachen die Richter sich darin doch dafür aus, dass die Aufforderungs-Mail mit darin enthaltenem Aktivierungs-Link zur Verifizierung eines zukünftigen Empfängers von Werbe-Mails bereits als unzulässige Spam zu werten sei. Diese Auffassung würde dazu führen, dass das zwingende Double-Opt-in-Verfahren kaum sinnvoll durchzuführen ist. Nun hat ein anderes OLG, nämlich das in Düsseldorf, mit Urteil vom 17. März 2016 (Az. I-15 U 64/15) glücklicherweise die gegenteilige Ansicht vertreten.

Zwar stehen sich hier die wohl einzigen beiden OLG-Entscheidungen zu dieser Thematik konträr gegenüber, sodass im Grunde nach wie vor ein gewisses Maß an Rechtsunsicherheit besteht. Allerdings sind und werden diejenigen Stimmen lauter, welche die ältere Entscheidung aus München als Ausreißer werten. Eine Abweichung von der nach wie vor gängigen praktischen Umsetzung des Double-Opt-in-Verfahrens ist daher nicht zu empfehlen.

Online-Handel

Mit schöner Regelmäßigkeit gibt es gerichtliche Entscheidungen, die mehr oder weniger dringlich von Online-Händlern zu beachten sind. Auch im Jahr 2016 ist dies nicht anders. So hat etwa der Bundesgerichtshof (BGH) hinsichtlich der Angaben zur Energieeffizienzklasse geurteilt, die grundsätzlich bereits seit dem 1. Januar 2015 verpflichtend sind (Urteil vom 4. Februar 2016, Az. I ZR 181/14). Konkret ging es um die korrekte Platzierung des Etiketts, das kurz und eindeutig über die jeweilige Energieeffizienzklasse informieren soll, sowie des etwas ausführlicheren Produktdatenblatts. Es war bislang fraglich, wo und wie genau das Energie-Etikett und das Produktdatenblatt im Webshop unterzubringen sind.

Auf den jeweiligen Produkteinzelseiten müssen Grafiken in der Nähe sowie auch in der gleichen Größe der Preisangabe platziert werden. Hierbei ist nicht nur die eigentliche Energieeffizienzklasse, sondern auch die entsprechende Skala anzugeben. Ein Verweis auf das Produktdatenblatt kann per sprechendem Link erfolgen. ■

Michael Rohrlich
ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.
www.rechtssicher.info

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Monatliches Ranking

In 5 Jahren gibt es 25 Millionen Entwickler

Laut einer Prognose von Evans Data wird es im Jahr 2021 weltweit 25 Millionen Entwickler geben. Das stärkste Wachstum der Entwicklerzahlen wird für den Raum Asien und Pazifik sowie für Russland erwartet. Die Abschätzung stammt aus dem Global Developer Population and Demographics Report, der gerade in seine zwanzigste Auflage geht. Der Report schätzt die aktuelle Zahl der Software-Entwickler nicht nur über die vier Hauptregionen der Welt, sondern wagt auch einen Ausblick auf die Entwicklung der kommenden Jahre. So beträgt beispielsweise die Zahl der gegenwärtig an mobilen Applikationen arbeitenden Entwickler nach der Evans-Data-Schätzung 11,9 Millionen. Für 5,9 Millionen davon ist Android das primäre Betriebssystem, für das sie arbeiten, während 2,8 Millionen primär auf iOS zielen.

In der Auswertung der Jobangebote für Webentwickler in den 15 größten deutschen Städten fiel die starke Nachfrage in Frankfurt auf, das es diesmal mit 12,2 Prozent der Treffer auf den dritten Rang schaffte (Bild 1). Berlin musste sich in der aktuellen Auswertung (867 Treffer, das entspricht 17,9 Prozent) der bayerischen Landeshauptstadt München geschlagen geben (876 Treffer, 18,1 Prozent). Wie die Auswertung der Stellenangebote für Webentwickler nach Bundesländern zeigte, genügten die Angebote für Berlin darin nur für den fünften Rang (hier 11,3 Prozent) hinter Bayern (19,9 Prozent), NRW (18 Prozent), Baden-Württemberg (15,2 Prozent) und Hessen (11,9 Prozent). Auf die Plätze 6 und 7 kamen hier Hamburg mit 7,3 Prozent und Sachsen mit 4,2 Prozent.

Für Entwickler von mobilen Apps ist Berlin sogar ein noch besserer Standort. Hinter Bayern (744 Treffer) und NRW (587 Treffer) lag die Bundeshauptstadt in dieser Auswertung (Bild 2) mit 471 Treffern auf dem dritten Rang.

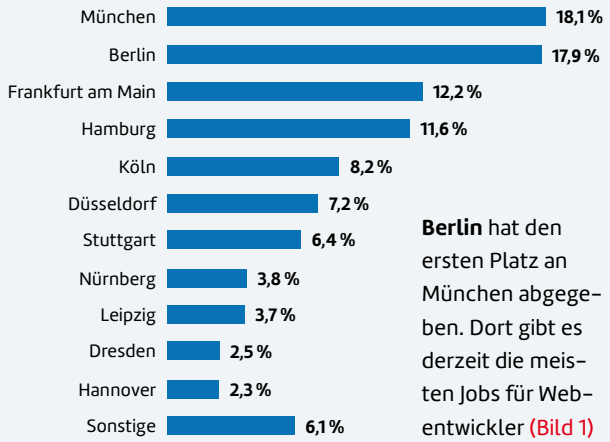
In Tabelle 1 finden Sie die Reihenfolge der in den Stellenanzeigen der letzten vier Wochen vor dem Erhebungsstichtag (1. August) enthaltenen Technologie-Begriffe. Abgefragt wird dafür die Datenbank der Meta-Suchmaschine Jobkralle.de. Weiterhin das mit Abstand wichtigste Technologie-Thema in Stellenanzeigen ist die Cloud. Big Data liegt auf Rang 5 hinter MySQL, HTML5 und SharePoint. iOS ist wieder ganz knapp hinter Android zurückgefallen. Die Oberflächentechnologie WPF wurde in 943 Einträgen der Jobdatenbank gefunden und damit knapp häufiger als das Stichwort Responsive Web (888 Treffer). Abgefragt wurden übrigens auch die Stichwörter Swift + iOS, Windows Phone und Windows Forms. Sie lieferten allerdings jeweils weniger als 300 Treffer und schafften es damit nicht in die Auswertung der Top 16.

Tabelle 1: Technologien

Rang	Technologie	Anteil *
1	Cloud	18,9 %
2	MySQL	11,5 %
3	HTML5	9,9 %
4	SharePoint	8,0 %
5	Big Data	7,6 %
6	Android	7,0 %
7	iOS	7,0 %
8	Microsoft SQL Server	5,9 %
9	CSS3	4,6 %
10	AngularJS	4,0 %
11	Windows 10	3,8 %
12	WPF	2,8 %
13	Responsive Web	2,6 %
14	NoSQL	2,5 %
15	Azure	2,4 %
16	WCF	1,5 %

* Prozentualer Anteil der Treffer

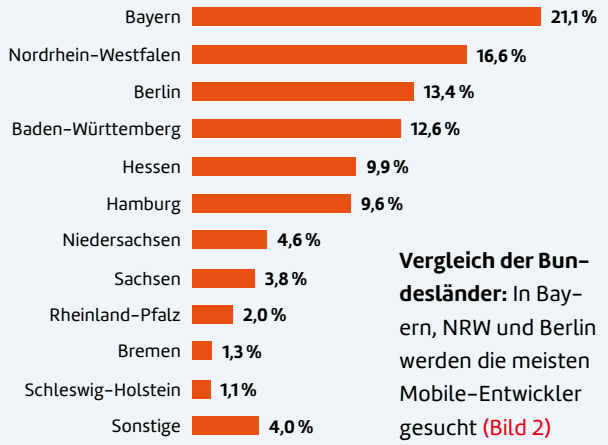
Jobangebote für Webentwickler



Berlin hat den ersten Platz an München abgegeben. Dort gibt es derzeit die meisten Jobs für Webentwickler (Bild 1)

web & mobile developer 10/2016 Quelle: Eigene Recherchen, Jobkralle.de

Jobs für Mobile-Entwickler



Vergleich der Bundesländer: In Bayern, NRW und Berlin werden die meisten Mobile-Entwickler gesucht (Bild 2)

web & mobile developer 10/2016 Quelle: Eigene Recherchen, Jobkralle.de

Zahl des Monats

Die ITK*-Umsätze in den EU-Staaten legen 2016 um **0,7 %** auf **686 Milliarden Euro** zu (2015: Plus 2,9 %). Die Umsätze mit IT-Hardware/-Dienstleistungen und Software steigen um **2,7 %** auf **388 Milliarden Euro**, die TK-Umsätze gehen zurück.

* ITK = Informationstechnologie und Telekommunikation

Quelle: Prognose des Bitkom

Bitkom

Am besten unter 30

Fast drei Viertel aller Unternehmen (72 Prozent) ergreifen besondere Maßnahmen, um Mitarbeiter unter 30 Jahren zu rekrutieren oder zu halten. Das ist das Ergebnis einer repräsentativen Unternehmensbefragung im Auftrag des Digitalverbands Bitkom. Mehr als jedes zweite Unternehmen (58 Prozent) gibt an, immer die neueste Gerätegeneration an Smartphones, Tablet-Computern und Notebooks zur Verfügung zu stellen, um jüngere Mitarbeiter zu binden.

Etwa ebenso viele (55 Prozent) setzen für die Rekrutierung Jüngerer vor allem auf Online-Netzwerke wie Xing oder LinkedIn sowie auf Social-Media-Kanäle. Jedes dritte Unternehmen (33 Prozent) bietet flexible Arbeitsmodelle, um bei der jungen Generation zu punkten. 16 Prozent versuchen, die jungen Mitarbeiter mit einer lockeren Arbeitsatmosphäre sowie Gemeinschaftssinn zu überzeugen und machen dazu ihren Beschäftigten zum Beispiel spezielle Wellness-Angebote (Bild 3).

»Gerade in der digitalen Wirtschaft sind junge Leute, die häufig über eine besonders hohe Digitalaffinität verfügen, sehr umworben und haben daher oft eine große Auswahl unter den Arbeitgebern«, sagt Bitkom-Hauptgeschäftsführer Dr. Bernhard Rohleder. Das stellt die Personaler vor große Herausforderungen. Rohleder: »Unternehmen, die die besten Nachwuchskräfte gewinnen wollen, müssen auch etwas zu bieten haben. Gerade bei der jungen Generation zählen dabei neben dem Gehalt auch weiche Faktoren.«

TNS Infratest, Microsoft

Wünsche an den Chef

Wie muss sich Führung in der digitalen Arbeitswelt verändern? Was müssen Manager in Zukunft leisten? Und wie beeinflusst der Einsatz moderner IT das Verhältnis zwischen Chef und Mitarbeitern? Um das herauszufinden, hat TNS Infratest im Auftrag von Microsoft Beschäftigte befragt, was sie von ihren direkten Vorgesetzten erwarten. Die Mehrheit hätte gern besseren Zugang zu Informationen und regelmäßigeres Feedback, sie möchte Entscheidungen selbstständiger treffen und flexibler arbeiten. Gleichzeitig wünschen sie sich mehr Unterstützung von ihren Chefs (Bild 4).

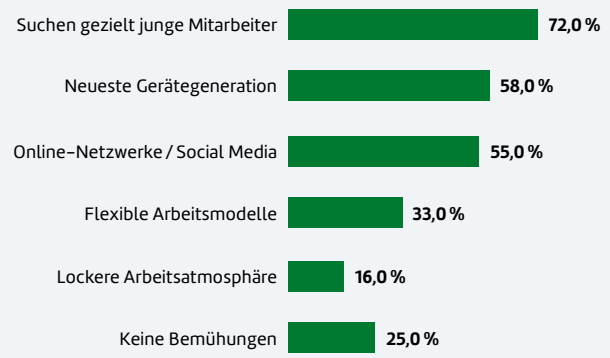
Markus Köhler, Senior Director Human Resources bei Microsoft Deutschland: »Damit Unternehmen schneller auf Marktveränderungen reagieren können, müssen Führungskräfte mehr Verantwortung in die Teams abgeben, die nahe am Markt und am Kunden agieren, und Mitarbeitern mehr Freiraum für eigene Entscheidungen überlassen. Manager sollten in Zukunft mehr coachen und weniger kontrollieren.« Das wünschen sich auch die Mitarbeiter selbst, allerdings ist nicht einmal jeder zweite Mitarbeiter (41 Prozent) mit seinem Vorgesetzten in seiner Rolle als Coach und Mentor sehr oder gar vollkommen zufrieden.

PaperC

Bewerbungsratgeber leihen statt kaufen

Der Online-Dienst PaperC stellt mehr als 146.000 E-Books zum Ausleihen bereit, darunter auch Bewerbungsratgeber, beispielsweise die Titel von Duden »Professionelles Bewerben«, »Die Bewerbungsmappe« oder »Die 50 wichtigsten Arbeitgeberfrage im Bewerbungsgespräch«. Die Preise für die Leihe liegen rund 50 Prozent unter den Kaufpreisen, man kann dabei bis zu 15 Euro gegenüber dem Print-Preis sparen und sich somit günstig sechs Monate lang in ein Bewerbungsthema einlesen – das ist eine passende Zeitspanne, um potenzielle Arbeitgeber zu überzeugen.

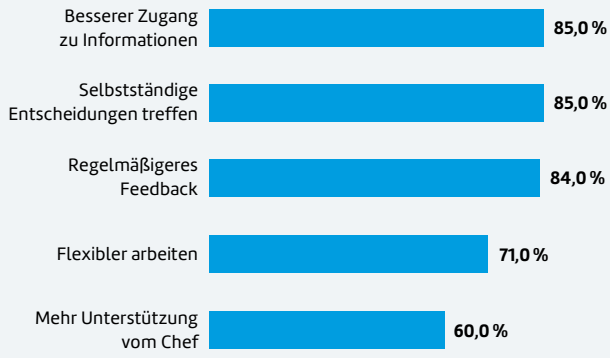
Junge Mitarbeiter anlocken



Fast drei Viertel der Unternehmen suchen gezielt junge Mitarbeiter, zumeist in Online-Netzwerken (Bild 3)

web & mobile developer 10/2016 Quelle: Bitkom. Mehrfachnennungen

Erwartungen an den Chef



Die Wünsche an den Vorgesetzten sind eindeutig: Informationen, Feedback und Selbstständigkeit sind gefragt (Bild 4)

web & mobile developer 10/2016 Quelle: TNS Infratest, Microsoft, Mehrfachng.

Stellenmarkt

dotnetpro + web & mobile Developer

- 25.800 Exemplare Gesamtauflage
- 25.300 Newsletter-Empfänger
- 66.600 PI'S

.NET Architektur HTML5/JavaScript iOS/Android

Kontakt:
Jens Schmidtmann, Klaus Ahlering • Tel. 089/74117-125 • sales@nmg.de

Anbieterverzeichnis

für Deutschland, Schweiz und Österreich.

Consulting / Dienstleister

ANEXIA Internetdienst-leistungs GmbH
Feldkirchner Straße 140
9020 Klagenfurt / AUSTRIA
T +43-50-556
F +43-50-556-500
info@anexia-it.com

ANEXIA wurde im Juni 2006 von Alexander Windbichler als klassischer Internet Service Provider gegründet. In den letzten Jahren hat sich ANEXIA zu einem stabilen, erfolgreichen und international tätigen Unternehmen entwickelt, das namhafte Kunden rund um den Globus mit Standorten in Wien, Klagenfurt, München, Köln und New York City betreut. ANEXIA bietet ihren Kunden hochwertige und individuelle Lösungen im Bereich Web- und Managed Hosting, sowie Individualsoftware und App Entwicklung.

---

prodot GmbH
Schifferstraße 196
47059 Duisburg
T: 0203 - 346945 - 0
F: 0203 - 346945 - 20
info@prodot.de
https://prodot.de

prodot - Software für Marktführer
Intelligente Software für internationale Konzerne und mittelständische Unternehmen: prodot stärkt seit über 15 Jahren namhafte Kunden im weltweiten Wettbewerb – mit effizienten, stabilen und kostensenkenden Lösungen.
Kunden schätzen unsere Kreativität. Mit Sorgfalt und Enthusiasmus entwickeln wir hochwertige Software. Digitale Prozesse und innovative Technologien sind unser Antrieb, Fortschritt und Kontinuität unser Anspruch.
ALDI SÜD, Microsoft und Siemens vertrauen uns bereits viele Jahre.
Gerne zeigen wir Ihnen warum! Sprechen Sie mich an. Pascal Kremmers.

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.

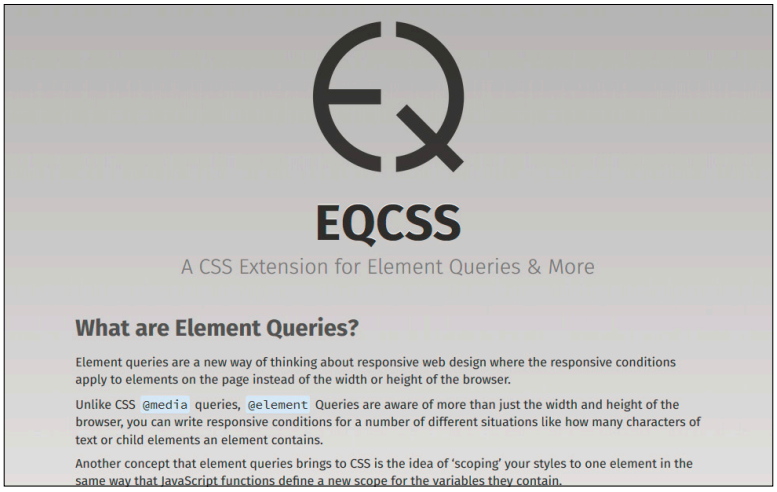
Web- / Mobile-Entwicklung & Content Management

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 11/2016 erscheint am 13. Oktober 2016

Responsives Design mit Element Queries



Im Allgemeinen gelten Media Queries als wichtigste Komponente für responsive Webseiten. Dank Media Queries lassen sich eigene CSS-Angaben für verschiedene Viewport-Größen machen. Das funktioniert sehr gut für das grundlegende Layout, schwieriger wird es aber für kleinere Komponenten. Für deren optimale Anzeige sind oft nicht die Viewport-Größen ausschlaggebend, sondern die sie direkt umgebenden Elemente. In diesem Fall kommen wir mit Media Queries nicht weiter, weil die zu ändernde Darstellung nicht von der Viewport-Größe abhängt. Genau das ist ein typischer Fall für Element Queries. Element Queries sind schon länger im Gespräch – und inzwischen gibt es eine schöne JavaScript-Bibliothek namens EQCSS, die es erlaubt, heute schon Element Queries einzusetzen.

Reaktive Programmierung

Der Begriff der reaktiven Programmierung dürfte so manchem Webentwickler schon begegnet sein. Doch was verbirgt sich eigentlich hinter diesem Programmierparadigma? Ein Artikel gibt eine detaillierte Einführung in die reaktive Programmierung mit JavaScript und erläutert anhand einiger Beispiele und eines konkreten JavaScript-Frameworks wie beispielsweise RxJS die Prinzipien dieser Programmiermethode.

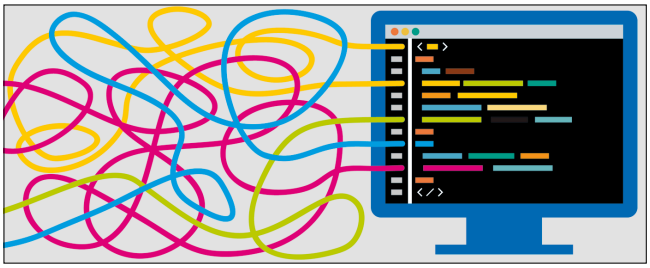
WebComponents

Dieser Artikel stellt die Vorteile komponentenorientierter Webentwicklung vor. Es wird sowohl das Konzept an sich erklärt als auch die diversen Ansätze und Frameworks, die das ermöglichen. Einerseits sind das natürlich WebComponents, aber auch React und Angular2 sind dafür anwendbar. Der Leser bekommt einen Überblick über die Methodik und Wegweiser. Auch wird der Artikel aufzeigen, wie der WebComponents-Standard gerade im Fluss ist.

Preact vs. React

Preact ist eine extrem kleine Implementierung von React, dem von Facebook entwickelten JavaScript-Framework. Der Artikel stellt die Bibliothek vor und zeigt, wie man sie einsetzt. Außerdem wird ein detaillierter Vergleich mit dem Original angestellt. Wo liegen Unterschiede vor? Wann sollte man welche Bibliothek einsetzen? Dabei wird auch besonders Wert auf komponentenorientierte Entwicklung gelegt.

dotnetpro



Ausgabe 9/2016 ab 18. August 2016 am Kiosk

Im Schwerpunkt der dotnetpro: Neue kleine und schnelle Editoren, die im Wesentlichen nur für das Bearbeiten von Code zuständig sind, versus All-inclusive-IDEs, die nicht nur das Editieren, sondern auch das Verwalten, Testen, Bauen und Verteilen übernehmen.
www.dotnetpro.de

Unsere digitalen Angebote

Wöchentlicher Newsletter
webundmobile.de/newsletter

Stellenmarkt
stellenmarkt.webundmobile.de

YouTube
youtube.com/user/developermedia

Facebook
facebook.com/webundmobile

Google +
gplus.to/webundmobile

Twitter
twitter.com/webundmobile

developer media

developer media Trainings

Updates für Ihr Know-How

„Das Entwickeln von qualitativ hochwertiger Software bedeutet auch, sich selbst ständig weiterzuentwickeln und nicht still zu stehen."

David Tielke
Softwareentwickler, Berater, Trainer

Codequalität mit JavaScript

Trainer: Golo Roden

3 Tage, 02.-04.11.2016, Köln
Ab 2.199 EUR zzgl. MwSt.

Moderne Webentwicklung mit ASP.NET Core

Trainer: David Tielke

3 Tage, 21.-23.11.2016, Köln
Ab 2.199 EUR zzgl. MwSt.

Hybrid-Apps mit Ionic, Cordova und Angular

Trainer: Hendrik Lösch

2 Tage, 14.-15.11.2016, Köln
Ab 1.799 EUR zzgl. MwSt.

Angular 2 mit TypeScript

Trainer: Johannes Hoppe/ Gregor Woiwode

3 Tage, 21.-23.11.2016, Köln
Ab 2.199 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

developer-media.de/trainings

Ihr Partner für mehr Online-Wachstum

Wir planen, entwickeln und steuern
Websites, **Apps** und **Kampagnen**.

Unsere Ziele sind Ihre Ziele:

- **Mehr Sichtbarkeit**
- **Mehr Traffic**
- **Mehr Leads**
- **Mehr Conversions**

- **Mehr Kunden**

Besuchen Sie uns unter
www.digitalmobil.com

digitalmobil