web & mobile DEVELOPER

webundmobile.de

Graph-Datenbank mit Java programmieren

So nutzen Sie die neuen Features der Graph-Datenbank Neo4j zur Programmierung von Datenbank-Applikationen mit Java ab S. 18

Amazon Alexa Skills Kit

Tam Hanna zeigt, wie Entwickler mit dem Amazon Alexa Skills Kit eigene Dienste in Alexa integrieren können S. 80

Ausgabe 12/16

Deutschland 14,95 EUR

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

4 198255 014951 12

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

NoSQL mit Java

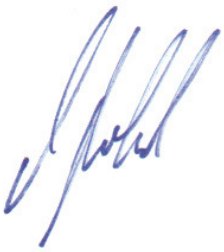
Der Schwerpunkt dieser Ausgabe bietet Know-how in Sachen Java und Graphdatenbank Neo4j. Navigation in Zeiten des Responsive Designs ist ein weiteres Thema.

Neo4j bietet dem Entwickler eine Vielzahl von Java-APIs und damit umfassende Wahlfreiheit bei der Lösung einer konkreten Problemstellung. Als Datenbank-Management-System (DBMS) für Graphen besitzt Neo4j Schnittstellen für alle in der Praxis relevanten Programmiersprachen. Bei der Implementierung des DBMS entschied sich der Hersteller Neo Technology jedoch von Anfang an für Java. Damit rückte Java, wie bereits aus dem Produktnamen der Datenbank durch die Endung 4j ersichtlich, als prädestinierte Technologie ins Zentrum des DBMS. Alsbald unterstützte Neo Technology den Architekturstil REST (Representational State Transfer) und für Java EE das Spring-Data-Projekt. Welche Möglichkeiten die Graphdatenbank Neo4j Java-Entwicklern bietet, erläutert ein Artikel von Frank Simon ab Seite 18.

»Man kann per PHP und Raspberry Pi Lichter zum Leuchten bringen.«

Eine gut durchdachte Navigation sorgt dafür, dass die einzelnen Inhalte und Unterseiten schnell gefunden werden. Aufgrund der Vielzahl an Seiten, die erreichbar sein sollen, ergeben sich bei Webseiten oft relativ komplexe Menüs. Bei Desktop-Versionen behilft man sich dann mit verschachtelten Navigationen oder nutzt mehrere Navigationen wie Haupt- und Nebennavigation. Ein solches Menü in derselben Form in der Smartphone-Variante anzeigen zu lassen ist keine gute Idee – dafür fehlt dort schlicht der Platz. Also sind Anpassungen erforderlich. Welche, das erläutert ein Artikel von Dr. Florence Maurice ab Seite 32.

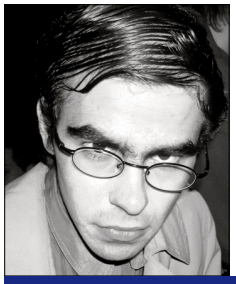
Programmierer, die einen Raspberry Pi besitzen und Erfahrung mit der Programmiersprache PHP haben, können ihre Kenntnisse auch auf diesem beliebten und preiswerten Mini-Computer einsetzen. Das eröffnet eine neue Welt: Man kann per PHP Lichter zum Leuchten bringen, Geräte einschalten oder den Input von Sensoren erfassen und das Ganze dann per Webtechnik von überall im Haus nutzen. Alles zum Thema PHP und Raspberry Pi erläutert Markus Schraudolph in seinem Artikel ab Seite 94.



Ihr Max Bold
chefredakteur@maxbold.de

Philip Ackermann

erläutert, wie per Messaging in JavaScript Komponenten entkoppelt werden (S. 38)



Tam Hanna

stellt die neuen Funktionen von Android Studio 2.2 im Detail vor (S. 64)

Frank Pientka

zeigt, wie man mit dem Apache Tomcat Webserver effizient arbeitet (S. 100)

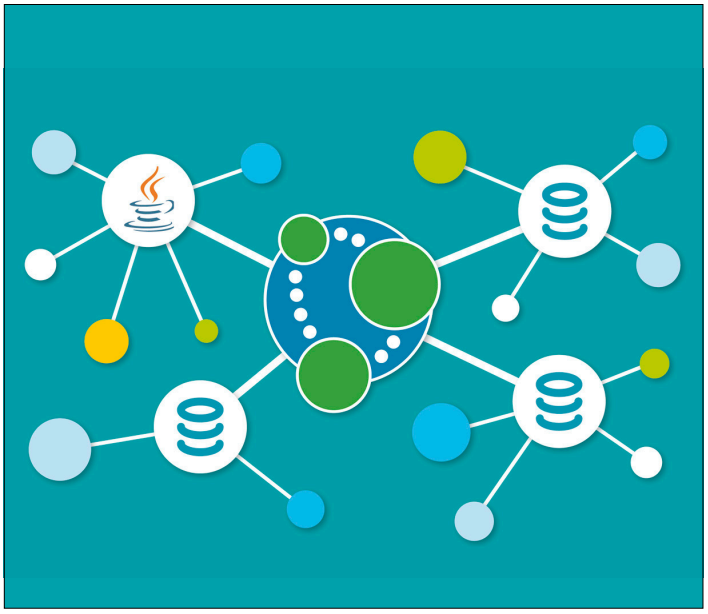
INHALT

Update

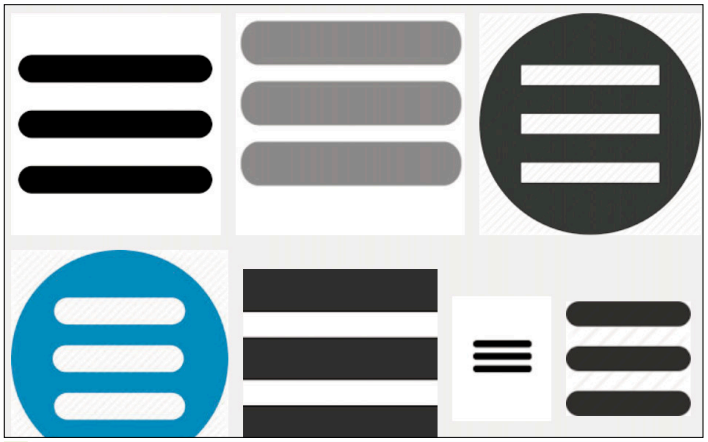
Feature

HTML, CSS & JavaScript

Mobile Development



Als Datenbank-Management-System für Graphen besitzt Neo4j Schnittstellen für alle relevanten Programmiersprachen 18

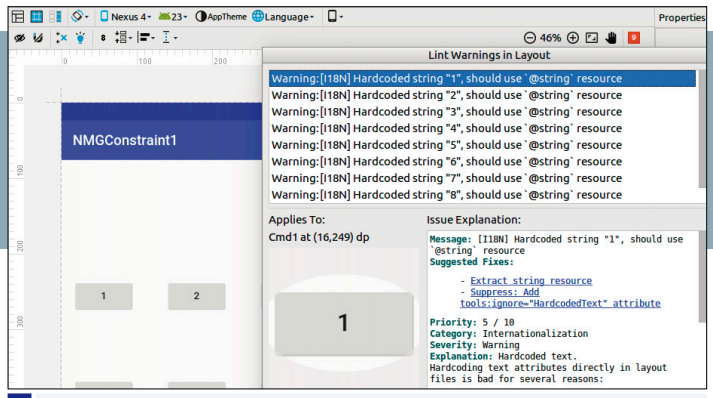


Komplexe Navigationen auf kleine Screens umzusetzen erfordert im Responsive Webdesign durchdachte Anpassungen 32

Experten in dieser Ausgabe

Markus Schraudolph erklärt in einem Artikel das interessante Zusammenspiel von PHP und Raspberry Pi 94

Jens Geyer zeigt, mit welchen Techniken und Methoden man erfolgreich Software entwickeln kann 122



Die neue Version Android Studio 2.2 bietet dem Entwickler von Android-Apps zahlreiche interessante Funktionen an **64**

Foto: Fotolia / Sashkin

In Omnichannel-Modellen sichert das reibungslose Management von Bestell-, Bestands- und Fulfillment-Informationen über eine Vielzahl von Kanälen den Unternehmenserfolg **90**

Adobe Photoshop CC 2015 liefert umfangreiche Werkzeuge und Funktionen, um Bildobjekte auszuwählen, diese freizustellen und zu bearbeiten **106**

Backend

Beyond Dev

Standards

NEWS & TRENDS

AKTUELLE NEWS FÜR ENTWICKLER

E-Commerce-Markt in Deutschland

Viele Kunden sind mit ihren Providern unzufrieden

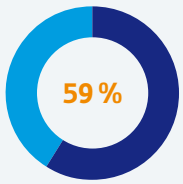
Mehr als jeder fünfte deutsche Kunde (22 Prozent) ist mit einem seiner Telefon-, Internet- oder Hosting-Anbieter unzufrieden. Das sind mehr als 10 Millionen Bundesbürger.

Zu diesem Ergebnis kommt die Verbraucherstudie »Telefonie, Internet & Webhosting: Die Deutschen und ihre Anbieter«, die Forsa im Auftrag des Berliner Internetproviders Strato durchgeführt hat.

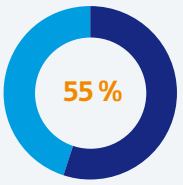
»Die Studienergebnisse sollten alle Anbieter der ITK-Branche wachrütteln: Wir müssen vieles ändern. Es kann nicht sein, dass wir unzufriedene Kunden mit langen Prozessen von der Kündigung abhalten. Die Anbieter müssen fairer werden und Prozesse einfacher gestalten, damit Kunden zufriedener sind«, sagt Dr. Christian Böing, CEO von Strato.

Gründe für die Unzufriedenheit von Kunden sind in erster Linie die Qualität des Dienstes (59 Prozent), das Preis-Leistungs-Verhältnis (55 Prozent) sowie die Kompetenz (27 Prozent) und Erreichbarkeit (23 Prozent) des Kundenservice. Weitere Fallstricke und Gründe für Unzufriedenheit lauern laut Studie in allen Phasen der Geschäftsbeziehung – von der Bestellung über den Service während der Vertragslaufzeit bis zur Kündigung.

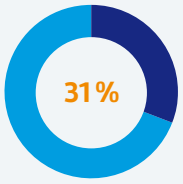
www.strato.de



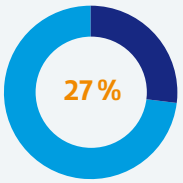
Qualität



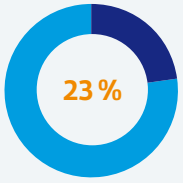
Preis-Leistungs-Verhältnis



Kein Wechsel in günstigeren Tarif während der Laufzeit



Kompetenz des Kundenservice



Erreichbarkeit des Kundenservice

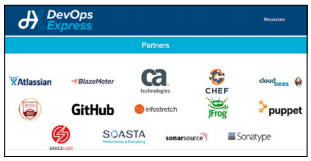
Quelle: Strato

Viele Kunden sind mit ihrem Telefon-, Internet- oder Hosting-Anbieter unzufrieden

Sonatype und CloudBees

DevOps-Express-Initiative gestartet

Sonatype hat heute den Start von DevOps Express angekündigt, einer Brancheninitiative, die einfachen Zugang zu einer Reihe von integrierten und bewährten DevOps-Lösungen bietet, um die Wertschöpfung in Unternehmen zu beschleunigen. Bis dato waren Käufer gezwungen, Best-in-breed-Lösungen selbst zu beurteilen, zusammenzusetzen und diese zu unterstützen. Als Gründungsmitglied von DevOps Express kooperierte Sonatype mit CloudBees, um einen Ansatz zu entwickeln, der Unternehmen den Erwerb, die Konzeption, die Integration sowie den Support nativer DevOps-Tools erleichtert. An der Initiative sind weitere Branchenführer wie Atlassian, BlazeMeter, CA Technologies, Chef, DevOps Institute, GitHub, Infostretch, JFrog, Puppet Labs, Sauce Labs, Soasta und SonarSource beteiligt.



Die Initiative will die Kundenzufriedenheit mit nativen DevOps-Lösungen verbessern

»Kein einziger Technologie-Anbieter bietet eine End-to-End-DevOps-Lösung. Darum ist die DevOps-Express-Initiative eine maßgebliche Entwicklung innerhalb der Branche«, erklärt Wayne Jackson, CEO bei Sonatype. »Bisher wurde in DevOps-Praktiken immer wieder betont, wie wichtig die Zusammenarbeit innerhalb einer Organisation ist, um die Software-Bereitstellung zu beschleunigen. Sonatype und die anderen DevOps-Express-Partner beziehen in diese Zusammenarbeit nun auch die Beschaffungswege mit ein.«

Im Rahmen der DevOps-Express-Initiative stellt Sonatype der Community insgesamt zwölf Integrationen zur Verfügung. Diese Integrationen umfassen: Atlassian (JIRA Software, Bitbucket Pipelines, Bamboo, HipChat, Crowd), CA Technologies (CA Release Automation und Action Packs), CloudBees (Jenkins und das neue Nexus Jenkins Plug-in), Puppet (automatische Installation und Konfiguration des Nexus Repository) und SonarSource (SonarQube)

Neben den Produktintegrationen hat Sonatype auch die Vorreiterrolle unter den DevOps-Express-Partnern übernommen, um eine erste Reihe von 31 DevOps-Referenz-Architekturen zu liefern. Jede dieser Architekturen offenbart reale Implementierungen, die Tools der beteiligten DevOps-Express-Mitglieder beinhalten.

»DevOps Express wurde entwickelt, damit Unternehmen ihre Software-Entwicklung und -Lieferung einfacher beschleunigen können«, schildert Sacha Labourey, CEO und Gründer von CloudBees. »In Zusammenarbeit mit Sonatype und anderen führenden Technologie-Anbietern sind wir in der Lage, neue Wege zu gehen, um Unternehmen mit integrierten DevOps-Lösungen zu versorgen.«

Zahl des Monats

Gut über **30 Prozent** Marktanteil hat Amazon bei Cloud-Infrastruktur-Services, das ist mehr als die drei stärksten Konkurrenten Microsoft, IBM und Google zusammen.

Quelle: Synergy Research Group

»Jede Organisation, die sich in Richtung Continuous Delivery und auf die DevOps-Ära zubewegt, wird von der organischen Zusammenarbeit der Branche, die DevOps Express exemplifiziert, profitieren«, erklärt Donnie Berkholz, Research Director for Development, DevOps & IT Ops bei 451 Research. »Anstatt Unternehmen zu zwingen, ihren eigenen Weg durch den Wandel zu finden, übernehmen die Gründungsmitglieder von DevOps Express eine aktive Rolle, um eine Reihe bewährter DevOps-nativer Lösungen zu identifizieren, zu integrieren und zu unterstützen.«
www.cloudbees.com/devops-express

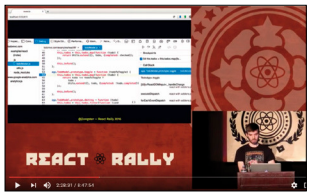
Debugger.html

Neuer JavaScript-Debugger für Firefox

Das neue Projekt des Firefox-DevTools-Teams trägt den Namen Debugger.html. Dabei handelt es sich um einen Open-Source-JavaScript-Debugger für Firefox und das Web. Debugger.html wurde als komplett neues Projekt aufgesetzt, basiert auf React- und Redux-Web-Architektur und wird auf eine neue Art und Weise gepflegt. Es wird direkt mit Firefox ausgeliefert und kann von Webentwicklern schnell und einfach verwendet werden.

Die Firefox DevTools wechseln aktuell nach und nach von XUL zu einer moderneren Architektur, die aus wiederverwendbaren React-Komponenten und einem Redux-Speichermodell besteht. Mit diesem Ansatz wird Code in kleinere Module verschlankt. Mozilla ist nach eigener Aussage davon überzeugt, dass der Debugger und alle zukünftig verfügbaren Developer-Tools dadurch zugänglicher, berechenbarer, verständlicher und leichter überprüfbar werden.

Mit Debugger.html möchte Mozilla eine verbesserte und insgesamt vertrautere Umgebung für Webentwickler schaffen. Auch wenn Mozilla die Entwicklung für Firefox als oberste Priorität nennt, kümmert sich das Team auch um andere Plattformen. Zwar gibt es noch keine entsprechende produktionsreife Version, aber in Zukunft soll der Debugger auch zusammen mit dem Chrome Debugging Protocol für Chrome-Tabs und Node-Prozesse verwendet werden können.
www.mozilla.org



Das DevTools-Team von Mozilla hat einen JavaScript-Debugger für Firefox vorgestellt

IBM-Studie

Die hybride Cloud macht das Rennen

Eine IBM-Studie besagt, dass rund 80 Prozent der Unternehmen die Cloud nutzen, und zwar meist in hybrider Form sowie maßgeschneidert auf individuelle Anforderungen. ▶

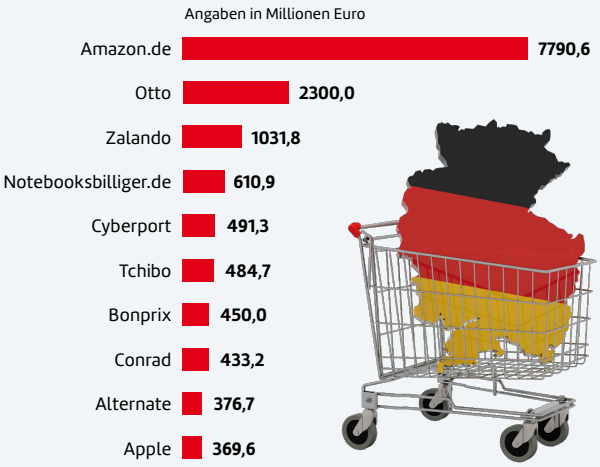
Verbraucherstudie

Amazon, Otto und Zalando beherrschen den Online-Handel

Die Studie »E-Commerce-Markt Deutschland 2016« von EHI Retail Institute/Statista macht deutlich, dass drei Große den E-Commerce in Deutschland praktisch unter sich ausmachen: Der Löwenanteil des deutschen Online-Umsatzes fällt auf die Top 3: Amazon.de (7,8 Milliarden Euro), Otto.de (2,3 Milliarden Euro) und Zalando.de (1,0 Milliarden Euro) erzielen gemeinsam über 11 Milliarden Euro Umsatz – das ist annähernd so viel wie die Shops auf den Rängen 4 bis 100 zusammen. Diese hohe Konzentration nimmt über die letzten Jahre weiter zu.

Dennoch war 2015 reichlich Bewegung im Online-Markt. So konnten sich einige etablierte Shops im Ranking nach vorne entwickeln, beispielsweise Mediamarkt.de und Saturn.de. Auch Shootingstars haben sich vorne im Ranking platziert. Die Otto Group stellt einen der Aufsteiger unter den umsatzstärksten Onlineshops in Deutschland: Aboutyou.de katapultiert sich gut eineinhalb Jahre nach Start auf Platz 70 mit einem Jahresumsatz von fast 70 Millionen Euro.

Erstmals befinden sich zwei Online-Händler mit Lebensmittel-Fokus unter den Top 100. Das Gros besteht hier allerdings auch in diesem Jahr aus Generalisten und Fashion-Onlineshops: 29 Shops bieten ihren Kunden ein Online-Kaufhaus ohne speziellen Produktfokus, 20 Shops verkaufen hauptsächlich Bekleidung, Textilien und Schuhe. Mit 12 Anbietern ist der Bereich Computer, Unterhaltungselektronik und Telekommunikation am drittstärksten vertreten. Sonst zeigen sich die umsatzstärksten Onlineshops in Deutschland erneut divers, es gibt Shops aus 15 weiteren Branchen.
http://de.statista.com



Im Jahr 2015 gab es reichlich Bewegung im Online-Markt

web & mobile developer 12/2016 Quelle: EHI Retail Institute/Statista

Steigende Angriffszahlen

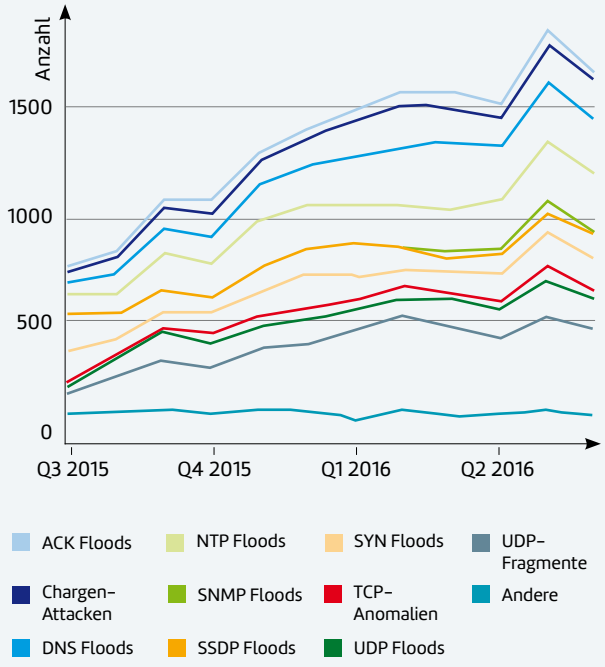
Akamais Security Report

Akamai hat seinen State of the Internet Security Report für das zweite Quartal 2016 veröffentlicht. Die wichtigsten Ergebnisse: Die Anzahl der DDoS-Angriffe steigt im Vergleich zum Vorjahr weiterhin kontinuierlich an. Brasilien löst die USA als Ursprungsland der meisten Angriffe auf Webanwendungen ab.

Der Quartalsbericht bietet Analysen und Einblicke in cyberkriminelle Aktivitäten, die auf der Akamai Intelligent Platform beobachtet wurden. Er enthält eine ausführliche Analyse sowie eine detaillierte Sicht auf die weltweite Cloud-Security-Bedrohungslandschaft. Insbesondere werden Trends bei DDoS- und Webanwendungsangriffen sowie schädlichem Bot-Datenverkehr analysiert. DDoS-Angriffe nahmen im zweiten Quartal 2016 gegenüber dem Vergleichsquartal 2015 um 129 Prozent zu. Im zweiten Quartal wehrte Akamai insgesamt 4.919 DDoS-Angriffe ab. Zwölf der während des zweiten Quartals verzeichneten Angriffe überstiegen 100 GBit/s.

»Zwar nimmt der Umfang der DDoS-Angriffe ab, die Angriffszahlen steigen jedoch weiterhin kontinuierlich an, da Angriffs-Tools immer effektiver, zugänglicher und damit profitabler für Hacker werden«, betont Martin McKeay, Editor-in-Chief State of the Internet / Security Report bei Akamai. »Die zunehmende Verbreitung der Tools führt zu wiederholten Angriffen auf Unternehmen, denen sie allein nicht gewachsen sind. Mit Blick auf den Cyber-Security-Awareness-Monat im Oktober in den USA ist es von entscheidender Bedeutung, dass Unternehmen verstehen, womit sie es zu tun haben. Ganz besonders deshalb, weil Angreifer immer öfter mit DDoS-Angriffen Lösegeld erpressen.«

http://stateoftheinternet.com/security-report



DDoS-Angriffe steigen weiterhin kontinuierlich an

Quelle: Akamai Faster Forward

Vier von fünf Unternehmen weltweit nutzen heute die Cloud, jedoch nicht ausschließlich, sondern angepasst auf dedizierte Anforderungen, das zeigt die jüngste Cloud-Studie von IBM. Vor vier Jahren gab nur etwa ein Drittel der Unternehmen an, aktiv eine Cloud zu nutzen. Vorherrschend ist bei den Unternehmen das hybride Modell, bei dem die IT aus eigenen Ressourcen kommt und Private und Public Clouds kombiniert. Zudem ergab die Studie, dass auch zukünftig etwa die Hälfte der IT-Anwendungen on premise bleiben wird, also auf eigenen Servern läuft.

Studie: 80 Prozent der Unternehmen nutzen die Cloud

Unter dem Studientitel »Tailoring hybrid cloud: Designing the right mix for innovation, efficiency and growth« hat IBM untersucht, wie die maßgeschneiderte Cloud der Unternehmen aussieht. Herausgekommen ist, dass vor allem die hybride Form zum Einsatz kommt, um die digitale Transformation voranzutreiben.

Als zentrale Herausforderungen für Cloud-Projekte identifizierte die IBM Studie drei Aspekte: Risiken für Sicherheit, Compliance und Anforderungen (47 Prozent), Kostenstruktur (41 Prozent) sowie das Risiko für IT-Ausfälle aufgrund neuer Cloud-Services (38 Prozent). Aufgrund der Studienergebnisse gibt IBM drei Empfehlungen, um die Cloud erfolgreich einzuführen: Erstens, im Unternehmen ein tiefes Verständnis dafür zu entwickeln, wie sich die Cloud auf Geschäftsentwicklung und Finanzen auswirkt. Als Zweites gilt es, das komplexe Zusammenspiel von mehreren Cloud-Partnern erfolgreich zu managen. Als Drittes rät IBM, Vorgaben für Sicherheit und Regulierungen durch zusätzliche interne Kapazitäten sowie externe Lösungen zu unterstützen.

http://ibm.biz/tailoringcloud

Thyssenkrupp

HoloLens für 24.000 Servicetechniker

Thyssenkrupp will HoloLens und Skype in seine Wartungslösung MAX integrieren. Wartungsarbeiten sollen sich damit bis zu viermal schneller erledigen lassen als früher. Mit MAX wartet das Unternehmen auf der Basis der Microsoft-Cloud-Plattform Azure seit 2015 Tausende von Aufzügen in der ganzen Welt. Für mehr als 24.000 Servicetechniker von Thyssenkrupp bedeutet der Einsatz der HoloLens als dem weltweit ersten eigenständigen holografischen Computer eine Revolution des Arbeitsalltags.

Mit der HoloLens sind die Techniker in der Lage, mögliche Probleme frühzeitig zu erkennen, zu visualisieren und freihändig Informationen aus der Zentrale über die für die Wartung notwendigen Schritte zu erhalten.

www.thyssenkrupp.com/de

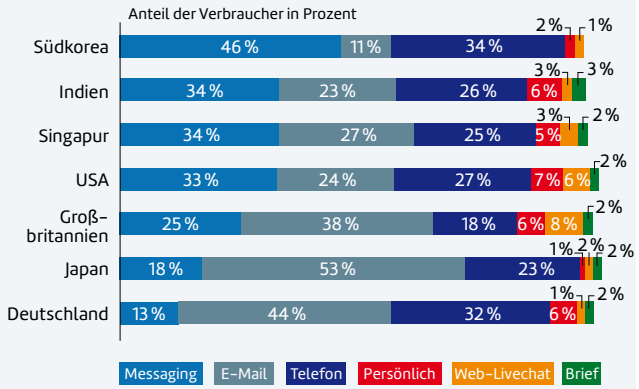
Die Wartungslösung MAX mit HoloLens

Studie von Twilio

Kurznachrichten bevorzugt

Der Cloud-Anbieter Twilio Inc. hat die Ergebnisse seines »Global Mobile Messaging Consumer Reports« veröffentlicht. Die Ergebnisse zeigen, dass 9 von 10 Verbrauchern Kurznachrichten verwenden wollen, um mit Marken zu kommunizieren. Allerdings verfügen weniger als die Hälfte der global operierenden Unternehmen über die dafür nötige Infrastruktur. Twilio befragte mehr als 6000 Verbraucher in Nordamerika, Europa und Asien.

Mobile Messaging ist hierbei weltweit für Verbraucher die beliebteste Kommunikationsform. Der durchschnittliche Mobile-Nutzer hat drei Messaging-Apps auf dem Homescreen seines Smartphones, er nutzt drei verschiedene Messaging-Apps pro Woche und sendet durchschnittlich drei Nachrichten pro Stunde.



Messaging steht bei Kunden hoch im Kurs

web & mobile developer 12/2016

Die Akzeptanz von Push-Benachrichtigungen bei Messaging-Anwendungen liegt bei 77 Prozent – Nachrichten über diesen Kanal werden also mit hoher Wahrscheinlichkeit gelesen. 66 Prozent der Verbraucher geben Messaging gegenüber allen anderen Kommunikationsformen den Vorzug, um mit Marken in Kontakt zu treten oder von ihnen erreicht zu werden.

In Deutschland ist für 54 Prozent der Verbraucher Messaging die beliebteste Form der Kommunikation mit Unternehmen. Je jünger Personen sind, umso wahrscheinlicher ist es, dass sie Messaging gegenüber anderen Kanälen für den Kundenservice bevorzugen.

»Die heute vorgestellte Studie zeigt eine wachsende Kluft zwischen Konsumenten und Marken auf«, sagt Manav Khurana, Vice President Produktmarketing bei Twilio. »Verbraucher senden mehr Kurznachrichten als sie telefonieren, E-Mails schreiben oder auf Social Media posten, aber Unternehmen versuchen immer noch, sie über Kanäle zu erreichen, die sie nicht mehr nutzen. Um mit den Verbrauchern mitzuhalten, müssen Marken das Messaging als Kanal für sich entdecken und anfangen, mit ihren Kunden auf die gleiche Art und Weise zu kommunizieren, wie diese es untereinander tun.«

www.twilio.com

Smartphones

Android beliebtestes Betriebssystem

Das mit Abstand am weitesten verbreitete Betriebssystem für Smartphones in Deutschland ist Android.

Fast drei Viertel (72 Prozent) der privaten Smartphone-Nutzer verwenden derzeit Android. 14 Prozent nutzen das ausschließlich für Apple-Geräte verfügbare Betriebssystem iOS. Das hat eine aktuelle Umfrage unter 932 Smartphone-Nutzern

Samsung

Fast drei Viertel der Smartphone-Nutzer verwenden in Deutschland Android

im Auftrag des Digitalverbands Bitkom ergeben. Das Betriebssystem Microsoft Windows Phone kommt demnach auf 5 Prozent und Blackberry auf 1 Prozent. Andere Smartphone-Plattformen konnten sich in den vergangenen Jahren nicht in der Breite durchsetzen. 8 Prozent der befragten Smartphone-Nutzer können keine Angaben zu ihrem Betriebssystem machen. „Das Smartphone-Betriebssystem ist das technische Herzstück der Geräte“, sagt Dr. Frank Termer, Bereichsleiter Software beim Bitkom. Das Betriebssystem stelle die Grundfunktionen der Geräte bereit wie die Bedienung per Touchscreen, die Steuerung der Kamera oder zentrale Sicherheitsfunktionen. Zusätzliche Anwendungen können die Nutzer aus den für das jeweilige Betriebssystem verfügbaren App-Stores herunterladen.

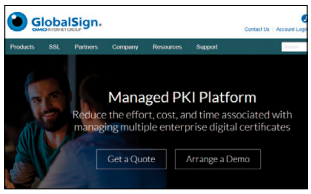
Der Name Android geht auf ein kleines Software-Unternehmen zurück, das ein neues Betriebssystem für Handys entwickeln wollte. Für die Verbreitung des Systems gründete sich die „Open Handset Alliance“, an der neben Google Geräteherstellers wie Samsung, HTC oder Motorola, Netzbetreiber wie T-Mobile oder Telefónica, Chip-Hersteller wie Intel sowie andere Technologie-Unternehmen beteiligt waren. Im Jahr 2008 kam dann das erste Android-Gerät von HTC auf den Markt – ein Jahr nach der ersten iPhone-Generation. Seitdem entwickelt Google Android schrittweise weiter und brachte zuletzt einmal pro Jahr eine neue Version auf den Markt. Aktuell wird Android 7.0 ausgerollt, das auch unter dem Namen Nougat bekannt ist. Als quelloffenes Betriebssystem auf Linux-Basis können die Gerätehersteller und andere Software-Unternehmen Android nach ihren Wünschen anpassen. Allerdings macht Google den Produzenten bestimmte Vorgaben, wenn es um die Nutzung seiner Apps geht. Auch Apple wird ist im Begriff, eine neue Version seines Betriebssystems unter der Versionsnummer iOS 10 zu verteilen. Microsoft hat mit Windows 10 bereits eine neue Plattform auf den Markt gebracht, die sowohl auf Smartphones und Tablets als auch auf Desktop-Computern läuft.

www.bitkom.org

GlobalSign

Mehr Sicherheit bei Enterprise Mobility

GMO GlobalSign, ein führender Anbieter von Identitäts- und Sicherheitslösungen für das Internet of Everything (IoE), hat mit MobileIron eine Partnerschaft geschlossen. Ziel ist es, Unternehmen ein strafferes Verwaltungssystem für die Ausstellung von digitalen Zertifikaten zur mobilen Authentifizie-



Verwaltungssystem für die Ausstellung von digitalen Zertifikaten

rung zu bieten. Die gemeinsame Lösung stellt digitale Identitäten automatisch von der MobileIron Cloud Enterprise Mobility Management (EMM) Plattform für mobile Geräte bereit, ohne dass der Endnutzer eingreifen müsste.

Die Consumerization der IT hat Mitarbeitern mehr Flexibilität gebracht. Das gilt auch, wenn die von ihnen bevorzugten mobilen Geräte für den Zugriff auf Unternehmensnetzwerke und -ressourcen benutzt werden. Darin liegt allerdings eine Herausforderung für die IT. Sie muss gewährleisten, dass Mitarbeiter auf alles, was sie brauchen, zugreifen können, ohne dass Schwachstellen oder der Verlust von unternehmens- oder personenbezogenen Daten die Folge sind.

Unterstellt man, dass 87 Prozent der verkauften vernetzten Geräte im nächsten Jahr Tablets beziehungsweise Smartphones sein werden, entsteht ein immenser Bedarf, einen unberechtigten Zugriff auf Unternehmensressourcen zu verhindern. Digitale Zertifikate auf mobilen Geräten ermöglichen nachprüfbare Geräteidentitäten, sodass Gerätenutzern in Unternehmensnetzwerken und -ressourcen (WLAN oder VPN) vertraut werden kann.

»Der Umgang mit BYOD im Unternehmen ist komplex. Aber die richtige EMM-Plattform kann Unternehmen helfen, die Produktivität zu verbessern, ohne dabei die Sicherheit zu opfern«, so John Morgan, Vice President of Product und Ecosystem bei MobileIron. »MobileIrons Zusammenarbeit mit GlobalSign erleichtert es IT-Administratoren, Mitarbeiter in die Lage zu versetzen, schneller und intelligenter zu arbeiten und gleichzeitig Unternehmensdaten und Privatsphäre der Mitarbeiter auf mobilen Geräten zu schützen.«

Als Zertifizierungsstelle (CA) klinkt sich GlobalSigns cloudbasierte Zertifikat-Management-Plattform, die auf Public Key Infrastructure (PKI) basiert, direkt in das MobileIron Cloud-Admin-Portal ein und erleichtert es Unternehmen, Authentifizierung für Mobilgeräte bereitzustellen. Mit MobileIron und GlobalSign werden Zertifikats-Auftrag und -Verwaltung vollständig automatisiert. Sobald ein neues Gerät bei der MobileIron Cloud registriert ist, kann ein digitales Zertifikat von GlobalSign für dieses Gerät ausgestellt werden. Die Integration entlastet IT-Mitarbeiter davon, Zertifikate auf jedem einzelnen Gerät manuell installieren und verwalten zu müssen. Dies reduziert den Administrationsaufwand und senkt die Gesamtbetriebskosten.

www.globalsign.com/de-de

Bitkom

Muster-AGB für Software as a Service

Der Bitkom hat seine Empfehlungen für Allgemeine Geschäftsbedingungen (AGB) um eine Vorlage zu Software as a Service (SaaS) erweitert. Das neue Modul regelt typische Fragen für die browserbasierte Bereitstellung von Software über die Cloud. In Deutschland nutzt laut einer aktuellen Bitkom-Studie mehr als die Hälfte der Unternehmen Cloud-Dienste (54 Prozent). Auf SaaS setzen die Unternehmen insbesondere bei Office-Anwen- ▶

DDoS-Attacken

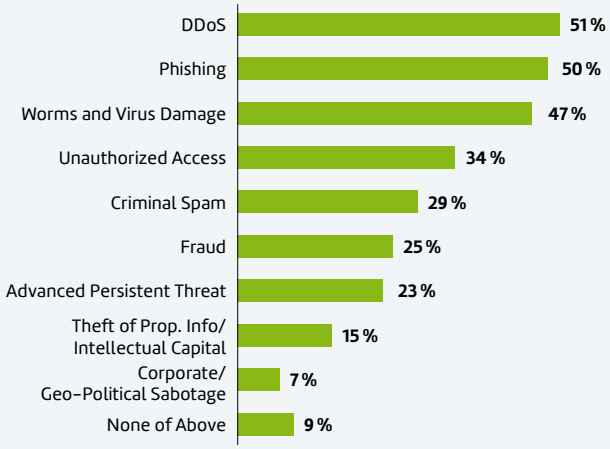
Der Hacker in der Cloud

Cloud-Plattformen werden immer häufiger missbraucht, um DDoS-Attacken mit hohen Datenvolumen zu realisieren, berichtet der DDoS-Spezialist Radware. Auch für Phishing-Versuche haben Hacker die Cloud entdeckt.

Die Gründe für diesen Trend hin zur Cloud sind nach Angaben des Emergency Response Teams von Radware vor allem die Rechenleistung und die schnelle Anbindung von Cloud-Plattformen, die sehr hohe Datenraten ermöglicht, sowie der zusätzliche Level an Anonymität durch Nutzung einer fremden Infrastruktur statt einer eigenen. Dabei nutzen Hacker in der Regel kostenlose Cloud-Angebote, für deren Dienste man sich meist mit Hilfe einer anonymisierten E-Mail-Adresse registrieren kann. Innerhalb eines solchen Cloud-Angebots kann ein Hacker in kürzester Zeit attackierende Skripts auch auf mehrere Instanzen laden, um massive DDoS-Attacken zu starten, ohne auf kostenpflichtige Dienste wie Network Stresser angewiesen zu sein. Während die meisten Cloud Services kein Spoofing erlauben und den Angreifer auf diese Weise limitieren, gibt es doch manche, bei denen diese Möglichkeit besteht. In diesem Fall kann der Hacker in der Cloud auch Reflection- und Amplification-Floods initiieren. Auch Attacken mit dynamischen IP-Adressen zur Generierung eines 3-Wege-Handshakes haben zunehmend ihren Ursprung in der Cloud. Solche Attacken ähneln legitimem Verkehr sehr stark, sind daher nur sehr schwer zu erkennen und können mit herkömmlichen Abwehrmaßnahmen auf Basis von IP-Adressen nicht bekämpft werden.

»Die Cloud bietet Unternehmen sehr flexible und leistungsfähige Dienste an«, kommentiert Georgeta Toth, Regional Director DACH bei Radware, »aber leider nicht nur diesen. Vor allem stellen sie für Hacker eine ideale Möglichkeit dar, weit größere Datenvolumen zu generieren als mit einer eigenen Infrastruktur. Bei der Planung von Abwehrmaßnahmen sollte daher jedes Unternehmen auch massive, cloudbasierte Attacken auf dem Radarschirm haben.«

www.radware.com



Auch für Phishing-Versuche haben Hacker die Cloud entdeckt

web & mobile developer 12/2016

Wertvollste Markenunternehmen

Apple, Google und Microsoft an der Spitze

Das European Brand Institute hat, basierend auf aktuellsten ISO-Standards, mehr als 3000 Markenunternehmen und deren Marken in 16 Branchen analysiert. Apple bleibt weiterhin die wertvollste Marke der Welt mit 148,531 Milliarden Euro (+8 Prozent), gefolgt von Google mit einem Markenwert von 91,850 Milliarden Euro (+25 Prozent) und Microsoft mit 75,572 Milliarden Euro (+13 Prozent). Amazon steigerte seinen Markenwert um 54 Prozent und rückt auf Platz 7 vor. Europäische IT-Unternehmen können an der Weltspitze nicht mithalten. In Europa steht die französische Aktiengesellschaft LVMH (Inhaber diverser Luxusmarken wie Louis Vuitton und Moët Hennessy) mit einem Markenwert von 43,510 Milliarden Euro (global Platz 14) an der Spitze, gefolgt von der belgischen AB Inbev Group (Brauereigruppe Anheuser-Busch) mit 39,468 Milliarden Euro (+10 Prozent) und Nestlé mit 37,957 Milliarden Euro (+11 Prozent). Deutschlands Nummer 1 ist die Volkswagen Group mit 26,662 Milliarden Euro, sie rangiert global auf Platz 28.

Dr. Gerhard Hrebicek, Vorstand des European Brand Institute, resümiert: Das Durchschnittswachstum der Top 100 beträgt 13 Prozent, die Top 10 USA steigen um 17 Prozent, die Top 10 China um 22 Prozent und die Top 10 Europa um 13 Prozent. Alibaba weist mit 77 Prozent das höchste Markenwertwachstum auf, gefolgt von Facebook (57 Prozent). Europas Wachstumssieger sind Robert Bosch und BT Group, die ihre Markenwerte um 28 Prozent steigern konnten.

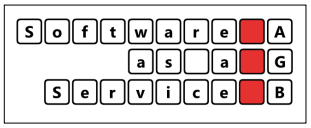
www.eurobrand.cc

Ranking 2016 (2015)	Markenunternehmen	Branche	Land des Firmensitzes	Markenwert in Mio. €	Jahres-Zuwachs
1 (1)	Apple	IT & Technologie	USA	148.531	8 %
2 (2)	Google	IT & Technologie	USA	91.850	25 %
3 (3)	Microsoft	IT & Technologie	USA	75.572	13 %
4 (4)	Coca-Cola	Konsumgüter	USA	73.426	10 %
5 (9)	AT&T	Telco	USA	59.486	26 %

Apple bleibt weiterhin die wertvollste Marke der Welt

dungen (43 Prozent), Groupware und branchenspezifischen Anwendungen (jeweils rund 35 Prozent). »Mit der zunehmenden Bedeutung von SaaS-Lösungen in den Unternehmen ist auch die Nachfrage nach standardisierten AGB bei den Anbietern stark gestiegen«, so Anja Olsok, Geschäftsführerin der Bitkom Servicegesellschaft.

Mit den Muster-AGB wendet sich Bitkom besonders an Start-ups und kleine und mittlere Unternehmen (KMU). Die Unternehmen können die an die aktuelle Rechtsentwicklung angepassten Muster-AGB entweder komplett übernehmen



Das AGB-Angebot des Bitkom ist modular aufgebaut

oder individuell anpassen. »Gerade junge und kleine Unternehmen haben oft keine eigene Rechtsabteilung. Mit den Muster-AGB geben wir den Dienstanbietern eine rechtssichere Basis an die Hand und nehmen ihnen juristische Feinarbeit ab«, sagt Olsok.

Das AGB-Angebot des Bitkom ist modular aufgebaut. Grundlage sind die Allgemeinen Vertragsbedingungen, die übergreifende Fragen beispielsweise zu Haftung und Leistungsstörungen für alle Vertragstypen in identischer Weise regeln. Ergänzt werden die allgemeinen Vertragsbedingungen durch neun Leistungsmodule, die AGB für spezifische Geschäftsmodelle und Vertragstypen enthalten. Begleitende Hinweise führen in die Verwendung der AGB-Module ein und geben erste Erläuterungen zum Hintergrund einzelner Klauseln. Im Angebot zusätzlich enthalten sind Musterformulare, die die Nutzer bei der Vertragsabwicklung unterstützen. Die Muster-AGB stehen gegen eine Schutzgebühr zum Download bereit.

www.bitkom-consult.de/muster-agb

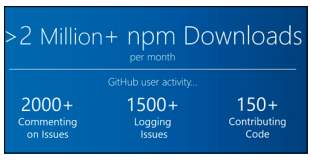
Microsoft

TypeScript 2.0 ist da

Microsoft hat die finale Version von TypeScript 2.0 freigegeben.

TypeScript ist eine typsichere Open-Source-Programmiersprache, deren Output pures JavaScript liefert. Zusätzlich zu den Neuerungen, die bereits Bestandteil der Beta-Version von TypeScript 2.0 waren, wartet die fertige Version noch mit einigen zusätzlichen Erweiterungen auf. So bringen beispielsweise die Tagged Unions Funktionalitäten von Sprachen wie F#, Swift oder Rust für JavaScript. Hinzugekommen sind auch neue Literal-Typen. In Version 2 haben neben Strings auch Boolean- oder Number-Variablen sowie Enum-Members ihren eigenen Typ. Inzwischen meldet TypeScript mehr als 2 Millionen Downloads pro Monat. TypeScript 2.0 steht für die Nutzung mit Visual Studio 2015 (Update 3 ist erforderlich) oder für Visual Studio Code (VS Code) zur Verfügung. Die Entwicklungsumgebung VS Code läuft nativ sowohl auf Windows als auch auf Linux und Mac OS. Für Nutzer der kommenden Version von Visual Studio (Codename Visual Studio 15) soll TypeScript 2.0 in den nächsten Preview-Release aufgenommen werden.

www.typescript.org ▶



Die finale Version von TypeScript 2.0 steht ab sofort zur Verfügung

DDC .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	Konferenz	Workshops
05.12.2016	**06.12.2016**	**07.12.2016**
8 halbtägige Workshops	1 Keynote, 12 Vorträge	Ganztägiges Praxistraining

Ihre Experten (u.a.):

Bernd Marquardt
Freier Consultant und Autor

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**

Partner: proximity TECHNOLOGY, Saxonia Systems So geht Software.

Präsentiert von: dotnetpro

Veranstalter: developer media, Neue Mediengesellschaft Ulm mbH

E-Mails

Fluch und Segen zugleich

E-Mails können im Büroalltag eine echte Plage sein. Fast jeder ärgert sich dann und wann über sie. Doch seien wir ehrlich: Das Problem sind weniger die E-Mails an sich, sondern der falsche Umgang mit ihnen. Der erste und häufigste Fehler ist, eine E-Mail zu schreiben, wenn man keine schreiben sollte: Jeder vierte deutsche Büroangestellte ärgert sich regelmäßig über E-Mails, weil ein persönliches Gespräch deutlich besser gewesen wäre. 17 Prozent sind ebenfalls genervt von Kollegen, die ständig »allen antworten«, obwohl der E-Mail-Inhalt gar nicht für alle bestimmt beziehungsweise relevant ist. Auch den Vorgesetzten bei allem und jedem in Kopie zu setzen (15 Prozent), weitergeleitete Mails, die man längst erhalten hat (13 Prozent), und Kritik oder negatives Feedback via E-Mail (8 Prozent) kommen bei den Deutschen nicht allzu gut an. Dies ist das Ergebnis einer aktuellen Adobe-Studie.

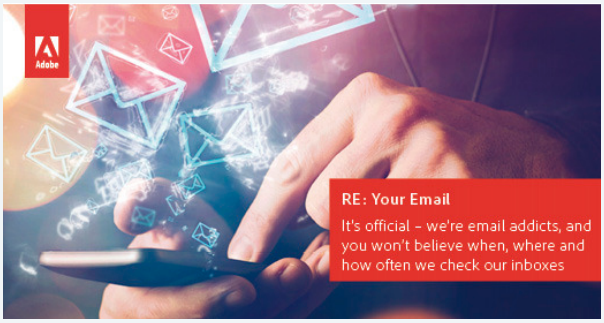


Foto: Adobe

E-Mail spielt im privaten und beruflichen Umfeld eine zentrale Rolle

Einer der größten Nerv-Faktoren ist nicht zuletzt die zunehmende E-Mail-Flut: Wenn das Postfach mal wieder überzuquellen droht, wird die E-Mail schnell zum echten Zeitfresser. Um dem entgegenzuwirken, haben sich 81 Prozent der Deutschen eine Strategie zurechtgelegt. 38 Prozent reagieren sofort auf eingehende E-Mails. Nicht gelesene Newsletter werden von 35 Prozent regelmäßig abbestellt. Bei mehr als jedem Fünften hat es sich bewährt, nur die letzte E-Mail zu einem Thema im Posteingang zu belassen – alle anderen werden gelöscht oder in entsprechende Ordner sortiert.

Trotz aller Widrigkeiten spielt die E-Mail im privaten und beruflichen Umfeld nach wie vor eine zentrale Rolle. Für jeden fünften deutschen Büroangestellten ist sie auch heute noch das bevorzugte Kommunikationsmittel, um sich mit Kollegen auszutauschen. Mehr als vier Stunden verbringen sie täglich damit, ihre privaten und beruflichen E-Mails zu checken. Für das Marketing bleibt die E-Mail damit auch in Zeiten von Snapchat und boomender WhatsApp-Kommunikation ein relevanter Kanal, um potenzielle Interessenten zu erreichen. Und mehr noch: Für 55 Prozent der Deutschen ist sie sogar der bevorzugte Weg, um von einer Marke kontaktiert zu werden. Und das am liebsten mobil, denn: Mehr als zwei Drittel (69 Prozent) lesen ihre E-Mails regelmäßig auf dem Smartphone.

www.adobe.com

Microsoft

Windows Server 2016 ist da

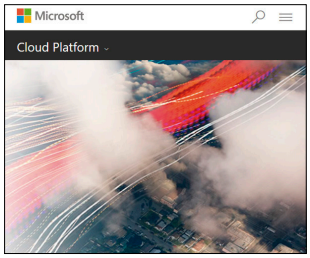
Windows Server 2016 und System Center 2016 sind ab Oktober 2016 generell verfügbar. Das cloudfähige Serverbetriebssystem Windows Server 2016 kommt mit Docker-Tools und Nano Server.

Windows Server 2016 ist Microsofts cloudfähiges Serverbetriebssystem für Geschäftskunden. System Center 2016 ermöglicht eine vereinfachte Rechenzentrumsverwaltung für komplexe, heterogene Arbeitslasten (Workloads). Windows Server 2016 bringt Unterstützung für Windows- und Linux-Container mit. Die Container lassen sich über die Open-Source-Engine Docker verwalten und stellen Applikationen automatisch bereit. Kunden können Docker-Container-Anwendungen erstellen, die auf Windows-Servern im eigenen Rechenzentrum genauso lauffähig sind wie unter Windows Server 2016 in einer virtuellen Maschine auf Microsoft Azure, unabhängig von der Hosting-Umgebung und dem Cloud-Provider.

Windows Server 2016 unterstützt auch Hyper-V-Container. Hier handelt es sich um mit Docker verwaltbare Container, die Microsofts Hypervisor Hyper-V für die Virtualisierung nutzen. Dabei virtualisiert Hyper-V den Container und nicht das gesamte Betriebssystem. Diese Virtualisierung schafft eine zusätzliche Isolationsschicht und ist für den Einsatz von Containern in mandantenfähigen Umgebungen nützlich.

Beim Nano Server handelt es sich um eine Minimalversion von Windows Server, die nur etwa ein Zwanzigstel der Größe des Windows Server Core hat. Sie bringt nur absolut notwendige Komponenten mit: Hyper-V, Clustering, Networking, Storage, .NET und Common Language Runtime – aber keine grafische Oberfläche. Durch den neuen Nano Server werden Angriffspunkte auf firmeneigene IT-Infrastrukturen deutlich reduziert. Nano Server minimiert häufige Neustarts (Reboots) von Rechnern wegen des Einspielens von Updates und Patches und verringert damit auch das Risiko eines möglichen Eindringens von Schadsoftware von außen.

»Vom Nano Server versprechen wir uns eine massive Reduktion des Aufwands für das Patchen des Betriebssystems und damit wesentlich kürzere Downtimes unserer Maschinen«, sagt Hansjörg Sonnleitner, Vice President Operational Services IT Systems bei der Infineon Technologies AG. »Sobald Windows Server 2016 in unserer Umgebung qualifiziert ist, werden wir das System durchgängig als Standard festlegen. Wir planen den Abschluss der Qualifikation für Mitte 2017.«



Windows Server 2016 und System Center 2016 sind ab Oktober 2016 generell verfügbar

www.microsoft.com

Ignite-Konferenz

Ignite-News im Schnelldurchlauf

Auf der Microsoft-Konferenz Ignite haben Adobe und Microsoft eine strategische Partnerschaft bekannt gegeben. Microsoft präsentierte auf der Konferenz zahlreiche Neuhei-

Auf der Microsoft-Konferenz Ignite wurden zahlreiche Neuigkeiten bekannt gemacht

ten, darunter Windows Server 2016 und System Center 2016, die beide ab Oktober verfügbar sind. Außerdem stellte Microsoft auf der Ignite neue Funktionen und Services rund um Windows 10, Office 365, Azure und Enterprise Mobility + Security (EMS) vor. Die wichtigsten News im Überblick:

Im Rahmen der neuen Partnerschaft mit Microsoft gab Adobe bekannt, künftig seine Marketing-, Creative- und Document-Cloud auf der Basis von Microsoft Azure anzubieten. Microsoft empfiehlt zudem künftig die Adobe Marketing Cloud als seine bevorzugte digitale Marketing-Lösung für Dynamics 365 Enterprise.

Windows Defender Application Guard soll aus Microsoft Edge in Windows 10 den sichersten Browser für Unternehmen machen. Der Guard nutzt isolierte, direkt in die Hardware integrierte Container, um die unkontrollierte Ausbreitung von Schadcode über Links und E-Mails zu unterbinden.

Windows Defender Advanced Threat Protection (WDATP) und Office 365 ATP teilen sich künftig ihre Analysefunktionen, um Administratoren schnellere Reaktionen auf Sicherheitsvorfälle rund um Windows 10 und Office 365 zu ermöglichen. Die Sicherheitsfunktionen sammeln und verarbeiten Auffälligkeiten, die auf unentdeckte Schadsoftware hindeuten. WDATP ist Teil von Windows 10 Enterprise E5, Office ATP ist ab sofort Teil der E5-Lizenz und steht für andere Office-Lizenzpläne als Add-on zur Verfügung.

Beide Lösungen sind zudem Teil des neuen Bundles Secure Productive Enterprise E5, mit dem Microsoft vom 1. Oktober an die Sicherheits- und Produktivitätsfunktionen von Office 365, Windows 10 Enterprise und der neuen Enterprise Mobility + Security bündelt.

https://ignite.microsoft.com

Bluetooth SIG

Neue Developer-Toolkits für das IoT

Die Bluetooth SIG bietet neue Toolkits für höhere IoT-Sicherheit, Interoperabilität und die wachsenden Herausforderungen im IoT-Markt.

Die Bluetooth Special Interest Group (SIG) hat mehrere neue Werkzeuge und Updates ihrer Wireless-Entwicklungssuite veröffentlicht. Die Updates des Bluetooth-Developer-Toolkits umfassen ein Bluetooth Secure Gateway Toolkit, das Bluetooth Starter Kit, den Application Accelerator 2.1 sowie ein Beacon Smart Starter Kit. Mit den Updates und neuen Tools sind Entwickler schneller und smarter in der Lage, mobile Apps, kostengünstige Beacons sowie Internet Gateways herzustellen, die Sensoren im Internet der Dinge (Internet of Things – IoT) abgesichert steuern.

Das Bluetooth Secure Gateway Toolkit stellt Beispiel- ▶

Die Bluetooth Special Interest Group (SIG) hat neue Developer-Toolkits für das IoT veröffentlicht

Mastercard

Bezahlen per Fingerabdruck und Selfie

Ab heute ist Identity Check Mobile in Europa verfügbar. Diese neue Zahlungstechnologie nutzt für die Verifizierung der Identität der Karteninhaber biometrische Verfahren – einzigartige Merkmale wie Fingerabdruck- oder Gesichtserkennung – und vereinfacht damit das Online-Shopping. Nach erfolgreich absolvierten Tests in den Niederlanden, den USA und Kanada startet Mastercard diese Technologie nun in folgenden zwölf Ländern: Deutschland, Österreich, Belgien, Niederlande, Großbritannien, Spanien, Tschechien, Ungarn, Dänemark, Norwegen, Schweden und Finnland. Bei den derzeit verfügbaren Lösungen zum Nachweis der Identität und damit der Autorisierung der Zahlung passiert es nicht selten, dass Benutzer die Website eines Händlers beziehungsweise die jeweilige mobile App wieder schließen, weil sie aufgefordert werden, ein Passwort einzugeben. Sich an das jeweilige Passwort zu erinnern, gegebenenfalls ein neues Passwort zu generieren oder dieses einzugeben kann aufwendig und zeitraubend sein. Bei falscher Eingabe wird die Transaktion nicht ausgeführt und oft wird der Kauf dann frustriert abgebrochen. Mit Mastercard Identity Check Mobile ist

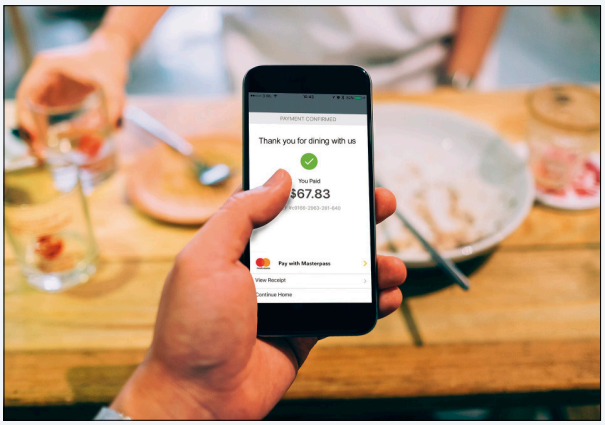


Foto: Mastercard

Mastercard führt eine neue App für biometrische Bezahlverfahren ohne Passworteingabe beim Online-Shopping ein

keine Passworteingabe mehr erforderlich. Stattdessen profitieren Karteninhaber von schnelleren Einkaufserlebnissen und mehr Sicherheit, da sie zum Identitätsnachweis lediglich ihren Finger auf den Scanner ihres Smartphones legen müssen oder die Gesichtserkennung per Selfie nutzen können.

»Unsere Ziel ist es, Karteninhabern und Händlern möglichst reibungslose Online-Zahlungserlebnisse zu ermöglichen, ohne dabei irgendwelche Kompromisse bei der Sicherheit einzugehen«, so Ajay Bhalla, President von Enterprise Risk & Security bei Mastercard. »Dieses neue Zahlungsverfahren stellt einen bedeutenden Meilenstein dar. Das Einkaufserlebnis in Ladengeschäften hat sich schon durch unsere Innovationen, wie kontaktlosfähigen Karten und die Bezahlung über Mobilgeräte oder Wearables, grundlegend verändert. Als nächster Meilenstein ist Identity Check Mobile für Online-Shopping in Europa und bald auch weltweit verfügbar.«

www.mastercard.com

Lange Nacht der Start-ups

Productive Mobile gewinnt Detecon Valley Pitch

Das Start-up Productive Mobile hat den Detecon Valley Pitch im Rahmen der langen Nacht der Start-ups in Berlin gewonnen. Das junge Unternehmen ermöglicht es IT-Abteilungen, webbasierte Intranetanwendungen schnell und einfach in mobile Apps zu transformieren. Kunden profitieren somit von höherer Produktivität und Prozessgeschwindigkeit, ohne hierfür komplexe und aufwendige Optimierungen für die Mobilgeräte vornehmen zu müssen. Anthony Hsiao, Mitgründer und CEO von Productive Mobile, freut sich: »Unser Sieg zeigt, dass wir mit

Productive Mobile hat den Detecon Valley Pitch im Rahmen der langen Nacht der Start-ups in Berlin gewonnen

unserer Strategie, Mitarbeiter zufriedener und mobiler machen zu wollen, auf dem richtigen Weg sind. Mit unserer Plattform für Rapid Mobile App Development lassen sich mobile Enterprise-Apps in wenigen Tagen oder Stunden entwickeln«.

Das Gewinnerteam freut sich nun über eine Reise ins Silicon Valley und einen Besuch des Detecon Innovation Institute in San Francisco. Die Jury rund um Managing Partner Lars Theobaldt von Detecon sowie Adriaan Ligtenberg (CEO AllMobile Fund), Florian Steger (Senior Manager hub:raum) und Tilo Bonow (CEO Piabo PR) war begeistert: »Die professionelle Präsentation, aber vor allem das Produkt selbst, das flexibles, unabhängiges Arbeiten ohne Medienbrüche erlaubt, gaben den Ausschlag«, so Theobaldt. Insgesamt hatten sich neben Productive Mobile die fünf weiteren Finalisten Teraki, SatoshiPay, Twyla, Cookies App und Illusion Walk KG für den Detecon Valley Pitch qualifiziert.

Dr. Heinrich Arnold, Chief Executive Detecon Digital, der als Leiter der Telekom Laboratories (T-Labs) das Start-up-Event selbst mitinitiierte, betont: »Die Faszination der langen Nacht rührt daher, dass sich Start-ups mit ihren Pitches direkt vor ihren potenziellen Kunden beweisen müssen. Unabhängig von Hypes oder technologischen Netzwerken erfahren sie im direkten Austausch, ob ihre Innovationen tatsächlich gewollt sind.« Interessant sei für die Partner und Förderer zudem immer wieder, ob und wie sie Impulse und Ideen der Start-ups in eigenen agilen Einheiten übernehmen wollen.

www.startupnight.de

Komponenten bereit und unterstützt eine einfache Ersteinrichtung sicherer Internet-Gateways. Es bietet Labs für ein tief reichendes Technologieverständnis sowie praktische Übungen für die Entwicklung dieser Gateways. Mit ihnen lassen sich Bluetooth-fähige Sensoren aus der Ferne überwachen und steuern.

Das Toolkit erlaubt es, die IoT-Funktionalitäten Bluetooth-basierter Produkte zu erweitern. Bluetooth 4.2 bietet dazu industrieführende Sicherheitsfunktionen, die hohen staatlichen Standards entsprechen.

Bluetooth-Geräte können beispielsweise künftig über ein Internet-Gateway direkt eine sichere Verbindung mit Sensoren herstellen, ohne dass ein Smartphone oder Tablet dazwischengeschaltet werden muss.

Das umfassende Trainingspaket Bluetooth Starter Kit umfasst praktische Labs sowie Beispiel-Quellcodes zu Bluetooth-Anwendungen und vermittelt Entwicklern grundlegende Kenntnisse zu drahtlosen Verbindungen mit Bluetooth-fähigen Produkten. Die neueste Version bietet eine neue, kostengünstigere Arduino-/Genuino-Hardware. Arduino ist eine Open-Source-Plattform, die einfach zu nutzende Hardwarekomponenten für Entwickler vermarktet.

Die Trainings-Labs verwenden jetzt das Bluetooth Developer Studio (BDS), was Herstellern die Entwicklung ihrer ersten Bluetooth-fähigen Geräte erleichtert.

Auch das BDS hat Erweiterungen erfahren. Sie erlauben es Developern, konsistente Bluetooth-Implementierungen über verschiedene Chipsätze sicherzustellen. Die Produktentwicklung wird schneller und Bluetooth-Dienste funktionieren über verschiedenste Hardwareplattformen hinweg.

www.bluetooth.com

Mastercard Developers

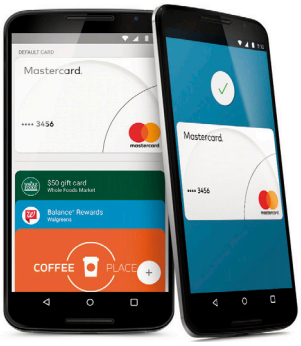
Plattform für E-Commerce-Lösungen

Mit Mastercard Developers unterstützt Mastercard Entwickler innovativer E-Commerce-Lösungen. Die Plattform hilft, neue Ideen aus den Bereichen Augmented Reality, Virtual Reality und dem Internet of Things zu entwickeln und zu testen.

Über das zentrale Gateway Mastercard Developers erhalten die Partner von Mastercard Zugang zu verschiedenen APIs für Zahlungs-, Daten- und Sicherheitslösungen. Darüber hinaus umfasst die Plattform eine API-Kategorie mit der Bezeichnung »New and Experimental«, mit der die Unternehmen innovative Technologien und Anwendungen testen können. Die Test-APIs unterstützen die Partner beim Entwickeln, Skalieren und Integrieren von Zahlungsfunktionen in neue Plattformen. Zugleich können Ideen aus ganz neuen Bereichen, wie etwa Augmented Reality, Virtual Reality und dem Internet of Things (IoT), getestet werden.

Auf der Plattform haben die Partner jetzt Zugriff auf über 25 Mastercard APIs – einschließlich der Walletlösung Masterpass und Mastercard Digital Enablement Services.

http://developer.mastercard.com



Entwickler erhalten von Mastercard Zugang zu verschiedenen APIs für Zahlungs-, Daten- und Sicherheitslösungen

Jetzt kostenlos testen!

Das Fachmagazin für .NET-Entwickler

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

probeabo.dotnetpro.de/kostenlos-testen/

NEO4J – GRAPHDATENBANK AUF JAVA-BASIS

Durchgängig Java auch für Graphen

Neo4j bietet dem Entwickler eine Vielzahl von Java-APIs und damit umfassende Wahlfreiheit bei der Lösung einer konkreten Problemstellung.

Als Datenbank-Management-System (DBMS) für Graphen besitzt Neo4j Schnittstellen für alle in der Praxis relevanten Programmiersprachen. Bei der Implementierung des DBMS entschied sich der Hersteller Neo Technology jedoch von Anfang an für Java. Damit rückte Java, wie bereits aus dem Produktnamen der Datenbank durch die Endung 4j (gesprochen: for j) ersichtlich, als prädestinierte Technologie ins Zentrum des DBMS. Alsbald unterstützte Neo Technology den Architekturstil REST (Representational State Transfer) und für Java EE das Spring-Data-Projekt.

Das REST-Interface bietet Zugriffe per JSON-Datenschnittstelle auf einen Neoj4-Server zur Realisierung verteilter Systeme über Webservices. REST kann von Java, aber natürlich auch von jeder anderen Programmiersprache genutzt werden. Mit Spring Data Neo4j (SDN) stand schon recht frühzeitig eine Template-Schnittstelle für das im Umfeld der Java Platform Enterprise Edition (Java EE) populäre Spring-Framework zur Verfügung. Alle genannten Aspekte trugen maßgeblich zu einer schnelleren Verbreitung von Neo4j in der Anwendungsentwicklung bei.

Bis zur Version 2 erfolgte die Programmierung der Graphdatenbank auf der Basis eines spezifischen Java-APIs, auch Native Java API genannt. In der Fachwelt setzte sich für dieses Native Java API von Anfang an die Bezeichnung Neo4j OGM (Object Graph Mapping Library for Neo4j) durch. Gesammelte Projekterfahrungen führten zu fortlaufenden Verbesserungen und Modularisierungen der OGM Library, daher sollte man das alte Native Java API heutzutage besser als Core Java API bezeichnen. Beginnend mit Version 1.4 der Neo4j-Datenbank stellte Neo Technology erstmals CQL (Cy-

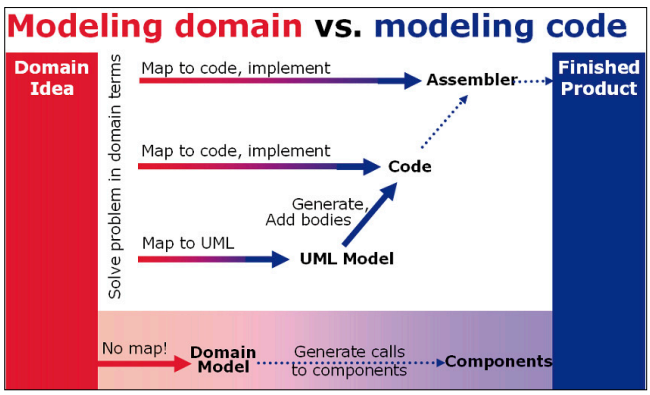
pher Query Language) als experimentelles Feature der Öffentlichkeit vor. CQL von Neo4j entspricht der Abfragesprache SQL in der relationalen Welt. Für CQL realisierte der Hersteller ein Java-API, um die Elemente der Graphdatenbank über Cypher in Java-Anwendungen zu verarbeiten. Im vergangenen Jahr startete das Unternehmen mit einer Reihe von Partnerfirmen das Projekt openCypher, um Cypher als künftigen Standard für Abfragesprachen von Graphdatenbanken weiterzuentwickeln.

Mittels Cypher als Neo4j-spezifische Abfragesprache gewährleistet der Hersteller die seitens eines jeden DBMS postulierte Datenunabhängigkeit (Data Independence): Ein Benutzer oder Anwendungsprogramm benötigt keinerlei Kenntnisse über die technische Realisierung der Speicherung oder des Zugriffs auf die Daten. Datenunabhängigkeit erreicht jedes DBMS durch die Realisierung abstrakter Datentypen, die wohldefinierte Schnittstellen für Programmierung, Betrieb und Administration bereitstellen. OGM bis zur Version 1.1.x (verfügbar für Neo4j bis Version 2.1.x, 2.2.x, 2.3.x) übersetzt Java-APIs in Cypher- und REST-Statements (Transactional HTTP), sodass Cypher wie auch REST für diese Neo4j-Versionen als Vermittler zwischen Java und OGM fungiert.

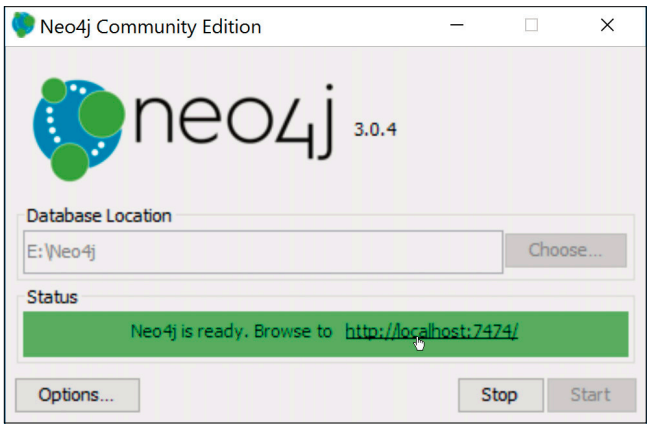
Performante Sprachschnittstellen

Zur Verbesserung der Performance führte Neo4j Version 3 BOLT als leichtgewichtiges und effizientes Protokoll für Datenbankzugriffe ein. Zudem machte der Hersteller BOLT als Protokoll anderen Softwarehäusern für die Entwicklung eigener Produkte zugänglich. Neo Technology selbst realisierte auf Basis von BOLT mit Version 3 erstmals offizielle Sprachschnittstellen für Java, .NET, JavaScript und Python.

Gleichzeitig entstand mit Neo4j Version 3 ein modulares OGM 2.x, das aus dem OGM-Core und im Server-Betrieb (zusätzlich BOLT oder HTTP) oder für Embedded Neo4j zusätzlich Embedded Driver nutzen kann. Ergänzend entschied sich der Hersteller, Stored Procedures für die Datenbank zu unterstützen. Ihre Programmierung erfolgt in Java oder einer auf der JVM (Java Virtual Machine) lauffähigen Scripting-Sprache.



MetaCase will durch Domain-Specific Modeling die bestehende Lücke zwischen Anforderungen und Programm nicht mittels Mapping, sondern durch Code-Generierung und Einbindung von Standardkomponenten überbrücken (Bild 1)



Ein Klick auf den URL-Link im Fenster des Datenbank-Starter-Dialogs von Neo4j öffnet einen Webbrowser mit der Neo4j-Browser-Oberfläche (Bild 2)

Als anwendungsspezifische Sprache (Domain-Specific Language, DSL) bietet JCypher mehrere API-Schichten für Java mit unterschiedlichen Abstraktionsniveaus. Wolfgang Schützelhofer (IoT-Solutions) entwickelt JCypher im Rahmen eines gleichnamigen Projekts über ein GitHub-Repository weiter und macht die DSL darüber hinaus auch in Maven Central verfügbar.

Die verschiedenen Ebenen unterschiedlicher Abstraktionsniveaus von JCypher zielen letztlich darauf ab, das Domain-Modell des Anwendungsbereichs und nicht das Mapping der Programmiersprache auf die Datenbank in den Mittelpunkt zu rücken. Dies soll auch Endbenutzer befähigen, Anwendungen durch Code-Generierung zu entwickeln (Bild 1).

Programmierumgebung für Java

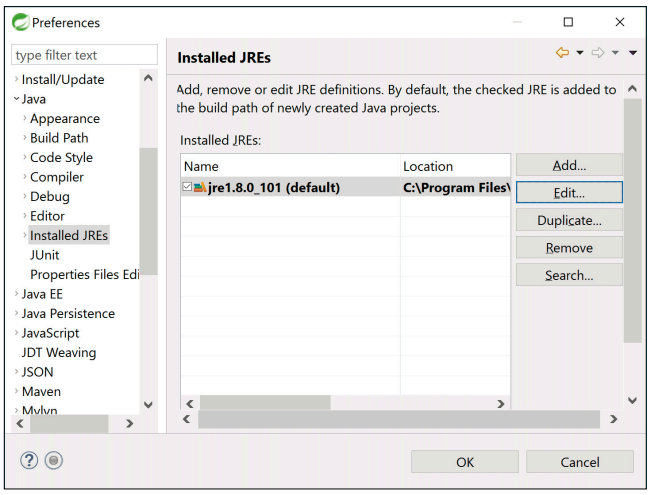
Als DBMS basiert Neo4j seit Version 3 auf Java 8 und setzt deshalb die Installation eines JDK (Java Platform Standard Edition Development Kit) ab Version 8 voraus. Nach der Installation des JDK richtet man eine Umgebungsvariable *JAVA_HOME* ein, die auf das Installationsverzeichnis des JDK zeigt. Zudem sollte die *PATH*-Systemvariable auf das *bin*-Unterverzeichnis von *JAVA_HOME* verweisen, um aus jeder Umgebung heraus Java-Anwendungen ausführen zu können.

Nach dem Einrichten des JDK 8 erfolgt die Installation von Neo4j entweder über die *Setup.exe*-Datei oder durch Entpacken des betriebssystemspezifischen ZIP-, TAR- oder DMG-Archivs. Analog zum JDK referenziert eine Umgebungsvariable *NEO4J_HOME* das Installationsverzeichnis von Neo4j. Das DBMS von Neo4j startet man über das ausführbare Programm der Community- oder der Enterprise-Edition. Daraufhin öffnet sich das Dialogfenster zur Auswahl einer Graphdatenbank und zum Starten einer Instanz des Datenbank-Servers (Bild 2).

In der Praxis setzten sich für die Programmierung mit Java vor allem die Spring Tool Suite (STS), Eclipse IDE for Java Developers, IntelliJ IDEA oder die NetBeans IDE durch. Sehr verbreitet für die Java-EE-Entwicklung hat sich das Spring-Framework, deshalb implementierte Neo Technology für seine Graphdatenbank auch die Java/Spring-Schnittstelle ▶

Spring Data Neo4j (SDN). Als Programmierumgebung basiert die STS-IDE von Pivotal Software auf einer für die Entwicklung von Anwendungen mit Spring entsprechend angepassten Eclipse-IDE.

Die frei zugängliche STS-IDE enthält alle für die Entwicklung von Java-Anwendungen nötigen Zusatz-Tools wie Maven (Build-, Change- und Konfigurationsmanagement), JUnit für Tests, den Java Profiler YourKit und zahlreiche weitere. Zusätzlich liefert Pivotal Software die Programmierumgebung zusammen mit einem zu Apache Tomcat kompatiblen und für Spring optimierten Java-Application-Server Pivotal tc Server aus. Nach dem Entpacken des Download-Archivs startet der Befehl *STS.exe* die Entwicklungsumgebung. Danach sollte man prüfen, ob STS die seitens Neo4j für die Ausführung von Java-Programmen erforderliche Version 8.x der JRE (Java Runtime Environment) tatsächlich verwendet (Bild 3).



Die für Java-EE-Projekte in der STS-IDE verwendete JDK-Version erkundet man über *Window, Preferences* und die Auswahl von *Installed JREs* in der Kategorie *Java* der linken Java-Dropdown-Liste (Bild 3)

JVM als Basis auch für Embedded-Systeme

Da Neo4j in Java und Scala implementiert wurde, setzt der Betrieb einer Graphdatenbank als Basis immer die Installation einer JVM voraus. Im Fall eines vollständigen Embedded-Betriebs befinden sich die Datenbank und die von der Anwendung benötigten JVM-Funktionalitäten innerhalb der über das JRE ausführbaren Anwendungsdatei. Die hohe Performance resultiert aus kurzen Antwortzeiten beim Zugriff auf die in der Anwendung eingebettete Graphdatenbank. Die zur Programmierung benötigten JAR-Dateien befindet sich im *lib*-Unterverzeichnis der Neo4j-Installation oder in einem der IDE zugänglichen Repository (Maven).

Innerhalb der eingesetzten IDE macht man die zur Programmierung von Neo4j benötigten JAR-Dateien dem Entwicklungsprojekt zugänglich. Das Build-Tool der IDE erzeugt aus dem Projekt eine ausführbare JAR-Datei, welche die Anwendung inklusive der Graphdatenbank und die zur Ausführung benötigten JVM-Features enthält. Im einfachsten Fall verwendet eine Anwendung ein Embedded-Neo4j-System nur im Read-only-Modus. Dabei handelt es sich um den für

Maven – leistungsfähige Automation für Java-Builds

Das von der Apache Software Foundation entwickelte Maven gilt als das Standard-Build-Tool für Java-Projekte.

Maven basiert auf einer durch Plug-ins erweiterbaren Architektur und nutzt zur Ausführung von Builds in der Java-Welt gängige Konventionen wie die Ablage von Projektdateien und Ähnliches.

Die XML-Datei *pom.xml*, Project Object Model genannt, steuert die Build-Automation; deren Inhalt beschreibt Abhängigkeiten und Abweichungen von den Konventionen.

Zusätzlich definiert die POM-Datei über spezielle XML-Deklarationen Artefakte, Gruppen-IDs, Gültigkeitsbereiche, Versionen und Weiteres mehr.

Für die Durchführung von Builds greift Maven auf lokale und Remote-Repositories zurück. Das POM-File referenziert die Zugriffe auf die benötigten Repositories. Als Zentrale für die Ablage von Repositories steht The Central Repository im Internet zur Verfügung.

The Central Repository besitzt für Repositories (Maven, sbt) eine komfortable Suchfunktion, die man über *Advanced Search* erreicht

Parallelverarbeitung mittels Scala nutzen

Neo Technology setzt neben Java für die Implementierung von Neo4j zunehmend auch die Open-Source-Programmiersprache Scala ein.

In Scala lassen sich JARs ansprechen und nutzen; umgekehrt war dies nicht immer problemlos möglich. Seit Version 1.6 basiert Scala vollständig auf der JRE – es handelt sich dabei um eine auf der JVM-lauffähige Programmiersprache.

Seit dieser vollständigen Umsetzung von Scala auf der JVM kann man Quellcode-Dateien von Scala und Java im Projekt beliebig kombinieren. Für Scala (und auch Java) gibt es mit sbt (Simple Build Tool) ein spezielles Werkzeug für den Build-Prozess. Gelegentlich bezeichnet man die auf der JVM-lauffähigen Sprachen auch als Scripting Languages.

Außer zur Nebenläufigkeit eignet sich Scala besonders für die Erstellung eigener domänenspezifischer Sprachen (DSLs). Zudem besitzt Scala als funktionale Programmiersprache elegante Sprachmittel, die zu einem prägnanten Programmierstil und damit besserer Code-Verständlichkeit führen.

Domänenspezifische Sprache (DSL)

Eine DSL (Domain-Specific Language) stellt eine formale Sprache für ein bestimmtes Problemfeld (die Domäne) dar.

Im Unterschied zu einer universell einsetzbaren Sprache wie C, Java oder UML rückt bei einer DSL die konkrete fachliche Domäne ins Zentrum.

Aus Sicht der Anwendungsentwicklung benötigt man eine Entwicklungsumgebung als Workbench, um für die jeweils spezifische Fachwelt eine DSL zu erstellen und effizient mit ihr zu arbeiten.

Bekannte Vertreter einer DSL-Workbench stellen die Eclipse-Modeling-Projekte, MetaEdit+ von MetaCase, Microsoft Delve (Codename: Oslo) oder das Meta Programming System (MSP) von JetBrains dar.

Embedded-Systeme auf Kleinstgeräten am häufigsten anzutreffenden Normalfall. Ein konkurrierender Zugriff auf dieselbe Graphdatenbank durch verschiedene Benutzer tritt dann im Fall eines Embedded-Betriebs in der Regel nicht auf.

Um eine Neo4j-Datenbank innerhalb einer Anwendung zu nutzen, muss man entweder eine neue Graphdatenbank erzeugen oder eine bestehende Datenbank öffnen (Listing 1). Danach stehen alle Elemente der Graphdatenbank über die seitens des Core Java API verfügbaren Features zur Verfügung. Natürlich kann man die Einstellungen für den Betrieb von Neo4j auch im Embedded-Fall über eine Konfigurationsdatei *neo4j.conf* im Programm einlesen oder innerhalb des Programms selbst setzen. Soll die Graphdatenbank nur im lesenden Betrieb gestartet werden, setzt man die *read_only*-Einstellungen von *GraphDatabaseSettings* auf den Wert *true*. Vor dem Beenden einer Anwendung muss die Graphdatenbank auch im Embedded-Betrieb gestoppt werden.

Häufig integrieren Gerätehersteller Java for Embedded Devices (Java Embedded) direkt in ihre Hardware (eingebettete Geräte, Embedded Devices). Für Entwickler stellt Oracle ein für eingebettete Geräte optimiertes JDK (Java SE Embedded; kurz: EJDK) zur Verfügung. Dieses spezielle JDK passt man über Profile an die konkreten Bedingungen der jeweiligen Hardware an. Zudem enthält das EJDK ein spezielles Tool JRECreate, um als Entwickler eine für die Hardware und die Anwendung passende JVM zu erzeugen. Für Geräte, die mindestens 70 MByte für ein Java-System zur Verfügung haben, empfiehlt Oracle jedoch, nicht EJDK, sondern die für Desktop und Server gängige Java SE (Standard Edition) einzusetzen.

Als DBMS bietet Neo4j Datenunabhängigkeit: Programmierer müssen nur die Objekte einer Graphdatenbank auf der logischen Ebene kennen. Die Abbildung der logischen Ebene auf die physikalische Ebene der internen Datenstrukturen übernimmt Neo4j automatisch. Die logische Ebene präsentiert sich dem Benutzer gegenüber als abstrakte Datentypen. Das Object Graph Mapping Framework (OGM) von ▶

Listing 1: Beispiele für Java-Quellcode

```
// Neuanlage oder Öffnen einer bestehenden
// Graphdatenbank
String dir=DB_PATH
graphDb = new GraphDatabaseFactory().
newEmbeddedDatabase(dir);
registerShutdownHook(graphDb);

// Einstellungen der Datenbank über
// Konfigurationsdatei lesen
GraphDatabaseService graphDb = new
GraphDatabaseFactory().
  newEmbeddedDatabaseBuilder
  (testDirectory.graphDbDir()).
  loadPropertiesFromFile(pathToConfig +
  "neo4j.conf").
  newGraphDatabase();

// Einstellungen für die Datenbank-Konfiguration
// im Programm setzen
GraphDatabaseService graphDb = new
GraphDatabaseFactory().
  newEmbeddedDatabaseBuilder
  (testDirectory.graphDbDir()).
  setConfig(GraphDatabaseSettings.pagecache_memory,
  "512M").
  setConfig(GraphDatabaseSettings.string_block_size,
  "60").
  setConfig(GraphDatabaseSettings.array_block_size,
  "300").
  newGraphDatabase();

// Instanz einer existierenden Datenbank im
// Read-only-Modus starten

graphDb = new GraphDatabaseFactory().
newEmbeddedDatabaseBuilder(dir).
  setConfig(GraphDatabaseSettings.read_only,
  "true").
  newGraphDatabase();

// Graphdatenbank entsprechend dem Core Java API
// nutzen
...

...
// Vor Beenden der Anwendung die Datenbank
// herunterfahren
graphDb.shutdown();
```

Neo4j bildet über das Core Java API diese abstrakten Datentypen als Plain Old Java Objects (POJOs) im Programm ab, die im Quellcode letztendlich Java-Klassen entsprechen.

Bei POJOs handelt es sich um einfache Java-Objekte, die wenig bis gar keine externen Abhängigkeiten mehr besitzen. Gemäß OGM verwendet man zur Deklaration der POJOs im Quellcode üblicherweise die vordefinierten Java-Annotationen. Sollten diese OGM-Annotationen bei den Java-Klassen der Graphdatenbank fehlen, so spricht man von Non-Annotated Objects, die das Framework über eigene implementierte Konventionen verarbeitet. Für das Speichern von Relationship-Entities (Beziehungen im Graph mit zusätzlichen Attributen) benötigt man allerdings immer OGM-Annotationen; alle anderen POJOs lassen sich auch ohne Java-Annotationen speichern.

Ein POJO stellt ein Objekt der Modellwelt im eigentlichen Sinne der Objektorientierung dar: Es kapselt Daten und erlaubt Zugriffe darauf nur über definierte Schnittstellen (Information Hiding). Das POJO wird dabei als Einheit betrachtet; es besteht also aus Daten und Verhalten und sollte eine möglichst niedrige Kopplung, aber eine starke Kapselung besitzen. Im Regelfall umfasst ein POJO mehr als nur eine Ansammlung von Gettern und Settern. Das konzeptuelle Modell der Datenbank, auch Domain-Object-Modell oder kurz Domain Model genannt, repräsentiert die Graphdatenbank. Danach setzt sich das Domain Model der kompletten Datenbank in der Welt eines Java-Entwicklers aus einer Menge von POJOs zusammen.

Um das OGM-Framework in einer Programmierumgebung wie der STS verwenden zu können, definiert man für das im Projekt eingesetzte Build-Werkzeug die spezifischen Abhängigkeiten in der Build-Datei des Projekts. Neo Technology unterstützt alle in der Java-Welt gängigen Build-Tools wie Maven, Gradle, Ivy und andere. Während der Programmierung mit Java sollten in der IDE für im Projekt definierte Abhängigkeiten tatsächlich auch die notwendigen Aktualisierungen vorgenommen werden. In der Regel führt man diese auf Projektebene manuell von Hand über das *Update*-Menü des Build-Werkzeugs durch: Auswahl des Projekts im Package-Explorer der STS-IDE und bei Einsatz von Maven die Befehlsfolge: *Maven, Update Project …* (Bild 4).

Java-EE-Programme einfacher entwickeln mit Spring

Das Open-Source-Framework Spring eignet sich besonders für die schnelle Implementierung verteilter, mehrschichtiger und komponentenbasierter Architekturen.

Das Framework selbst besteht aus einer großen Sammlung mächtiger Module für die Entwicklung unternehmensweiter Anwendungen gemäß der Java-EE-Spezifikation.

Um die oben genannten Ziele zu erreichen, unterstützt Spring unter anderem zwei wichtige Programmierparadigmata:

- Dependency Injection (DI): Ein Inversion-of-Control (IoC)-Container konfiguriert und verwaltet mittels Reflection Java-Objekte. Dabei injiziert ein Methodenaufruf zur Laufzeit vorhandene Abhängigkeiten zwischen Komponenten, sodass eine Implementierung entfällt. Dieser Ansatz reduziert die Abhängigkeiten zwischen den Komponenten, verbessert deren Wartbarkeit und erhöht zugleich ihre Wiederverwendbarkeit.
- Aspektorientierte Programmierung (AOP): Mittels Annotationen oder XML-Schemata lassen sich Querschnittfunktionen wie Logging, Druck oder Fehlerhandling realisieren, sodass sie nicht für jede einzelne Komponente immer wieder neu und separat implementiert werden müssen.

Neo4j – DBMS mit flexiblen Betriebsmodi

Neo4j unterstützt als DBMS zwei verschiedene Betriebsformen in der Praxis:

- Embedded (In-Process): Bezeichnet den Betrieb von Neo4j als Teil einer Anwendung, die als Schnittstelle zur Datenbank das Core Java API nutzt. Eine eingebettete Datenbank eignet sich speziell für Klein- und Kleinstgeräte (mobile Geräte) mit einer schwachen Rechnerleistung. Embedded Systeme bieten auch Performance-Vorteile für Computercluster, die Hochverfügbarkeit bieten müssen.
- Server: Dabei handelt es sich um die gängige Betriebsform aller relationalen Datenbanksysteme; im Server-Betrieb verbinden sich verschiedene Anwendungen als Client mit einer Neo4j-Datenbank und bearbeiten parallel die Daten. Neo4j verfügt dabei über ein Transaktionsmanagement im Sinne von ACID (Atomarität, Konsistenzerhaltung, Isolation, Dauerhaftigkeit). Neben dem Core Java API kann man für diese Betriebsform auch den HTTP- oder BOLT-Treiber einsetzen.

OGM-Annotationen für Knoten

Eine Datenbank besteht aus Entitäten und Beziehungen mit Attributen beziehungsweise Eigenschaften, die über eine Beschreibungssprache definiert und im Data Dictionary abgelegt sind. Im Fall der Graphdatenbank Neo4j stellen die Entitäten die Knoten dar, die eine beliebige Anzahl von Attributen (sogenannte Key-Value-Paare) enthalten. Um verschiedene Rollen der Entitäten im Anwendungsbereich eines Graphen abzubilden, versieht man den Knoten mit Labels. Ein Knoten der Graphdatenbank besteht also aus Attributen, auch Properties genannt, und eventuell mehreren Labels.

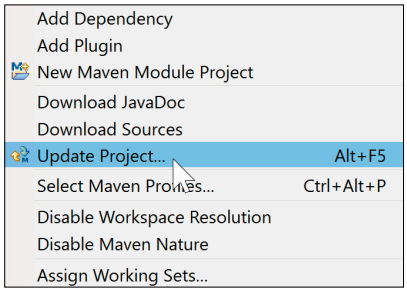
Beziehungen zwischen den Entitäten bilden in Neo4j Relationships (Kanten) ab. In der Graphdatenbank verbindet eine Relationship immer zwei Knoten: einen Start- und einen End-Knoten. Dadurch ist stets eine Richtung für die Relationship vorgegeben. Als weitere Bauteile kann eine Relationship eine beliebige Anzahl von Properties aufnehmen, die meist quantitative Eigenschaften wie Kosten, Entfernungen, Bewertungen oder Ähnliches beschreiben. Um den Knoten einer Beziehung zu löschen, muss immer auch die zugehörige Beziehung als Verbindung aus der Graphdatenbank entfernt werden. Diese Regel unterstützt das DBMS Neo4j direkt als systemimmanente Integritätsbedingung.

Insgesamt betrachtet implementiert Neo4j als Datenmodell einen Labeled-Property-Graphen. Im Java-Quellcode definiert man einen Knoten als Java-Klasse und fügt dieser die Annotation *@NodeEntity* hinzu. Um einen Knoten mit einem Label zu versehen, richtet man bei der Java-Annotation einer *@NodeEntity* eine *label*-Property mit dem zugehörigen Namen des Labels ein. Listing 2 definiert beim Knoten *Mitarbeiter* das Label *AbtLeiter* als Parameter der Annotation.

Zusätzlich kann man zur Laufzeit eines Programms weitere Labels einführen oder bestehende Labels löschen. Hierzu steht die *@Labels*-Annotation zur Verfügung: Bei der Entity *Abteilung* (in der Fachsprache der Graphentheorie spricht man von Knoten) deklariert das Listing eine Liste *rolle*, um später Labels für einen solchen Knoten dynamisch anlegen zu können.

Das OGM-Framework von Neo4j implementiert für die Verarbeitung einer Relationship im Java-Quellcode die nachfolgenden Konventionen:

- Im Start-Knoten der Beziehung gibt es einen Verweis auf einen Datentyp des End-Knotens; im Fall einer mehrwertigen Beziehung greift man beim Datentyp auf einen Java-Set-Typ zurück.



Aktualisierungen für das Projekt seitens des Build-Tools (hier: Maven) nimmt man über den Package-Explorer der Java-IDE von Hand vor (Bild 4)

Listing 2: Datenbank-Schema im Java-Quellcode

```
@NodeEntity
public class Abteilung {
  private String bezeichnung;
  private String abtLeiter;
  private Set<Mitarbeiter> mitarbeiter;

  private Set<Standort> standorte;

@Labels
  private List<String> rolle = new ArrayList<>();
}

@NodeEntity(label="AbtLeiter")
public class Mitarbeiter {
  private String vorname;
  private String nachname;

  private Abteilung abteilung;
}
```

Listing 3: Datenbank-Schema mit Relationship-Typen

```
@NodeEntity
public class Abteilung {
  private String bezeichnung;
  private String abtLeiter;

  @Relationship(type = "BESTEHT_AUS_MITARBEITERN")
  private Set<Mitarbeiter> mitarbeiter;

  private Set<Standort> standorte;

}

@NodeEntity(label="AbtLeiter")
public class Mitarbeiter {
  private String vorname;
  private String nachname;

@Relationship(type = "BESTEHT_AUS_MITARBEITERN",
direction = "INCOMING")
  private Abteilung abteilung;
}
```

- Im End-Knoten existiert ein Zeiger als Verweis auf die Klasse des Start-Knotens der Beziehung.

Um die POJOs als Knoten inklusive ihrer Relationships, den Kanten, in der Graphdatenbank zu speichern, wendet das OGM-Framework diese Konvention an. Um direkt im Java-Quellcode eine Beziehung zwischen Start- und End-Knoten ersichtlich zu machen, greift man auf die *@Relationship*-Annotation zurück.

Eine *@Relationship*-Annotation beschreibt sowohl den Typ der Beziehung zwischen den Knoten als auch die jeweilige Richtung (Listing 3). Die Kante nimmt vom Start-Knoten der Beziehung eine ausgehende Richtung *(OUTGOING)* ein und vom End-Knoten eine hineingehende Richtung *(INCOMING)*. Bei der *@Relationship*-Annotation einer Beziehung legt man einen Typ über die *type*-Property und die Richtung über deren *direction*-Parameter fest. Fehlt bei einer Kante ein *direction*-Parameter, so belegt das OGM-Framework die Richtung automatisch mit dem Standardwert *OUTGOING*.

Eine Relationship mit eigenen Properties muss man als eigenständige Klasse im Java-Quellcode abbilden. Derartige Beziehungen, die über eigene Attribute verfügen, bezeichnet man auch als Relationship Entity (Beziehungsentität). Für die Definition des Schemas im Java-Programm sieht das OGM-Framework eine spezielle Annotation *@RelationshipEntity* vor. Ferner kennzeichnet man den an der Beziehung beteiligten Start-Knoten und den End-Knoten mittels einer *@StartNode*- und einer *@EndNode*-Annotation. Listing 4 definiert eine Beziehungsentität *Standort* mit den beteiligten Knoten *(Abteilung* und *Mitarbeiter)* und zwei eigenen Properties *strasse* und *ort*. ▶

Eigene Automatismen

Jede Entity einer Datenbank besitzt auf der internen Ebene immer ein eindeutiges Identifikationsmerkmal, einen internen, technischen (manchmal spricht man auch von physischen) Systemschlüssel, den das Datenbanksystem (DBS) eigenständig, das heißt selbstständig automatisch, verwaltet. In einer Neo4j-Graphdatenbank nennt sich dieser Schlüssel *id;* eine solche ID gibt es für jedes extern zu speicherndes Element des Graphen (Bild 5). Mittels der ID findet Neo4j ein angesprochenes Element der Datenbank eindeutig auf der internen Ebene und rekonstruiert so aus den auf einem externen Speichermedium abgelegten Datenstrukturen den aus Anwendungssicht benötigten Graphen – die externe Ebene.

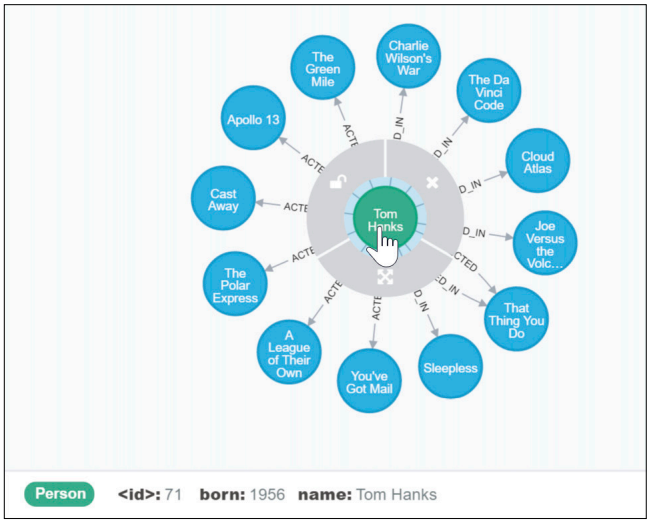
Im Schema einer Neo4j-Datenbank definiert man eine ID über ein numerisches Feld der Klasse *Long* für jede Entität der Datenbank. Damit erweitert sich das Datenbank-Schema zur Deklaration einer Entität als Java-Klasse um eine Zeile für die Definition einer privaten Klassenvariablen *id* und eine öffentliche Methode, um deren Wert mittels Getter zu lesen (Listing 5). Findet Neo4j eine derartige Deklaration eines Long-ID-Felds, benutzt das DBMS dieses Feld automatisch für den eigenen internen Systemschlüssel. Aus Gründen der Lesbarkeit und Konsistenz sollte man das Long-ID-Feld aber besser über eine *@GraphId*-Annotation im Java-Quellcode kennzeichnen.

Neo Technology empfiehlt, über die zusätzliche Definition einer abstrakten Klasse *Entity* die generelle Funktionalität eines internen Systemschlüssels bereitzustellen. Anstatt bei jedem Element der Graphdatenbank eine eigene Long-ID zu deklarieren, übernimmt diese Aufgabe dann die abstrakte Klasse *Entity*. Anschließend referenziert man diese abstrakte Klasse in der Schemadefinition des jeweiligen Elements der Graphdatenbank mittels der Java-Anweisung *extends Entity*. Diese Erweiterung reicht die Definition der IDs auf der internen Ebene automatisch weiter und macht so eine Deklaration einer Long-ID und die *@GraphId*-Annotation für die einzelnen Entitäten des Graphen hinfällig.

Eleganter und einfacher wäre es, wenn Neo4j als DBMS – wie alle relationalen Systeme auch – seine internen Systemschlüssel vollständig alleine erzeugen und verwalten würde.

Listing 4: Definition einer Relationship Entity

```
@RelationshipEntity(type = "ARBEITET_AM")
public class Standort {

  @StartNode
  private Abteilung abteilung;

  @EndNode
  private Mitarbeiter mitarbeiter;

  private String strasse;
  private String ort;
}
```



Der Neo4j-Browser zeigt das *id*-Feld in der Statusleiste zwar an; es gehört aber nicht zu den Property-Keys der Graphdatenbank (Bild 5)

Im Unterschied zu gängigen relationalen DBMS muss ein Anwendungsprogrammierer im externen Schema einer Neo4j-Datenbank interne Systemschlüssel stets bei jeder Entität (Knoten, Kante) deklarieren und im schlechtesten Fall gegebenenfalls auch lesen. Dadurch entstehen Unzulänglichkeiten, die zu einer Aufweichung der Grenzen zwischen konzeptuellen, internen und externem Datenmodell/Schema führen. Dies könnte auch die (logische) Datenunabhängigkeit in Frage stellen.

Alle relationalen DBMS verlangen im externen Schema der Datenbank innerhalb eines Anwendungsprogramms keine Deklaration von internen Systemschlüsseln. Ein Verzicht auf derzeit notwendige Definitionen von Long-ID-Feldern im externen Schema einer Neo4j-Datenbank innerhalb eines Java-

Listing 5: Definition eines internen Systemschlüssels

```
@NodeEntity
public class Abteilung {

  private Long abteilId;

  public Long getId() {
    return abteilId;
  }

  private String bezeichnung;
  private String abtLeiter;

  @Relationship(type = "BESTEHT_AUS_MITARBEITERN")
  private Set<Mitarbeiter> mitarbeiter;

  private Set<Standort> standorte;

}
```

Programms ließe sich leicht im OGM-Framework implementieren. Dazu eignet sich zum einen die beschriebene abstrakte Entity-Klasse, zum anderen könnte man den erforderlichen Code während des Übersetzungsprozesses oder des Builds einer Anwendung hinzugenerieren. Dann kämen Entwickler erst gar nicht in Versuchung, interne Neo4j-Primärschlüssel extern verwenden zu wollen.

Unter Kontrolle der Anwendungen

Für alle DBMS, also auch für Neo4j, gilt: Grundsätzlich sollte man niemals einen internen Systemschlüssel außerhalb des die Entität nutzenden Anwendungssystems für anderweitige Zwecke extern speichern. Es kann nicht garantiert werden, dass ein bestimmter ID-Wert immer die gleiche Entität eindeutig identifiziert. Nach dem Löschen einer Entität kann eine ID-Ausprägung innerhalb der Graphdatenbank für eine andere Entität von Neo4j wiederverwendet werden. Sollte man also Long-ID-Felder einer Neo4j-Datenbank extern speichern, kann dies zu Problemen bei der Identifikation eines Elements in der Datenbank führen.

Innerhalb einer Anwendungslandschaft muss man jedoch häufig Elemente einer Datenbank eindeutig identifizieren. Für diese Identifikation kommen auf der logischen Ebene der Anwendung sogenannte Primärschlüssel zum Einsatz. Im Fall von Neo4j (Graph-Modell) sollte man hierfür, wie bei jedem anderen Datenmodell auch, nicht die internen IDs (Systemschlüssel), sondern eigene, von den Anwendungen verwaltete Surrogat-Keys (Stellvertreterschlüssel) verwenden.

Java-Annotationen im Quelltext

Seit Version 5 kennt Java als Sprachelement Annotationen, um Metadaten in den Quellcode einzubinden.

Eine Annotation beginnt mit dem @-Zeichen, darauf folgt ein möglichst sprechender Name (beginnend mit einem Großbuchstaben) und optional eine durch Komma getrennte und in Klammern gefasste Parameterliste.

Während der Übersetzung verarbeitet ein Annotation-Prozessor mittels Compiler-Plug-in die Annotationen im Quelltext; eventuell finden entsprechend ihrer Bedeutung Auswertungen zur Laufzeit statt.

Eigene Annotations-Typen definiert man über das Schlüsselwort *@interface* durch Erweiterung der Schnittstelle *java.lang.annotation.Annotation;* als Ergebnistypen ihrer Methoden sind nur bestimmte Typen erlaubt:

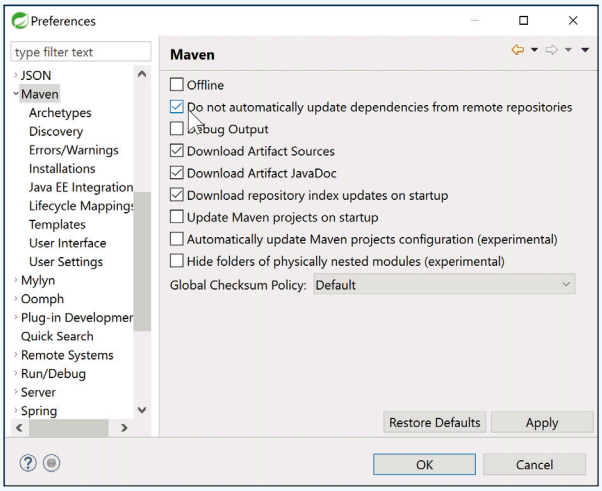
```
// Deklaration eines Annotations-Typs:
public @interface MeineAnnotation {
  double param1();
  String param2();
}

// Anwendung des Annotations-Typs im Java-Programm
@MeineAnnotation(param1=wert1, param2=wert2)
```

Einstellungen des Build-Tools in der IDE

Sollte ein von Hand angestoßenes Projekt-Update das Build-Werkzeug im Projekt nicht durchführen, so muss man dessen Vorgaben in der Programmierumgebung prüfen.

Die Einstellungen des Build-Werkzeugs in der IDE findet man im zugehörigen Preferences-Dialog; dort wählt man in der linken Auswahlliste das für den Build im Projekt eingesetzte Werkzeug aus (zum Beispiel Maven). Rechts erscheinen dann die verfügbaren Einstellungen auf allgemeiner Ebene.

Für Dependencies darf das automatische Update über die Remote-Repositories nicht ausgeschaltet sein. Sinnvollerweise sollte ein weitergehender Download spezifiziert sein; beispielsweise im Fall von Maven: Download von Artifact Sources und der Artifact JavaDoc.



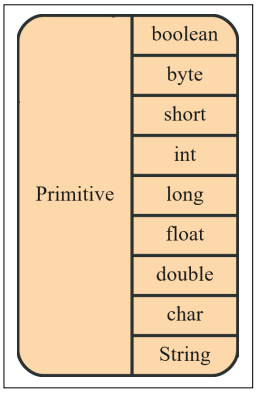
Eine integrierte Entwicklungsumgebung (hier: STS-IDE) nimmt im Bedarfsfall verschiedene Aktualisierungen über die im Projekt zugeordneten Repositories automatisch vor, falls deren Automatismen nicht deaktiviert sind

In einer Neo4j-Datenbank empfiehlt es sich, für die Implementierung von Schlüsselwerten der Surrogat-Keys auf die UUID-Bibliothek von GraphAware zurückzugreifen. Die Features dieser Bibliothek stellen neben Transparenz auch sicher, dass man einen Surrogat-Key nicht versehentlich ändert oder löscht. Das Akronym UUID stammt aus dem Fachjargon von Java und steht für Universally Unique Identifier. Alternativ kann man auch den UUID-Standard der Open Software Foundation (OSF) oder einen anderen für die JVM-verfügbaren UUID-Generator einsetzen.

Neo4j unterstützt als Datentypen für die Werte eines Attributs alle seitens Java bekannten primitiven Typen und aus diesen aufgebaute Arrays (Bild 6). Geschachtelte, strukturierte Datentypen wie Record-Strukturen oder Dateitypen lassen sich derzeit nicht direkt mit den Neo4j-Datentypen implementieren. Die Ausprägungen der jeweiligen primitiven Datentypen richten sich im Wertebereich nach der Java-Language-Spezifikation. ▶

Um Property-Felder unmittelbar bei der Deklaration einer Java-Klasse ersichtlich zu machen, verwendet man die optionale *@Property*-Annotation. Anhand einer Parameter-Deklaration *name* vergibt man im Neo4j-Schema des Java-Programms einen Namen für eine Property: *@Property(name="Nachname") String nachname*.

Felder einer Java-Klasse mit der *@Transient*-Annotation speichert das DBMS nicht im Graphen; dabei handelt es sich um flüchtige, sogenannte nichtpersistente Felder. Transiente Felder muss man im Java-Schema im Unterschied zu den persistenten explizit kennzeichnen; sie eignen sich vor allem für aus der Datenbank abgeleitete Feldwerte. Persistente Felder eines Graphen müssen nicht mit dem in Java gängigen Schlüsselwort *final* deklariert werden.



Die Werte der Felder (Label, Attribute oder Properties) einer Neo4j-Datenbank entsprechen den gängigen Standard-Java-Datentypen (Bild 6)

OGM wandelt die Werte aller Datentypen automatisch in einen String um, vorausgesetzt eine Konvertierung ist möglich. Zusätzlich besitzt das OGM-Framework weitere standardmäßig vorhandene Converter, welche die nachfolgenden Datentypen in Java wie folgt konvertieren:

- *binary data:* in einen base-64 String,
- *java.math.BigDecimal*: in einen String,
- *java.math.BigInteger:* in einen String,
- *java.util.Date:* in einen String im ISO-8601-Format,
- *java.lang.Enum:* gemäß der *Enum.name()*- und der *Enum.valueOf()*-Methode.

Das OGM-Framework verarbeitet auch Sammlungen primitiver oder konvertierbarer Datentypen. Neo4j speichert diese als Array des jeweiligen Typs oder gegebenenfalls als String ab. Diese Standard-Converter erweitert OGM um zwei Tages-Konverter: *@DateString* und *@DateLong*. Bei der Deklaration eines Tagesdatums über *@DateString("yy-MM-dd")*, wandelt OGM das Datum in eine gemäß der in Klammern angegebenen String-Darstellung um, während die *@DateLong*-Annotation das Datum in einen *long*-Value überführt.

Schlussendlich befähigt OGM den Programmierer über die *@Convert*-Annotation, eigene Custom-Konvertierer für Datentypen zu spezifizieren (Listing 6). Dazu implementiert der Programmierer über das AttributeConverter-Interface des OGM-Frameworks zwei Methoden in einer Konverter-Klasse (Listing 7). Die beiden Methoden des Custom-Konvertierers wandeln den gespeicherten Wert in den Property-Wert des Graph-Elements um und konvertieren umgekehrt den Property-Wert der Entity in den extern in der Datenbank zu speichernden Wert.

Listing 6: Definition eines Custom-Typ-Konvertierers

```
@NodeEntity
public class Abteilung {

  @Convert(AbtNameKonverter.class)
  private AbteilungsName value;
}
```

Listing 7: Implementierung der Konverter-Klasse

```
public class AbtNameKonverter implements Attribute
Converter<ZeichenKetteDarstellung, String> {

  @Override
  public String toGraphProperty
  (ZeichenKetteDarstellung value) {
    return /* Code zu Umwandlung in die
    Graph-Darstellung */;
  }

  @Override
  public ZeichenKetteDarstellung
  toEntityAttribute(String value) {
    return /* Code zu Umwandlung in die
    Entity-Darstellung */;
  }
}
```

JCF für wiederverwendbare Datensammlungen

Beim Java Collections Framework (JCF) handelt es sich um eine Java-Bibliothek mit wiederverwendbaren Datenstrukturen.

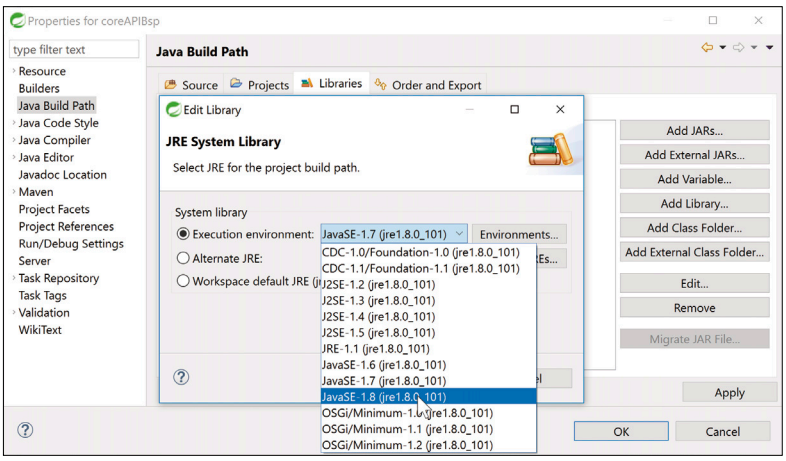
JCF ergänzt die gängigen Datenstrukturen von Java um weitere, wie Listen, Stacks, Queues, Mengen und Maps. Für jede dieser Datenstrukturen gibt es ein spezielles *java.util*-Interface, das die typischen Methoden für die Verarbeitung ihrer Elemente definiert. Erweitert man das Interface oder führt man Implementierungen der jeweiligen Datenstruktur ein, so erhält man zusätzliche Datenstrukturen.

Mengen im mathematischen Sinne definiert das *java.util.Set*-Interface. Eine Menge kennt keine Reihenfolge ihrer Elemente, besitzt also keinen Index für einen Direktzugriff; zudem befinden sich in einer Menge auch keine doppelten Elemente. Die Implementierung einer Menge erfolgt über die Klassen *HashSet, LinkedHashSet, TreeSet* und weitere.

Durch Erweiterung des *java.util.Set*-Interface erhält man das *java.util.SortedSet*-Interface. Die Elemente eines Sorted Sets besitzen eine Sortierung, sodass ein erstes und ein letztes Element in der Datensammlung existiert. Die Implementierung des *SortedSet*-Interface erfolgt in der Klasse *java.util.TreeSet*.

IDE-Konfiguration

Die neue Version 3 des Neo4j-DBMS basiert zwar auf dem JDK 8, doch die Einstellungen einiger aktuell ausgelieferter Java-IDEs stehen immer noch auf den Konventionen der vorherigen Java-Sprachversion 7. Daher muss man in diesen Java-IDEs nach erfolgter Installation für die Kompilierung als JDK Compliance die aktuellen Vorgaben von Java 8 zugänglich machen. Hierzu wählt man beispielsweise in der STS-IDE über *Window, Preferences* in der Kategorie *Java, Compiler* den Eintrag *1.8* für den *Compiler compliance level* sowie die Option *Use default compliance settings* (Bild 7) aus.

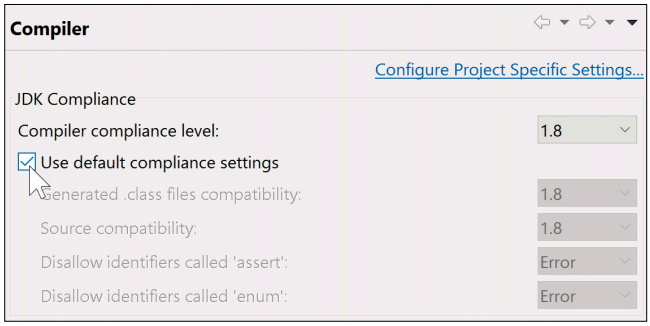


Das Build-Tool nutzt für die Kompilierung einen vorgegebenen Java Build Path; dieser sollte mit der verwendeten JRE übereinstimmen (Bild 8)

Anschließend berücksichtigt STS für jedes neue Java-Entwicklungsprojekt die neuen Sprachkonventionen von Java 8. Zusätzlich müssen bereits vorhandene Java-Projekte individuell von Hand über deren Projekt-Properties umgestellt werden. In einem Java-Projekt arbeitet das Build-Tool der IDE für die Übersetzung des Java-Quellcodes nur dann fehlerfrei, wenn es auf die JRE des JDK 8 zugreifen kann. Dies kontrolliert man im Project Explorer bei den Projekt-Properties: In der Kategorie *Java Build Path* muss die korrekte JRE System Library eingestellt sein.

Dazu klickt man im Dialog *Java Build Path* auf den Registerreiter *Libraries;* dort muss beim *Java Build Path* für *JARs* und *class folders* der Eintrag *JRE System Library [JavaSE-1.8]* stehen. Gegebenenfalls legt man diese Version über die *Edit*-Schaltfläche und das sich öffnende Dialogfenster fest (Bild 8). Alle Bibliotheken für die Java-Programmierung mit Neo4j befinden sich im *lib*-Unterverzeichnis der DBMS-Installation. Dies setzt unter Windows allerdings voraus, dass die Neo4j-Installation über den Download des ZIP-Archivs erfolgte, da diese derzeit die *Setup.exe* nicht enthält.

Ordnet man einem OGM-Projekt die Neo4j-JARs über dieses *lib*-Unterverzeichnis zu, so benutzt das Build-Tool der IDE immer die zur vorhandenen Neo4j-Installation passenden JARs. Bindet man hingegen alternativ über die Anweisungen des eingesetzten Build-Tools (Maven, Ivy und andere) (Listing 8) die Neo4j-JARs ein, so legt dies direkt alle Abhängigkeiten offen. Gleichzeitig unterstützt der Einsatz eines Build-Tools alle Vorgaben des Entwicklungsprojekts und ermöglicht es, gezielt bestimmte Versions- oder Release-Stände zu testen (Bild 9). Um eine automatische Aktualisierung von Änderungen in Remote-Repositories zu verhindern, installiert man ein lokales Repository und setzt Maven in den Offline-Modus.



Abweichende Einstellungen der IDE-Vorgaben spezifiziert man auf Projektebene; für bereits vorhandene Projekte macht man über *Configure Project Specific Settings ...* eventuelle Abweichungen wieder rückgängig (Bild 7)

Transaktions- und Session-Management

Bei Neo4j handelt es sich um ein DBMS mit ACID-Eigenschaften, demnach erfolgen Datenänderungen in einer Graphdatenbank immer innerhalb von Transaktionen.

Neo4j OGM kennt spezielle Objekte für die Verwaltung und Verarbeitung von Transaktionen. Eine Transaktion basiert immer auf einem Session-Objekt, das man über eine SessionFactory (Listing 9) erzeugt: Im Beispiel befindet sich das externe Modell der Datenbank im Package *org.neo.* ▶

Interne Systemschlüssel – kein Neo4j-Spezifikum

Jedes Datenbankmodell, ob relational, objektorientiert, netzwerkartig oder hierarchisch, kennt interne technische Schlüssel. Dabei handelt es sich um interne Primärschlüssel, die Elemente der internen Ebene einer Datenbank eindeutig identifizieren.

Einige relationale DBMS, zum Beispiel Oracle oder PostgreSQL, legen diesen Identifikator offen: Oracle bezeichnet ihn mit ROWID-Pseudocolumn; in PostgreSQL gibt es für ihn gleich mehrere Alias-Typen wie Object identifiers (OIDs), *regproc, regoper, regclass, regtype* und weitere.

SQL Server und MySQL dokumentieren nur wenig über ihren Systemschlüssel. Während SQL Server eine *rid (File:Page:Slot)* kennt, die man über die virtuelle Spalte *%%physloc%%* erreicht, muss man in MySQL über eine benutzerdefinierte Variable und die Funktion *row_number* den intern vorhandenen Systemschlüssel nach außen simulieren.

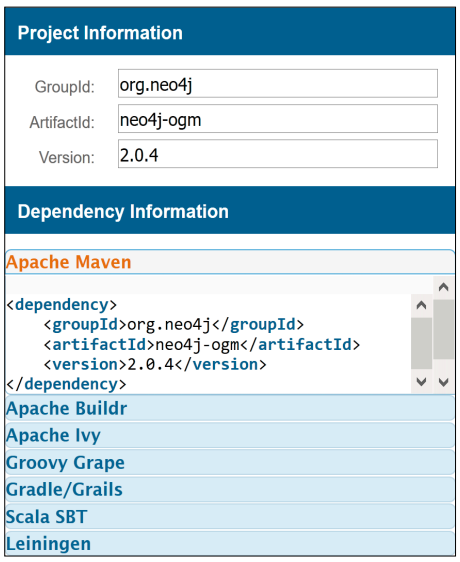
models. Der Konstruktor der SessionFactory durchsucht *org.neo.models* nach allen Entity- und Relationship-Klassen und erzeugt anschließend daraus die für das Object Mapping benötigten Metadaten.

Das seitens OGM 2.x für die Abwicklung der Kommunikation notwendige Protokoll (BOLT, HTTP, Embedded) legt man standardmäßig in einer Datei *ogm.properties* fest (Listing 10). Diese Vorgehensweise macht die Programme völlig unabhängig von dem in einer Anwendung konkret zu verwendenden Protokoll. Will man für eine Anwendung ein anderes Protokoll nutzen, so genügt es, die Einstellungen in der Konfigurationsdatei *ogm.properties* abzuändern. Dieser Sachverhalt ermöglicht es auch, Integrationstests im Embedded-Betrieb durchzuführen und die Anwendung im Erfolgsfall direkt in die Produktion auf eine Client-Server-Umgebung zu überführen. Bei der Produktionsübergabe fallen aufgrund dieser Protokollunabhängigkeit keinerlei Änderungen im Quellcode der Anwendung mehr an.

Ein Session-Objekt stellt Methoden bereit, um ein Element der Graphdatenbank zu erzeugen, zu laden, zu speichern oder zu löschen. Man spricht von CRUD-Operationen, die das Service-Interface als Session-Operationen implementiert. Neo4j OGM kennt für das Verarbeiten (nicht nur für das Laden) der Elemente eines Graphen das Konzept des Persistence Horizon, manchmal einfach Depth genannt. Die Depth im OGM-Framework beschreibt, welche Elemente ausgehend vom Ausgangsobjekt zusätzlich bei den Session-Operationen zu berücksichtigen sind: Eine Depth von *0* berücksichtigt nur das Ausgangsobjekt, während bei einer Depth von *1* zusätzlich zu diesem noch alle von ihm erreichbaren Nachbarobjekte aus dem Graphen hinzukommen.

Optimierung von Transaktionen

Im Bedarfsfall aktiviert man eine konkret gewünschte Depth im Programm über ein Depth-Argument bei den Session-Methoden *(DEPTH_LIST=0, DEPTH_ENTITY=1)*. Während *DEPTH_LIST* die Depth einer Trefferliste definiert, die man zum Beispiel über *loadAll* lädt, gibt *DEPTH_ENTITY* die



The Central Repository Search Engine zeigt Zusatzinformationen für verschiedene Build-Tools (Maven, Buildr, Ivy, Grape und weitere) für den Aufbau und die Definition von Abhängigkeiten an (Bild 9)

Interner Systemschlüssel versus Surrogatschlüssel

Ein interner System/Primärschlüssel stellt keinen Surrogate-Key (Stellvertreterschlüssel) dar – eine wichtige Feststellung!

Analog zu einem internen Schlüssel dient ein Stellvertreterschlüssel, häufig als Primärschlüssel verwendet, der eindeutigen Identifikation eines Elements der Datenbank. Beide Schlüsselformen basieren nicht auf einem Datum eines Objekts der Datenbank, vielmehr werden sie künstlich erzeugt.

Im Unterschied zum internen Systemschlüssel, den das Datenbanksystem selbst bereitstellt und verwaltet, erzeugen und verwalten Anwendungen ihre eigenen Surrogatschlüssel.

Listing 8: Maven-Abhängigkeiten für BOLT-Treiber

```
<dependency>
   <groupId>org.neo4j</groupId>
   <artifactId>neo4j-ogm-bolt-driver</artifactId>
   <version>2.0.3</version>
</dependency>
```

Listing 9: Code-Rahmen für SessionFactory

```
public class Neo4jSessionFactory {

  private final static SessionFactory
  sessionFactory = new SessionFactory
  ("org.neo.models");

  private static Neo4jSessionFactory factory = new
  Neo4jSessionFactory();

  public static Neo4jSessionFactory getInstance() {
    return factory;
  }
  private Neo4jSessionFactory() {
  }

  public Session getNeo4jSession() {
    return sessionFactory.openSession();
  }
}
```

Listing 10: Minimale ogm.properties-Datei für BOLT

```
driver=org.neo4j.ogm.drivers.bolt.driver.BoltDriver
URI=bolt://neo4j:password@localhost
encryption.level=NONE
```

Weitere UUID-Generatoren für Java-Anwendungen

In der Praxis eignen sich für Java die folgenden alternativen UUID-Generatoren; eine Auswahl sollte man ausgerichtet auf den Anwendungsfall treffen:

- Commons Id: Eine über Apache Commons-Projekt bereitgestellte Java-Komponente, die über verschiedene Algorithmen zur Generierung von Identifikatoren verfügt.
- java.util.UUID: Eine seit Java 5 vom JDK bereitgestellte Klasse.
- Java Uuid Generator (JUG): Von Tatu Saloranta programmierter UUID-Generator, der längere Zeit auch von Neo Technology genutzt wurde.
- eaio-uuid: Stammt aus der Feder von Stephen Connolly.
- UUID generator: Entwicklung von Johann Burkard, ausgehend von den Konzepten des eaio-uuid-Projekts

Depth für eine zu ladende Entity vor. Standardmäßig verwendet Neo4j OGM den Wert *1* als Voreinstellung für die Depth beim Laden, während beim Speichern der negative Wert *-1* genutzt wird. Kennt man diese Konventionen, so muss man sich als Programmierer meist nicht mehr damit auseinandersetzen. Zudem erlaubt die Vorgabe eines Depth-Werts, die Elemente der Graphdatenbank aus Sicht des Laufzeitverhaltens optimal zu verarbeiten. Alle Elemente eines Graphen lädt man über einen Depth-Wert von *-1;* was aus Performancegründen in der Regel nur für kleine Graphen, die vollständig in den Hauptspeicher passen, sinnvoll ist. Die Methode *beginTransaction* eines Session-Objekts (Listing 11) erzeugt ein spezielles Transaction-Objekt zur Verwaltung einer Transaktion. In diesem Fall führt das OGM-Framework über dessen *commit*-Methode einen Datenbank-Commit durch oder verwirft alle Änderungen mittels der Methode *rollback*.

Innerhalb des Transaktions-Managements stehen über ein Session-Objekt die Methoden *load, loadAll, save* und *delete* zur Verfügung. Dabei legt das Depth-Argument der jeweiligen Methode fest, welche Objekte jeweils konkret die auszuführende Verarbeitung einbezieht. Die zuvor genannten Persistenz-Methoden eines Session-Objekts setzen einen impliziten Datenbank-Commit bei Änderungen ab. Allerdings muss bei einem derartigen Auto-Commit die Anwendung selbst die Konsistenz der Datenbank gewährleisten. Diese Anforderung entsteht, da Neo4j OGM Save-Operationen nicht automatisch für alle geänderten Elemente der Graphdatenbank durchführt.

Um Änderungen in einem Element der Datenbank persistent zu speichern, muss das zugehörige Session-Objekt eine *save()*-Methode ausführen. Sollte eine Transaktion über eine *close*-Methode beendet werden, so macht nur eine explizite Save-Anweisung das gewünschte Element in der Graphdatenbank persistent. Neo4j OGM führt keine automatischen Save-Anweisungen für Session-Objekte durch, auch dann nicht, wenn diese seitens der Anwendung geändert wurden. Sollte ein zu speicherndes Objekt im Graphen zu einem Dangling Pointer führen, so verhindert die JVM deren Entstehen, sodass alle beteiligten Objekte kaskadierend gespeichert werden.

Seit Version 2.0.4 verfügt Neo4j OGM über einen eigenen Event-Mechanismus für Persistenz-Ereignisse, sogenannte Persistence Events. Um auf deren Eintreten in einer Anwendung entsprechend zu reagieren, erzeugt man für Persistenz-Ereignisse eigene Event Listener. Dabei bezieht sich ein Event Listener nicht nur auf das Einstiegselement, sondern auch auf alle mit ihm verbundenen und durch das Ereignis ebenfalls betroffenen Objekte. Leider gewährleistet diese Deep-Persistence-Strategie des OGM-Frameworks nicht automatisch die Integrität der Graphdatenbank, sodass es unter Umständen zu inkonsistenten Daten kommt. ▶

OGM-Treiber individuell festlegen und optimieren

Ab Version 2.x von OGM stehen einem Programmierer verschiedene Treiber für eine Anwendung zur Verfügung: BOLT, HTTP oder Embedded Driver.

Die Auswahl des gewünschten Treibers erfolgt entweder über eine Konfigurationsdatei oder direkt im Java-Programm über ein Configuration-Objekt.

Die Voreinstellung *ogm.properties* in der Konfigurationsdatei überschreibt die Definition einer gegebenenfalls eigenständigen Systemeigenschaft oder Umgebungsvariablen, zum Beispiel: *ogm.properties = treiberAuswahl.*

Jeder der drei Treiber kennt weitere spezielle (teilweise obligatorische) Einstellungen, um ihn zu spezifizieren und die Graphdatenbank anzusprechen. So unterscheidet ein Embedded-Treiber den Betrieb eines Graphen als persistente oder impermanente Datenbank; im letzteren Fall verflüchtigen sich die Daten des Graphen nach Beenden der Anwendung.

Listing 11: Rahmenprogramm für eine Transaktion

```
Transaction tx = session.beginTransaction();
Abteilung abteilung = session.load(Abteilung.class,
abteilId);
Mitarbeiter mitarbeiter = session.load(Mitarbeiter.
class, mitarbeiterId);

// Beginn der Transaktionsverarbeitung
try {
  bearbeiteAbteilung(abteilung, mitarbeiter);
  verarbeiteMitarbeiter(mitarbeiter, abteilung);
  tx.commit();
}
catch (FehlerException e) {
  tx.rollback();
}
tx.close();
// Ende der Transaktionsverarbeitung
```

Wie bei allen DBS stellt erst eine geeignete Programmlogik die Konsistenz der Datenbank auf Anwendungsebene sicher. Unterstützt das DBMS die Implementierung von Datenbank-Triggern, so stellen diese ein mächtiges Hilfsmittel für die Wahrung der Datenkonsistenz dar. Mit Neo4j entwickelte Datenbanken kennen zwar noch keine Datenbank-Trigger, allerdings bieten sich Persistent-Events in Kombination mit den seit Version 3 verfügbaren Stored Procedures als Workaround an – zumal auch Datenbanktechniker Trigger häufig als speziellen Typ einer Stored Procedure auffassen. Realisiert man Stored Procedures zur Gewährleistung der Datenkonsistenz, so genügt es, diese in Abhängigkeit von den Persistent-Events auszuführen.

Beim Typ des Top-Level-Objekts handelt es sich um alle Objekte, die Neo4j OGM direkt als eigenständige Entity *(@NodeEntity)* bereitstellt – dazu gehören auch Relationship Entities *(@RelationshipEntity)*. Im einfachsten Fall einer Hierarchie oder eines Baums entspricht ein Top-Level-Objekt dem Wurzelelement. Aus Sicht der Graphentheorie erhält man einen Subgraphen der Graphdatenbank – dessen erstes erreichbares Element das Top-Level-Objekt darstellt. Für alle über den Subgraphen verbundenen und von den Datenbank-Operationen betroffenen Elemente löst das OGM-Framework Events aus.

Zu beachten gilt, dass Neo4j OGM analog einem Datenbank-Trigger keine Events für Leseoperationen auf der Datenbank auslöst. Innerhalb einer eventgetriebenen Programmierung muss sichergestellt sein, dass es zu keiner großen Event-Kaskade kommt, um die Gefahr möglicher Programmfehler auszuschließen. Zudem sollte aus Performancegründen eine Überflutung der Event-Verarbeitung ausgeschlossen werden. Dies verhindert nur eine Minimierung der erzeugten Events.

Minimierung der Kommunikation im Serverbetrieb

Das OGM-Framework minimiert bereits die erforderliche Kommunikation, um im Client-Server-Betrieb von Neo4j eine bestmögliche Performance für eine Anwendung zu erreichen.

Für diese Optimierung implementierte Neo Technology im OGM-Framework weitere Parameter, die Entwickler an ihre eigenen Bedürfnisse anpassen können.

Wählt man den Depth-Parameter entsprechend den jeweils in der Graphdatenbank betroffenen Ausschnitten, so minimieren sich die aus dem Graphen angesprochenen Elemente auf die tatsächlich benötigte Anzahl.

Ferner realisiert Neo4j OGM sogenanntes Smart Object Mapping, um nicht benötigte Anfragen an die Datenbank zu eliminieren. Damit schränkt man die seitens des DBMS zu behandelnden Elemente auf ein Minimum ein.

Smart Object Mapping erfolgt über Session-Objekte, deren Laufzeitverhalten ein Programmierer über Vorgaben steuern kann.

Das OGM-Framework gewährleistet die damit verbundenen Anforderungen nach Endlichkeit ausgelöster Ereignisse und ihrer Minimierung durch mehrere Vorkehrungen:

- Nur die beiden Methoden *save()* und *delete()* eines Session-Objekts lösen Persistenz-Ereignisse aus.
- Nur für eine beschränkte Auswahl der vom Top-Objekt erreichbaren Elemente des Subgraphen der Graphdatenbank löst Neo4j OGM das zugehörige Ereignis/den Event aus.
- Diese einmalige Auslösung eines Events für ein bestimmtes Element des betroffenen Subgraphen stellt selbst im Fall eines Zyklus im Graphen Endlichkeit sicher.
- Um die Anzahl der Events zu minimieren, löst Neo4 OGM Events nur für diejenigen Elemente des Subgraphen aus, die erzeugt, geändert oder gelöscht wurden.

Der mit dem OGM-Framework verbundene Event-Mechanismus kennt vier verschiedene Event-Typen: *PRE_SAVE, POST_SAVE, PRE_DELETE* und *POST_DELETE*. Die gewählte Bezeichnung drückt bereits selbsterklärend aus, wann der Event ausgelöst und welche Bedeutung mit ihm verbunden ist. So löst OGM beispielsweise den Event-Typ *PRE_DELETE* vor dem Löschen eines Elements in der Graphdatenbank aus. Die vier verschiedenen Typen der Persistent-Events realisiert das Event-Interface der Klasse *PersistenceEvent* aus dem OGM-Package *org.neo4j.ogm.session.event*. Sobald eine Anwendung einen Persistent-Event verarbeiten will, erzeugt OGM eine Instanz dieser Klasse, die man anwendungsseitig ansprechen kann.

Das zweite vom OGM-Framework neu eingeführte Java-Interface stellt die Klasse *EventListener* dar. Sie deklariert für jeden Event-Typ entsprechende Methoden: *onPreSave(), onPostSave(), onPreDelete()* und *onPostDelete()*. Innerhalb einer Anwendung muss vorher nicht notwendigerweise der Event-Typ ermittelt und entsprechend fallweise dessen Verarbeitung angestoßen werden. Vielmehr reicht es aus, diesen Methoden einfach den jeweiligen Event-Typ zu übergeben.

Diese beschriebene Programmiermethodik minimiert den Quellcode und erleichtert eine spätere Wartung der Anwendungen. Das Session-Objekt verfügt über die Methode *register()*, um für die Event-Verarbeitung die erforderlichen Handler einzurichten. Diese Vorgehensweise erfordert eine Dekla-

Realisierung von Microservices mit Neo4j

Mit Persistent-Events des OGM-Frameworks unterstützt Neo4j verteilte Anwendungen auf der Basis von SOA (Service-Oriented Architecture).

Dienste sollten klein sein, dabei weitgehend entkoppelt, das heißt möglichst unabhängig, und eine spezifizierte Aufgabe erledigen. Derartige Dienste nennt man Microservices; sie kommunizieren miteinander, um ein bestimmtes gemeinsam verfolgtes Ziel zu erreichen.

Vermutlich eignen sich Microservices besonders gut für die Realisierung zukünftiger Anwendungen auf SOA-Basis.

Listing 12: Rahmenprogramm für einen Microservice

```
class PreSaveEventListener extends
EventListenerAdaper {
  @Override
  void onPreSave(Event event) {
    // Bestimmen des betroffenen Entity
    DomainEntity entity = (DomainEntity)
    event.getObject();
    // Zugehoerige Verarbeitung der Entity
    ...
  }
}
```

ration der Event-Verarbeitung an einer zentralen Stelle. Sollte eine Anwendung vielseitige und komplexe Anforderungen besitzen, die zur Wahrung der Datenkonsistenz zu berücksichtigen sind, so führt dies zu Nachteilen.

Eine zentrale Event-Verarbeitung vergrößert die Verantwortung des betroffenen Objekts, was allen gängigen Architekturen von Anwendungen für eine parallele und verteilte Verarbeitung widerspricht. Heutige Anwendungen für das Internet, das mobile Web oder die Cloud sind verteilt, verarbeiten eine große Anzahl von Daten und verlangen vor allem nach Parallelität, um ihre Performance zu verbessern; sie benötigen nach wie vor aber auch Datenkonsistenz. Demnach verlangt die Praxis verstärkt nach Architekturen, die Ausfallsicherheit gewährleisten. Neo Technology deckt diese Anforderungen im OGM-Framework über eine abstrakte Klasse *EventListenerAdapter* ab. Dabei handelt es sich um eine Implementierung des EventListener-Interface ohne Methoden. Definiert man eine Erweiterung von EventListenerAdapter und deklariert man dort die Methoden des EventListener-Interface, so kann man die Verantwortlichkeiten über verschiedene kleinere Objekte verteilen (Listing 12).

Mehrere kleinere Objekte mit zentralen Verantwortlichkeiten unterstützen besser als ein großes zentrales Objekt die Verteilung und Parallelisierung innerhalb einer Anwendungslandschaft. Sie erhöhen die Ausfallsicherheit und verbessern dadurch die Verfügbarkeit des Gesamtsystems. Zudem erlaubt diese Programmiertechnik eine zukünftige Realisierung von Microservices, um die Komplexität in verteilten Welten insgesamt zu reduzieren. ■

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

Links zum Thema

- Homepage von Neo4j
  http://neo4j.com
- Homepage der Neo4j-Dokumentation
  http://neo4j.com/docs
- Neo4j Java Developer Reference
  https://neo4j.com/docs/pdf/neo4j-java-reference-3.0.pdf
- Neo4j Object Graph Mapping OGM Library
  https://neo4j.com/docs/pdf/neo4j-ogm-manual-2.0.pdf
- Neo4j Driver API für C#.NET, Java, JavaScript und Python
  http://neo4j.com/docs/developer-manual/current/drivers
- Neo4j Developer Manual für Java
  https://neo4j.com/docs/pdf/neo4j-developer-manual-3.0-java.pdf
- Homepage von openCypher
  www.opencypher.org
- Homepage von JCypher
  http://jcypher.iot-solutions.net
- GitHub-Home des JCypher-Projekts
  https://github.com/Wolfgang-Schuetzelhofer/jcypher/wiki
- The Search Engine für The Central Repository
  https://search.maven.org
- Homepage der Spring Tool Suite
  https://spring.io/tools
- Homepage von Eclipse
  https://eclipse.org
- Homepage von IntelliJ IDEA
  https://www.jetbrains.com/idea
- Homepage der NetBeans IDE
  https://netbeans.org
- Overview (Neo4j 3.0.4 API)
  https://neo4j.com/docs/java-reference/current/javadocs/overview-summary.html
- Homepage von Oracle Java SE Embedded
  www.oracle.com/technetwork/java/embedded/embedded-se
- Beispielprogramme für Embedded Neo4j on the JVM
  https://github.com/neo4j/neo4j/tree/3.0/manual/embedded-examples
- Homepage der GraphAware Neo4j UUID-Library
  https://github.com/graphaware/neo4j-uuid

NAVIGATIONSLÖSUNGEN FÜR RESPONSIVE SEITEN

Das passende Menü

Die richtige Navigation ist entscheidend bei responsiven Webseiten.

Eine gut durchdachte Navigation sorgt dafür, dass die einzelnen Inhalte und Unterseiten schnell gefunden werden. Aufgrund der Vielzahl an Seiten, die erreichbar sein sollen, ergeben sich bei Webseiten oft relativ komplexe Menüs. Bei Desktop-Versionen behilft man sich dann mit verschachtelten Navigationen oder nutzt mehrere Navigationen wie Haupt- und Nebennavigation.

Ein solches Menü in derselben Form in der Smartphone-Variante anzeigen zu lassen ist keine gute Idee – dafür fehlt dort schlicht der Platz. Also sind Anpassungen erforderlich. Entscheidend ist dabei, dass die Navigation auf einem Smartphone nicht den gesamten sichtbaren Platz in Beschlag nehmen sollte.

Diesen Fehler, mit der Navigation den ganzen Viewport zu belegen, hat man in der Anfangszeit des responsiven Webdesigns (RWD) häufiger gemacht. Das ist in zweierlei Hinsicht ungünstig: Zuerst einmal ist dadurch der zentrale Inhalt nicht direkt sichtbar. Der zweite Nachteil zeigt sich bei der Interaktion: Nach einem Klick auf einen Menübutton ist schwer zu erkennen, wann die neue Seite geladen ist, weil ja auch auf der neuen Seite im Viewport zuerst einmal nur die Navigation zu sehen ist.

Eine weitere Anforderung an die Navigation ist, dass die Buttons nicht zu klein sein dürfen. Das gilt allgemein für Links, Buttons und andere Schaltflächen auf Touchscreens. Für Navigationen ist die Größe der Buttons jedoch besonders zentral, weil hier mehrere Buttons direkt nebeneinander stehen. Damit ist die Gefahr größer, bei einer Berührung den falschen Button zu erwischen und damit eine andere Seite als die gewünschte aufzurufen.

Umsetzungsmöglichkeiten für Navigationen

Es gibt verschiedene Umsetzungsmöglichkeiten für Navigationen auf kleinen Screens, die im Folgenden anhand von Beispielen beschrieben werden. Interessant ist hier auch und gerade, für welche Navigationslösung sich bekannte Webseiten entscheiden, da diese einen prägenden Einfluss auf das Verhalten und die Gewohnheiten der Besucher haben.

Große Seiten setzen allerdings nicht zwangsläufig auf responsives Webdesign für die Anpassung an verschiedene Geräte. Stattdessen erstellen manche unterschiedliche Versionen, die aufgrund des vom Browser gesendeten User-Agents ausgeliefert werden. Solche separaten mobilen Versionen bieten oft aufschlussreiche Lösungen und werden ebenfalls im Nachfolgenden behandelt. Während Sie die Anpassungen für kleine Screens beim responsiven Webdesign durch Veränderung der Größe des Browserfensters sehen können, funktioniert das mit separaten Smartphone-Varianten nicht. Hilfreich sind hier die Entwickler-Tools von Chrome mit der Auswahl *Toggle device toolbar*. Hier können Sie zwischen verschiedenen Geräten wählen und zum Beispiel ein iPhone simulieren. Danach müssen Sie die Seite noch aktualisieren, damit der User-Agent neu gesendet werden kann und wirklich die Smartphone-Variante – sofern vorhanden – angezeigt wird. Eine ähnliche Funktion gibt es inzwischen auch bei Firefox. Wenn Sie in den Entwickler-Tools *Bildschirmgrößen testen* wählen, erhalten Sie in der neuen Ansicht rechts neben den oberen Buttons auch ein Feld *Benutzerdefinierter User-Agent*, in das Sie einen eigenen User-Agent-String eingeben können.

Sichtbare Navigationen

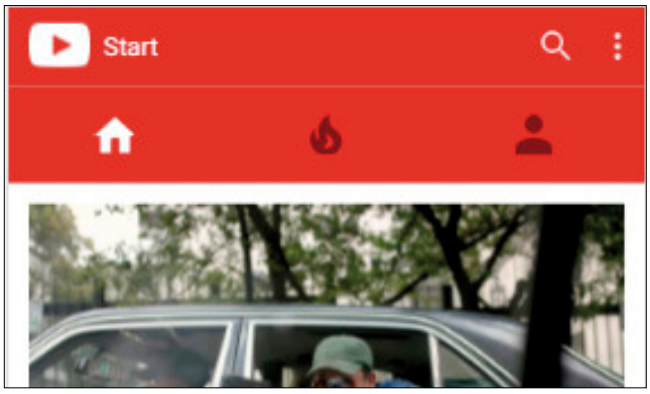
Einfache Navigationen lassen sich auch bei der Smartphone-Variante als horizontale Leiste oben platzieren. Damit sind alle Menüpunkte direkt sichtbar und ohne zusätzliche Zwischenklicks aufrufbar. Untersuchungen der Nielsen Norman Group zeigen, dass die Usability von sichtbaren Navigationen besser ist als die von versteckten. Das Problem an nicht sichtbaren oder hinter einem Button versteckten Navigationen ist eben auch, dass dem Benutzer die Begriffe nicht direkt ins Auge springen wie bei einer sichtbaren Navigation und dass er damit nie weiß, ob sich der Klick lohnt.

Im einfachsten Fall funktioniert diese horizontale Darstellung des Menüs sowohl auf Desktops als auch auf kleinen Screens, sodass keine Anpassungen an der Navigation durchgeführt werden müssen. Ein Beispiel hierfür ist die Website von Ethan Marcotte (Bild 1): Die vier Navigationspunkte sind bei den unterschiedlichen Varianten immer nebeneinander platziert.

Gleichzeitig wird am Beispiel der Marcotte-Webseite eben auch die prinzipielle Beschränktheit dieses Ansatzes klar: Es ist keine universelle Lösung, die Sie bei beliebig vielen Menüpunkten einsetzen können – mehr als drei oder vielleicht vier Punkte dürfen es nicht sein.

Eine Alternative besteht darin, die Elemente mehrreihig anzuordnen – etwa bei sechs Menüpunkten zwei Reihen mit drei Punkten. Allerdings geht hier schnell viel wertvoller Platz verloren, der eigentlich dem Inhalt gehören sollte, und deshalb dürfen es auch nicht zu viele Punkte werden. Platzsparender als Texte können Icons sein, wenn ihre Bedeutung klar ist – was dann wiederum an native Apps erinnert (Bild 2).

Zwei Einsatzbereiche gibt es für diese horizontale Navigation: Sie bietet sich einerseits bei sehr einfachen Webseiten mit wenig verschiedenen Seiten an. Andererseits kann diese Navigation auch zusätzlich zu einer anderen, beispielsweise aufklappenden Navigation eingesetzt werden – wie es bei der mobilen Version der YouTube-Seite praktiziert wird. Die Hauptpunkte sind direkt sichtbar, die andere Navigationspunkte verbergen sich hinter dem Icon mit drei Punkten. Diese Variante wird häufig auch Priority+-Navigation genannt, weil die wichtigsten Punkte besonders behandelt werden. Die Herausforderung ist dabei natürlich, die richtigen Menüpunkte auszuwählen, die direkt sichtbar sein sollen.



Navigation bei YouTube: oben eine Reihe mit Icons – weitere Einstellungen verbergen sich bei den drei Punkten (Bild 2)

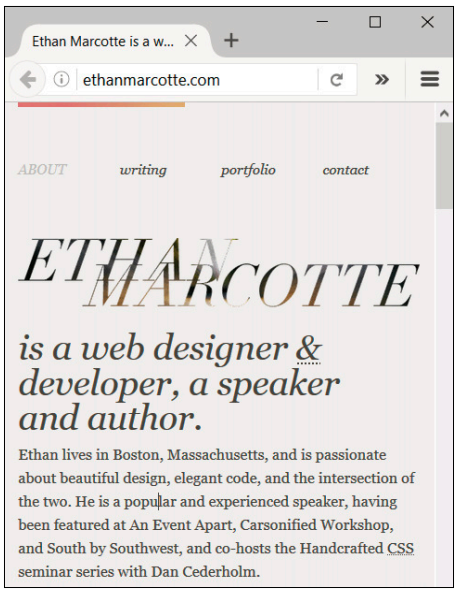
Sprungmenü – Ankernavigation

Ein anderes Beispiel für eine sichtbare Navigation ist die Ankernavigation oder das Sprungmenü. Hier befindet sich die Navigation nicht oben wie beim gerade vorgestellten Navigationstyp, sondern im Fußbereich der Webseite.

Darum kann sie auch mehrere Punkte umfassen und mehr Platz einnehmen. Erreichbar ist die Navigation durch einen Button im oberen Bereich: Wenn der Besucher auf diesen klickt, gelangt er mit einem seiteninternen Link nach unten zur Navigation – daher der Name Ankernavigation.

Diese Lösung hat mehrere Vorteile:

- Dieser Navigationstyp ist sehr einfach umzusetzen – es wird nur CSS benötigt.
- Im Fußbereich darf die Navigation auch ausführlicher sein, als wenn sie oben angezeigt ist.
- Für die Anordnung von Navigationen am Ende der Seite spricht auch die Überlegung, dass der Besucher, nachdem er sich durch den Inhalt der Seite gelesen hat, Vorschläge für neue Seiten bekommt.



Vier Navigationspunkte reichen Ethan Marcotte – und damit sind keine Anpassungen für kleinere Screens notwendig (Bild 1)

Das Problem ist jedoch, dass solche Navigationen im Allgemeinen nicht als intuitiv empfunden werden und dass das Nach-unten-Springen bei Klick auf den Menübutton eher für Irritationen sorgt, wenn nicht sogar für Desorientierung. Ein bisschen abmildern lässt sich das Problem, indem man den Scrollvorgang nach unten animiert – damit erhält der Benutzer die Chance, mitzubekommen, wohin er gelangt ist. Ein Beispiel dafür zeigt die Webseite http://iso.org.

Ein weiterer Nachteil ist, dass diese Lösung nicht als sonderlich schick gilt, und so hat sie sich nicht durchgesetzt.

Abgesehen davon ist es allerdings durchaus empfehlenswert, eine zusätzliche Navigation im Fußbereich unterzubringen – als Unterstützung der Hauptnavigation und als Anreiz für den Besucher, weitere Inhalte aufzurufen, wenn er am Ende einer Seite angekommen ist.

Select-Menü

Bisher wurden nur sichtbare Navigationen vorgestellt – kommen wir nun zu den bei Bedarf aufklappenden Menüs. Ein Beispiel dafür ist die Auswahlliste oder das Select-Menü. Bei diesem Ansatz wird die bei viel verfügbarem Platz klassisch dargestellte Navigation bei weniger Platz zu einer Auswahlliste. Platzmangel lässt sich so leicht umgehen.

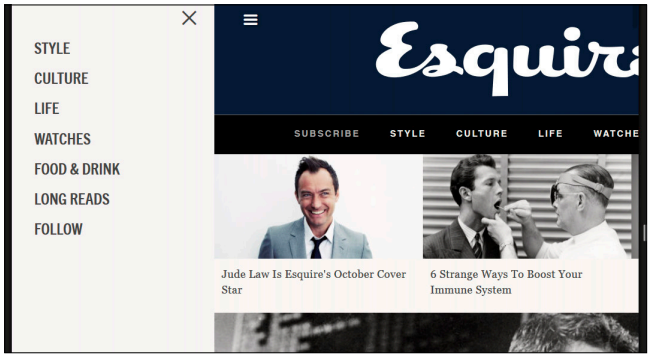
Aber die Nachteile überwiegen: Erst einmal passt eine Auswahlliste nicht zum restlichen Design der Webseite, da Smartphones üblicherweise eigene Lösungen für die Darstellung von Auswahllisten haben. Ein weiterer großer Nachteil ist, dass Auswahllisten ihre Berechtigung im Kontext von Formularen haben – bei Navigationen wirken sie eher bevormundend. Die Kombination der Nachteile ist sicher der ▶

Grund dafür, dass Select-Menüs für die Navigation heute kaum noch eingesetzt werden. Das Smashing Magazine hatte ursprünglich auf diese Navigationslösung gesetzt, die Auswahlliste ist jedoch inzwischen längst einer Klappnavigation gewichen.

Damit verlassen wir auch schon die Navigationslösungen, die sich rein über CSS umsetzen lassen – die nun vorgestellten Ansätze benötigen immer JavaScript.

Ausklappmenü

Das Ausklappmenü (Toggle) ist eine beliebte Menüvariante. Das Grundprinzip ist folgendes: Die Navigation ist beim ersten Aufruf der Seite nicht zu sehen, der Benutzer kann sie aber jederzeit durch Klick auf einen Button (Icon oder Text) aufrufen. Dadurch klappt die Navigation wie beim Akkordeon-Prinzip aus und schiebt die Inhalte weiter nach unten. Das Ausklappen erfolgt üblicherweise animiert.



Off-Canvas: Bei Klick auf den Menübutton schiebt sich das Menü von links herein und verdrängt den Inhalt ein Stück nach rechts (Bild 3)

Das Praktische an dieser Variante ist, dass die Anzahl der Menüpunkte nicht von vornherein beschränkt ist und trotzdem das Menü im eingeklappten Zustand so gut wie keinen Platz einnimmt.

Off-Canvas

Ähnlich wie das Ausklappmenü funktioniert auch die Off-Canvas-Variante. Hier ist die Navigation ebenfalls zuerst nicht sichtbar, erscheint aber bei Klick auf einen Button. Die Navigation hat dabei allerdings eine andere Position als beim Ausklappmenü: Sie befindet sich im Normalzustand außerhalb des Bildschirms (Off-Canvas), also beispielsweise links außerhalb des Browserfensters. Bei Klick auf den Button schiebt sich die Navigation von der linken Seite herein und verschiebt damit den eigentlichen Inhalt nach rechts.

Das Interessante bei Off-Canvas ist, dass – zumindest programmiertechnisch – der Raum außerhalb des Viewports genutzt wird, was auch bedeuten kann, dass der nach rechts geschobene Inhalt abgeschnitten wird.

Ein Beispiel für eine Off-Canvas-Navigation sieht man etwa bei www.esquire.co.uk (Bild 3). Hier wird die Off-Canvas-Navigation sogar bei großen Screens verwendet: Die Navigation wird durch Klick auf das Hamburger-Icon aufgerufen und schiebt sich von links herein. Bei kleinen Screens verdeckt jedoch die Navigation vollständig den Bildschirm, so dass wir es mit dem heute immer beliebter werdenden Pattern Overlay zu tun haben.

Eine weitere, aber etwas anders gelagerte Form von Off-Canvas zeigt Google bei seiner Suche: Hier kann der Benutzer zwischen verschiedenen Kategorien wie News, Bilder et cetera wählen. Diese werden unterhalb des Suchfelds horizontal angeordnet. Wenn es mehr Kategorien sind, als auf dem aktuellen Viewport Platz finden, so ist die Leiste mit den Kategorien teilweise abgeschnitten und rechts außerhalb des Screens platziert. Der Benutzer erreicht die weiteren Kategorien, indem er die Leiste nach links wischt beziehungsweise zieht (Bild 4).

Bei der Google-Suche ist nur ein Teil der möglichen Kategorien sichtbar – für weitere wie etwa die Rubrik *Videos* muss man den Bereich nach links scrollen (Bild 4)

Overlay

Das Overlay-Pattern ist ein Navigationsmuster, das zunehmend beliebter wird. Ein Overlay wird beispielsweise bei Webundmobile.de eingesetzt: Bei Klick auf das Hamburger-Icon legt sich die Navigation mit dunklem Hintergrund über den Bildschirm (Bild 5). Es gibt verschiedene Varianten des Overlay-Patterns – die Navigation kann den vollständigen Viewport verdecken (Fullscreen), oder aber es können noch weiterhin Teile des Inhalts zu sehen sein.

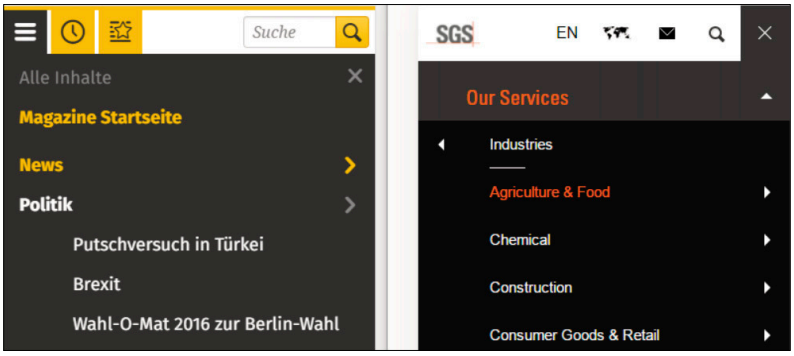
Das Overlay hat wie die Klapp- oder Off-Canvas-Navigation den Vorteil, dass die Navigation keinen kostbaren Platz einnimmt und nur bei Bedarf erscheint. Außerdem sieht diese Variante einfach gut aus – was mit der Grund sein dürfte, dass sie inzwischen auch schon häufiger für große Screens verwendet wird.

Komplexe Navigationen

Was macht man aber bei komplexen Navigationen? Also bei Navigationen mit Unterpunkten oder sogar mehreren Unterebenen? Bevor man sich Lösungen überlegt, lohnt es sich, abzuwägen, ob wirklich diese Komplexität notwendig ist und ob es nicht prinzipiell Möglichkeiten der Vereinfachung gibt – nicht nur für die Smartphone-Variante, sondern überhaupt. Sinnvoll ist es auch, sich anhand der Website-Statistiken einen Überblick zu verschaffen, was tatsächlich von den Benutzern besucht wird, und das in Relation zu den gewünschten Zielen zu setzen.

Die radikalste Lösung im Umgang mit Unterebenen bei der Navigation für kleine Screens besteht darin, sie wegzulassen – so geht etwa Spiegel.de bei seinem Auftritt vor. Die Seiten der Unterthemen erreicht man dann nicht direkt über das Menü, sondern nur über die jeweiligen Hauptseiten.

In den meisten Fällen wird man die Unterpunkte im Menü aufnehmen wollen, und auch dann gibt es unterschiedliche Möglichkeiten der Handhabung. Bei Web.de etwa sind alle Untermenüpunkte standardmäßig zu sehen. Das hat den Vorteil der Sichtbarkeit – der Nachteil ist natürlich, dass der Besucher dadurch nicht direkt alle Hauptthemen im Blick hat, sondern sich durch die Navigation scrollen muss. Diese Lösung funktioniert außerdem nicht bei beliebig vielen Verschachtelungsebenen. Häufig wird mit ausklappbaren Navigationspunkten gearbeitet. Diese funktionieren dann wie ein Akkordeon – Unterpunkte klappen nach unten auf – oder es kann auch ein Einklappen von rechts (Off-Canvas) eingesetzt werden. Eine Frage stellt sich dabei: Was passiert bei einem Klick auf einen Hauptmenüpunkt – soll dann die entsprechende Seite aufgerufen werden oder die Unternavigation ausklappen?



Komplexe Navigationen: Links Web.de mit standardmäßig ausgeklappten Unterpunkten, rechts SGS.com mit Split-Buttons: Der Rechtspfeil kennzeichnet unterschiedliche Aktionen (Bild 6)

Wenn beide Handlungen erwünscht sind, braucht man sogenannte Split-Buttons, die zwei Aktionen auslösen. Solche Split-Buttons finden sich etwa bei der Navigation auf Sueddeutsche.de. Die Buttons beinhalten den Namen der Rubrik wie beispielsweise *Politik* und zusätzlich einen Pfeil nach unten. Ein Klick auf das Wort *Politik* führt zur Startseite des Ressorts, ein Klick auf den Pfeil nach unten öffnet hingegen weitere Unterpunkte.

Ähnlich sieht die Lösung bei SGS.com aus: Nur wird hier die Subnavigation durch einen Klick auf einen Rechtspfeil aufgerufen, weil sich die Unterpunkte von der rechten Seite hereinschieben. Irritierend ist aber für den Benutzer, dass bei SGS.com der Rechtspfeil Unterkategorien andeutet, hingegen an anderen Stellen wie bei Web.de direkt die Seite aufruft (Bild 6).

Hamburger-Icon

Wenn das Menü standardmäßig ausgeblendet ist, benötigt man ein Icon oder einen Text, der zum Aufruf des Menüs dient. Dafür wird häufig das sogenannte Hamburger-Icon eingesetzt, das aus drei horizontalen Strichen besteht (Bild 7).

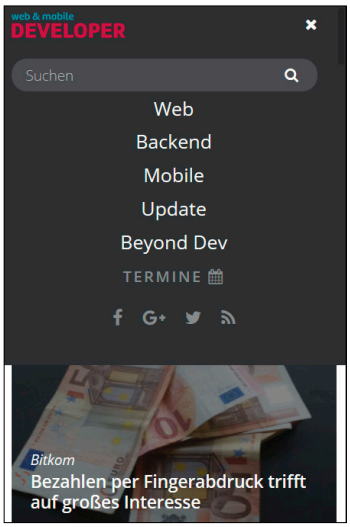
Je nach Zielpublikum und Kontext erfüllt das Hamburger-Icon seinen Zweck sehr gut bis schlecht. Zwei Extrembeispiele:

- Erster Fall: Das Hamburger-Icon wird auf der Smartphone-Variante einer Webseite für junges Publikum eingesetzt. In diesem Fall wird es gut funktionieren.
- Zweiter Fall: Dasselbe Icon wird auf einer Desktop-Seite benutzt, die sich an Senioren richtet. Hier ist damit zu rechnen, dass die auf diese Art versteckte Navigation nur selten genutzt wird.

Das Hamburger-Icon funktioniert auf Webseiten für Smartphones im Allgemeinen gut, weil es inzwischen auf solchen Webseiten sehr häufig benutzt wird und die Surfer die Funktion meist erlernt haben. Das ist auf dem Desktop nicht der Fall – daher wird es bei der Desktop-Varianten prinzipiell weniger gut erkannt. Erschwerend kommt hinzu, dass das Icon bei viel verfügbarem Platz auf einem Desktop in der Fülle der Informationen leichter übersehen wird als bei der Darstellung auf kleinen Screens.

Soll die Menüfunktionalität des Hamburger-Icons besser erkannt werden, bieten sich dafür verschiedene Möglichkeiten an. Hilfreich kann es sein, es durch Umrandung oder Ähnlich eindeutig als Button zu kennzeichnen.

Denkbar wäre auch, zusätzlich einen Text zu ergänzen oder sogar nur einen Text zu verwenden. Der erste Begriff, der einem hier in den Sinn kommt, ist *Menü*. Nach Möglichkeit sollte man aber einen spezifischeren Begriff wählen. Bei Zeitungen spricht man häufig von verschiedenen Rubriken: So findet sich beim Spiegel das Hamburger-Icon sogar auch auf der Variante für große Screens – aber, was hier sicher sinnvoll ist, durch den Begriff *Rubriken* ergänzt.



Bei Webundmobile.de wird ein Overlay für die Navigation eingesetzt (Bild 5)

Zurück zum Inhalt

Nicht immer wählt ein Benutzer aus einer Navigation einen Punkt aus – manchmal möchte er sich nur informieren, welche Seiten über die Navigation erreicht werden können, und entscheidet sich dann doch dafür, auf der aktuellen Seite zu bleiben. Das Zurückkehren zum Inhalt sollte so einfach wie möglich sein.

Wenn der Inhalt neben der Navigation zu sehen ist, sollte ein Klick darauf ebenfalls die Navigation schließen. Bei den Klapp/Off-Canvas/Overlay-Varianten kann zum Schließen dasselbe Icon dienen, das zum Aufrufen der Navigation benutzt wird. Sinnvoll ist es sicher auch, zum Schließen das x-Icon einzusetzen, das auch sonst zum Schließen von Fenstern benutzt wird.

Normalerweise ist das Menü nie die einzige Möglichkeit, auf einer Webseite zu navigieren – es gibt eine Reihe von Alter- ▶

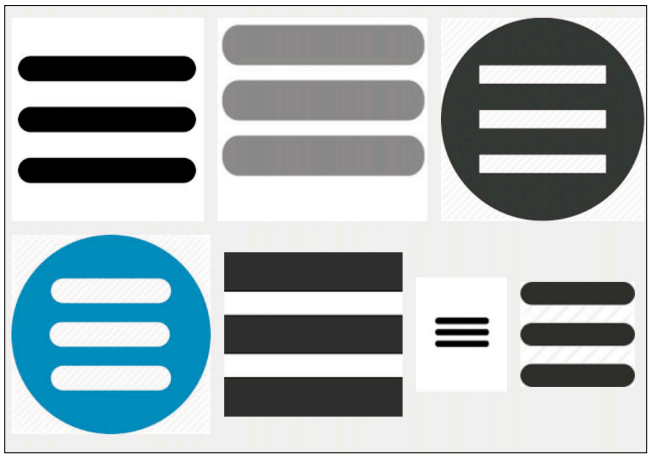
nativen und Ergänzungen. Sehr praktisch ist die Site-interne Suche, die am besten auf kleinen Screens möglichst weit oben und gut erreichbar platziert wird. Aber da die Eingabe auf Touchscreens mühsam ist, sollte die Suche nur eine Ergänzung, nicht die alleinige Navigationsmöglichkeit sein.

Weitere Navigationsmöglichkeiten bieten anklickbare Breadcrumbs, die gerade bei verschachtelten Strukturen für zusätzliche Orientierung sorgen. Außerdem sind interne Verlinkungen sinnvoll, und es gibt Navigationsmöglichkeiten über Tagwolken und Kategorien.

Diese alle können und sollten zur Unterstützung der Navigation benutzt werden.

Mobile als Vorbild für Desktop?

Das Hamburger-Icon zum Aufrufen eines Overlay-Menüs ist ein beliebtes Navigationspattern auf kleinen Screens. Dass dieser Ansatz auf Smartphones erfolgreich ist, heißt allerdings nicht automatisch, dass es eine gute Idee ist, diese Variante auch bei viel verfügbarem Platz, sprich in der Desktop-Variante, zu wählen.



Das Hamburger-Icon zur Kennzeichnung der Navigation (Bild 7)

Natürlich sind Overlay-Effekte schick und es macht Spaß, sie zu bedienen. Aber sie haben den einen entscheidenden Nachteil: Der Benutzer sieht die Navigationspunkte nicht direkt. Um eine neue Seite zu öffnen, sind damit mindestens zwei Klicks erforderlich – einer zum Öffnen der Navigation, der zweite zum Klick auf den Menüpunkt. Außerdem liefern sichtbare Navigationen auch die wichtige Information darüber, worum es bei diesem Webauftritt geht. Die Sichtbarkeit der Navigation ist der große Vorteil der klassischen Menüs, und diesen sollte man nicht ohne guten Grund aufgeben – gerade wenn man wie im Fall der Desktop-Variante genügend Platz und damit die Wahl hat.

Aber es gibt natürlich auch hier Ausnahmen: Im Artikel ging es bisher um klassische Webseiten, bei denen das Menü eine wichtige Rolle spielt und wo es erwünscht ist, dass die Besucher das Menü nutzen, um Seiten aufzurufen. Es gibt jedoch auch andere Fälle, in denen die wesentliche Interaktionsmöglichkeit im Klicken eines Links besteht und die Navigation demgegenüber unwichtig ist. Ein Beispiel wäre ein Call-To-Action (CTA) bei einer Landing-Page mit großem Button für die Hauptaktion, die ausgelöst werden soll. In diesem Fall kann bewusst auch bei der Desktop-Version eine erst bei Bedarf einklappende Navigation gewählt werden – weil eben die Navigation hier nicht die Hauptinteraktionsmöglichkeit darstellt. Ein ähnlicher Fall wäre, wenn über eine Webseite eine Geschichte erzählt wird, die der Benutzer in vorgegebener Weise durchklicken soll. Hier kann eine hinter einem Menübutton versteckte Navigation die richtige Entscheidung sein.

Links zum Thema

- Design einer komplexen Navigation
  https://www.smashingmagazine.com/2016/09/redesigning-sgs-seven-level-navigation-system-a-case-study
- Usability-Probleme bei versteckten Menüpunkten
  https://www.nngroup.com/articles/hamburger-menus
- Klappnavigationen auf dem Desktop
  https://www.sitepoint.com/are-users-ready-for-the-desktop-hamburger-icon

Fazit

Responsive Webdesign ist keine brandneue Technik mehr, was den Vorteil hat, dass es inzwischen eine ganze Palette an Navigationslösungen gibt. In den Anfangszeiten stellte sich die Frage: Wie mache ich das überhaupt mit der Navigation? Heute lautet die Frage eher: Wann wähle ich am besten welche Lösung? Einige Lösungen, die noch in der Anfangszeit praktiziert wurden, sind heute nicht mehr üblich, wie beispielsweise eine Select-Liste als Navigation zu verwenden oder einfach eine horizontale Navigation auf kleinen Screens untereinander darzustellen. Durchgesetzt haben sich hingegen die verschiedenen Formen von Klappnavigationen – und hier ist in letzter Zeit besonders das Overlay angesagt.

Wo es aber geht, sind sichtbare Navigationspunkte gegenüber versteckten vorzuziehen, deswegen sollte man bei genügend Platz eine versteckte Navigation nur aus guten Gründen einsetzen. Im Zweifelsfall helfen Tests, die das spezielle Zielpublikum mit ihrem besonderen Wissen und ihren Erwartungen berücksichtigen. Andernfalls läuft man Gefahr, die Benutzer zu überschätzen – oder auch zu unterschätzen. ■

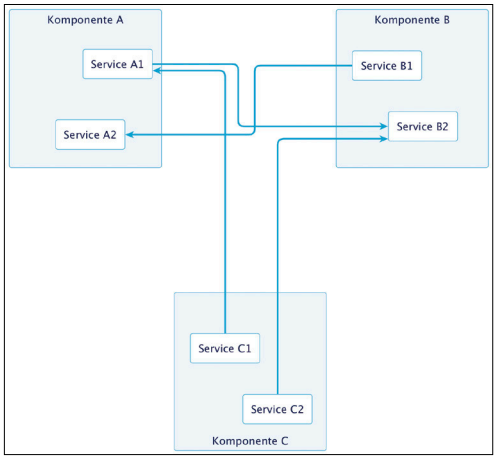
Dr. Florence Maurice
ist Autorin, Trainerin und Programmiererin in München. Sie schreibt Bücher zu PHP und CSS3 und gibt Trainings per Video. Außerdem bloggt sie zu Webthemen unter:
http://maurice-web.de/blog

MESSAGING IN JAVASCRIPT

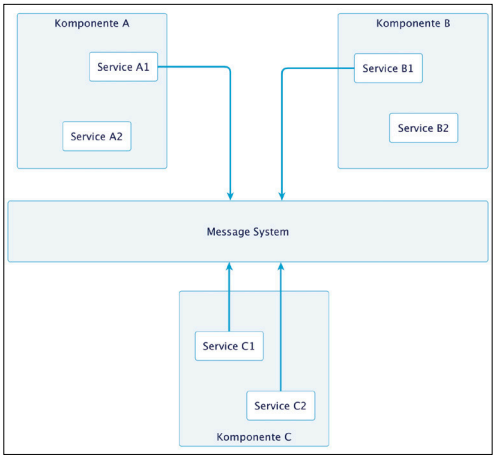
Komponenten entkoppeln

Messaging-Systeme helfen dabei, verschiedene Komponenten innerhalb eines Software-Systems voneinander zu entkoppeln.

Bei der Implementierung komplexer Software-Systeme, bei denen verschiedene Komponenten beziehungsweise Anwendungen miteinander interagieren müssen, können Messaging-Systeme dabei helfen, eine Kopplung zwischen den einzelnen Komponenten zu vermeiden. Statt dass diese direkt miteinander kommunizieren (Bild 1), findet die gesamte Kommunikation über das Messaging-System statt (Bild 2). Einzelne Komponenten schicken dazu Nachrichten an das Messaging-System oder rufen Nachrichten von dort ab. Messaging-Systeme übernehmen dabei verschiedene Aufgaben: Zum einen können eingehende Nachrichten nach bestimmten Regeln verteilt werden, sodass diese von anderen Komponenten abgerufen werden können (Routing), zum anderen sorgen Messaging-Systeme dafür, Nachrichten zwischen verschiedenen Messaging-Protokollen zu übersetzen.



Stark gekoppelte Anwendungen (Bild 1)



Lose gekoppelte Anwendungen (Bild 2)

RabbitMQ

Auf dem Markt existieren eine Reihe verschiedener Messaging-Systeme, von denen Tabelle 1 eine kurze Auswahl zeigt. Ein relativ bekanntes ist RabbitMQ, (https://www.rabbitmq.com), das in Erlang geschrieben ist und (neben anderen Protokollen) das offene Advanced Message Queuing Protocol (AMQP) implementiert. Die wesentlichen Komponenten bei RabbitMQ sind Producer beziehungsweise Publisher, Consumer beziehungsweise Subscriber, Exchanges, Queues sowie Bindings (Tabelle 2, Bild 3). Außerdem wird zwischen Connections und Channels unterschieden.

Tabelle 1: Auswahl bekannter Messaging-Systeme

Bibliothek	Link
ActiveMQ	http://activemq.apache.org
ActiveMQ Apollo	http://activemq.apache.org/apollo
Apache Kafka	http://kafka.apache.org
Apache Qpid	https://qpid.apache.org
RabbitMQ	https://www.rabbitmq.com
HornetMQ	http://hornetq.jboss.org
ZeroMQ	http://zeromq.org

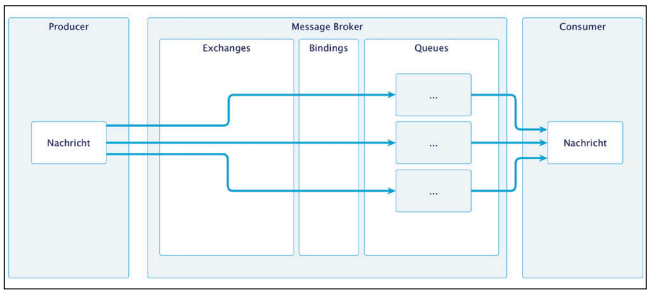
Als Producer werden die Komponenten bezeichnet, die Nachrichten an das Messaging-System schicken. Auf der anderen Seite befinden sich die Subscriber, sprich Komponenten, die die Nachrichten verarbeiten. Producer senden die Nachrichten dabei an sogenannte Exchanges, von denen es verschiedene Typen gibt (dazu gleich mehr) und die dafür verantwortlich sind, Nachrichten nach bestimmten Regeln an Queues zu verteilen (die Queues werden dazu an Exchanges gebunden, Stichwort: Binding).

Subscriber können dann die Nachrichten aus den Queues abrufen. Innerhalb von Queues werden – der Name lässt es bereits vermuten – die Nachrichten nach dem FIFO-Prinzip (First In, First Out) verwaltet: Nachrichten, die zuerst in eine Queue weitergeleitet werden, werden also auch zuerst bearbeitet (Bild 4).

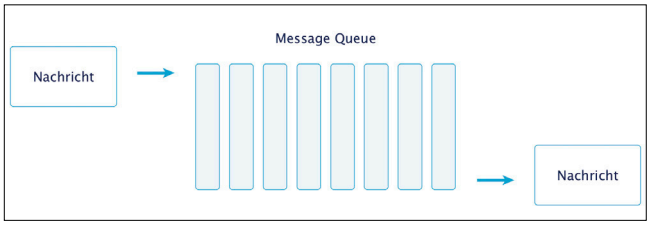
Sowohl Producer als auch Consumer müssen, bevor sie Nachrichten senden beziehungsweise empfangen können, eine (TCP-)Verbindung zu dem Messaging-System herstellen. Um dann auf Exchanges beziehungsweise Queues zugreifen zu können, werden innerhalb einer Verbindung sogenannte Channels – virtuelle Verbindungen innerhalb der TCP-Verbindung – aufgebaut.

Tabelle 2: Wichtige Begriffe im Messaging

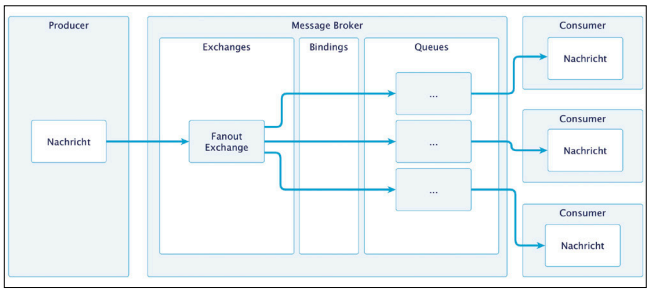
Begriff	Beschreibung
Producer/ Publisher	Komponente, die Nachrichten an das Messaging-System sendet.
Consumer/ Subscriber	Komponente, die Nachrichten aus dem Messaging-System verarbeitet.
Queue	Nachrichtenspeicher, der Nachrichten nach dem FIFO-Prinzip verwaltet.
Message	Daten, die vom Producer zum Consumer über das Messaging-System geschickt werden.
Connection	TCP-Verbindung zwischen einer Anwendung und dem Messaging-System.
Channel	Virtuelle Verbindung innerhalb einer Connection.
Exchange	Nachrichtenverteiler, der Nachrichten von Producern entgegennimmt und nach bestimmten Kriterien an Queues weiterleitet.
Binding	Verbindung zwischen Queue und Exchange.
Routing key	Schlüssel, nach dem ein Exchange entscheidet, an welche Queue(s) eine Nachricht weitergeleitet werden soll.
AMQP	Messaging-Protokoll Advanced Message Queuing Protocol
MQTT	Messaging-Protokoll Messaging Queue Telemetry Transport
STOMP	Messaging-Protokoll Simple Text Oriented Messaging Protocol



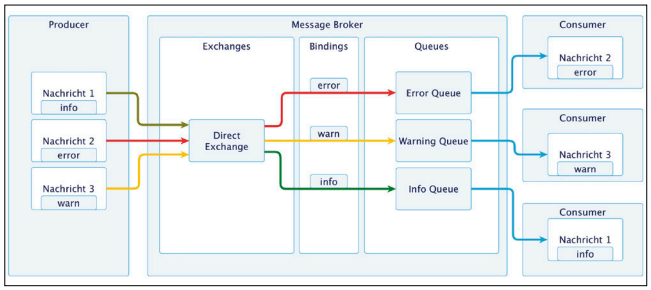
Nachrichtenfluss beim Messaging (Bild 3)



Prinzip einer Message Queue (Bild 4)



Prinzip eines Fanout Exchange (Bild 5)



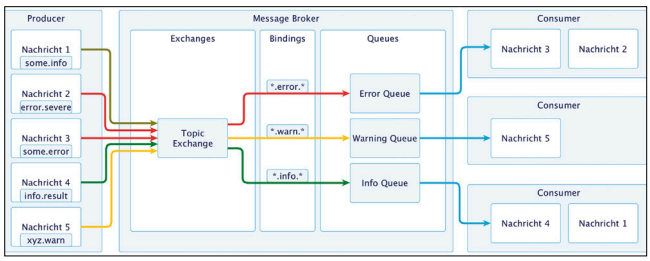
Prinzip eines Direct Exchange (Bild 6)

Insgesamt stehen in RabbitMQ standardmäßig vier verschiedene Typen von Exchanges zur Verfügung: Direct Exchange, Fanout Exchange, Topic Exchange und Headers Exchange. Darüber hinaus lassen sich über eine entsprechende Plug-in-Schnittstelle sogenannte Custom Exchanges implementieren. Hier kann man seinen Routing-Wünschen mehr oder weniger freien Lauf lassen (und beispielsweise geobasiertes Routing implementieren oder Ähnliches). Einige solcher Custom Exchanges findet man etwa unter https://www.rabbitmq.com/community-plugins.html.

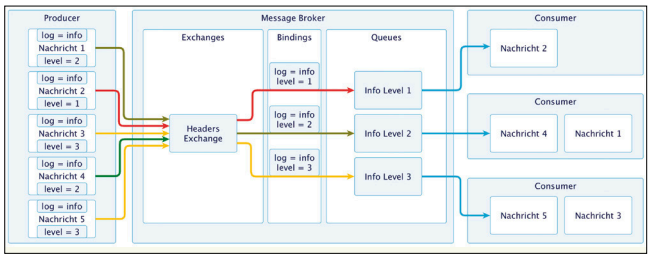
Der einfachste Fall der vier genannten Exchanges ist ein Fanout Exchange: Bei diesem werden eingehende Nachrichten an alle an diesen Exchange gebundenen Queues weitergeleitet (Bild 5). Typischer Anwendungsfall für die Verwendung von Direct Exchanges ist die Umsetzung des Publish-Subscribe-Entwurfsmusters: Eine oder mehrere Komponenten registrieren sich für einen bestimmten Event und werden alle bei Auftreten dieses Events benachrichtigt.

Bei dem Direct Exchange dagegen (Bild 6) wird anhand eines Routing Keys, den der Publisher mit der Nachricht sendet, entschieden, an welche Queue die Nachricht gesendet werden soll. Nur wenn der Routing Key der Nachricht dem Routing Key entspricht, der in dem Binding zwischen einem Exchange und einer Queue definiert ist, wird die Nachricht an die entsprechende Queue weitergeleitet.

Flexibler als Direct Exchanges sind die sogenannten Topic Exchanges (Bild 7). Das Prinzip dabei ist ähnlich wie bei Direct Exchanges, da auch hier anhand des Routing Keys entschieden wird, auf welche Queue eine Nachricht verteilt werden soll. Im Unterschied aber zum Direct Exchange, bei dem der Routing Key exakt mit dem am Binding zwischen Queue und Exchange definierten Key übereinstimmen muss, ist es beim Topic Exchange möglich, Wildcards beziehungsweise Routing Patterns am Binding zu definieren. ▶



Prinzip eines Topic Exchange (Bild 7)



Prinzip eines Header Exchange (Bild 8)

Wem dies nicht reicht, der kann auf die sogenannten Header Exchanges ausweichen, die die flexibelste Variante eines Exchange darstellen (Bild 8).

Anstatt das Routing auf Basis eines einzelnen Routing Keys durchzuführen, geschieht dies auf Basis von Header-Informationen, die mit einer Nachricht mitgesendet werden. Dieser Typ von Exchange eignet sich immer dann, wenn man das Routing anhand mehrerer Parameter vornehmen möchte und diese nicht alle in einem Routing Key unterbringen kann oder möchte.

Installation und erste Schritte

RabbitMQ kann für alle gängigen Betriebssysteme installiert werden, entsprechende Installationsdateien findet man auf der Homepage unter https://www.rabbitmq.com/download.html. Alternativ dazu steht mittlerweile auch ein Docker-Image zur Verfügung (Details siehe unter https://hub.docker.com/r/library/rabbitmq). Client-Bibliotheken für verschiedene Programmiersprachen wie Java, C# oder eben JavaScript finden sich unter https://www.rabbitmq.com/devtools.html.

Für JavaScript beziehungsweise Node.js dürfte das Modul amqplib (https://github.com/squaremo/amqp.node) am bekanntesten sein, über das sich Producer und Consumer implementieren, Exchanges und Queues verwalten, Verbindungen herstellen und Channels aufbauen lassen und vieles andere mehr.

Sein API stellt das Modul in zwei Varianten zur Verfügung: zum einen unter Verwendung von Callback-Funktionen, zum anderen unter Verwendung von Promises.

Installiert wird amqlib wie für Node.js-Module gewohnt mit Hilfe des Node.js Package Managers über den Befehl *npm install amqplib*. Anschließend wird es über *require('amqplib/callback_api')* (bei Verwendung des Callback-API) beziehungsweise *require('amqplib')* (bei Verwendung des Promise-API) eingebunden.

Listing 1: Versenden einer Nachricht an eine Queue

```
'use strict';
const amqp = require('amqplib/callback_api');
const configuration = {
  hostname: '<server_url>',
  username: '<user_name>',
  password: '<password>',
  connectionTimeout: 10000,
  authMechanism: 'AMQPLAIN',
  vhost: '/',
  noDelay: true,
  ssl: {
    enabled: true
  }
};
const queue = 'example-queue';
amqp.connect(configuration, (error, connection) => {
  connection.createChannel((error, channel) => {
    channel.assertQueue(queue);
    channel.sendToQueue(queue, new Buffer
    ('Hello World!'));
    console.log(' [x] Sent Hello World!');
  });
  setTimeout(() => {
    connection.close(); process.exit(0)},
  500);
});
```

Listing 2: Verarbeiten einer Nachricht

```
'use strict';
const amqp = require('amqplib/callback_api');
const configuration = {
  /* ... */
}
const queue = 'example-queue';
amqp.connect(configuration, (error, connection) => {
  connection.createChannel((error, channel) => {
    channel.assertQueue(queue);
    channel.consume(queue, message => {
      if (message !== null) {
        console.log(message.content.toString());
        channel.ack(message);
      }
    });
  });
});
```

Der einfachste Fall einer Kommunikation zwischen Producer und Consumer verzichtet ganz auf Exchanges und läuft direkt über eine Queue: Der Producer schickt seine Nachricht direkt an diese Queue, der Consumer holt diese Nachricht

Listing 3: Publisher mit Promises

```
'use strict';
const amqp = require('amqplib');
const configuration = {
  /* ... */
}
const queue = 'example-queue';
amqp.connect(configuration)
  .then(connection => connection.createChannel())
  .then(channel => {
    return channel.assertQueue(queue).then(ok => {
      return channel.sendToQueue(queue, new
      Buffer('Hello World!'));
    });
  });
```

Listing 4: Consumer mit Promises

```
'use strict';
const amqp = require('amqplib');
const configuration = {
  /* ... */
}
const queue = 'example-queue';
amqp.connect(configuration)
  .then(connection => connection.createChannel())
  .then(channel => {
    return channel.assertQueue(queue).then(ok => {
      return channel.consume(queue, message => {
        if (message !== null) {
          console.log(message.content.toString());
          channel.ack(message);
        }
      });
    });
  });
```

aus der Queue und verarbeitet sie. Den Code dafür zeigen Listing 1 und Listing 2 (Producer und Consumer bei Verwendung des Callback-API) sowie Listing 3 und Listing 4 (Producer und Consumer bei Verwendung des Promise-API).

In allen Fällen wird zunächst über den Aufruf der Methode *connect()* eine Verbindung zum Messaging-System hergestellt und innerhalb dieser (TCP-)Verbindung eine virtuelle Verbindung aufgebaut (über den Aufruf von *createChannel()*). Der Aufruf von *assertQueue()* ist optional: Wurde die Queue bereits vorher erstellt – entweder programmatisch oder aber über die Management-Oberfläche von RabbitMQ (https://www.rabbitmq.com/management.html) –, so wird die Queue nicht erneut erzeugt.

Um eine Nachricht an die Queue zu schicken, wird innerhalb des Codes für den Producer die Methode *sendToQueue()* aufgerufen. Ihr übergibt man den Namen der Queue sowie den Inhalt der Nachricht in Form eines Buffers. Auf Seite des Publishers wird über die Methode *consume()* eine Callback-Funktion an dem Channel registriert, der immer dann aufgerufen wird, wenn eine neue Nachricht für den Consumer in der Queue bereitgestellt wird. Zu beachten ist, dass die Nachrichten, falls mehrere Consumer an einer Queue registriert wurden, nach dem Round-Robin-Verfahren nacheinander an die Consumer verteilt werden.

Listing 5: Fanout Exchange

```
'use strict';
const amqp = require('amqplib');
const configuration = {
  /* ... */
};
const exchange = 'example-fanout-exchange';
const queue1 = 'example-queue';
const queue2 = 'example-queue-2';
amqp.connect(configuration)
  .then(connection => connection.createChannel())
  .then(channel => {
    channel
      .assertExchange(exchange, 'fanout')
      .then(ok => {
        return Promise.all(
          [
            channel.assertQueue(queue1).then(ok =>
            channel.bindQueue(queue1, exchange, '')),
            channel.assertQueue(queue2).then(ok =>
            channel.bindQueue(queue2, exchange, ''))
          ]
        );
      })
      .then(ok => {
        channel.publish(exchange, '', new Buffer
        ('Hello World!'))
      });
  });
```

Verwenden von Exchanges

RabbitMQ unterstützt wie bereits erwähnt standardmäßig vier verschiedene Typen von Exchanges, die definieren, auf welche Art und Weise Nachrichten an die Queues verteilt werden.

Listing 5 zeigt exemplarisch die Verwendung eines Fanout Exchange: Über die Methoden *connect()* und *createChannel()* werden – wie zuvor auch schon – zunächst TCP-Verbindung und virtuelle Verbindung zu dem Messaging-Broker aufgebaut. Anschließend wird über die Methode *assertExchange()* sichergestellt, dass es einen Exchange mit dem angegebenen Namen gibt (in diesem Fall *example-fanout-exchange*). Diese Methode funktioniert analog zu der in den vorigen Listings bereits verwendeten Methode *assertQueue()* ▶

Tabelle 3: Eigenschaften von Nachrichten

Methode	Beschreibung
expiration	Millisekunden, nach denen eine Nachricht aus dem System gelöscht wird.
userId	Wenn eine User-ID angegeben wird, wird diese mit der User-ID verglichen, für die die Verbindung zu dem Message Broker aufgebaut wurde. Wenn die User-IDs nicht übereinstimmen, wird die Nachricht abgelehnt.
CC	Ermöglicht die Angabe zusätzlicher Routing Keys.
priority	Zahlenwert, der die Priorität der Nachricht repräsentiert.
persistent	Boolesche Angabe darüber, ob die Nachricht persistiert werden soll und somit auch einen Neustart des Message Brokers überstehen soll.
deliveryMode	Zahlenwert oder boolesche Angabe darüber, ob die Nachricht persistiert werden soll (*1* beziehungsweise *true*) oder nicht (*2* beziehungsweise *false*). Stattdessen sollte jedoch die Eigenschaft *persistent* verwendet werden.
mandatory	Boolesche Angabe darüber, ob die Nachricht zurückgeschickt werden soll, falls es kein passendes Binding zwischen Exchange und einer Queue gibt und die Nachricht daher nicht an eine Queue weitergeleitet werden kann.
BCC	Wie die Eigenschaft *CC*, allerdings werden die Routing Keys nicht an den Consumer weitergeschickt beziehungsweise sind dann nicht mehr in den Headern enthalten.
immediate	Boolesche Angabe darüber, ob die Nachricht zurückgeschickt werden soll, falls die Nachricht nicht direkt an einen Consumer geschickt werden kann.
contentType	MIME-Typ des Nachrichteninhalts.
contentEncoding	Encoding des Nachrichteninhalts.
headers	Anwendungsspezifische Header.
correlationId	ID, die dazu dient, Anfragenachrichten zu Antwortnachrichten zuzuordnen.
replyTo	Ermöglicht die Angabe einer Queue, zu der eine Antwortnachricht gesendet werden soll.
messageId	Anwendungsspezifische ID der Nachricht.
timestamp	Zeitstempel der Nachricht.
type	Anwendungsspezifischer Typ der Nachricht.
appId	Anwendungsspezifische ID der Anwendung, durch die die Nachricht versendet wurde.
immediate	Boolesche Angabe darüber, ob die Nachricht zurückgeschickt werden soll, falls die Nachricht nicht direkt an einen Consumer geschickt werden kann.
contentType	MIME-Typ des Nachrichteninhalts.

Tabelle 4: Methoden des RabbitMQ-API

Methode	Beschreibung
assertQueue()	Prüft, ob es eine Queue gibt. Für den Fall, dass dem nicht so ist, wird die Queue angelegt.
checkQueue()	Prüft, ob es eine Queue gibt.
deleteQueue()	Löscht eine Queue.
purgeQueue()	Entfernt alle noch nicht verarbeiteten Nachrichten aus einer Queue.
bindQueue()	Bindet eine Queue an einen Exchange.
unbindQueue()	Löst die Bindung einer Queue zu einem Exchange.
assertExchange()	Prüft, ob es einen Exchange gibt. Für den Fall, dass dem nicht so ist, wird der Exchange angelegt.
checkExchange()	Prüft, ob es einen Exchange gibt.
deleteExchange()	Löscht einen Exchange.
bindExchange()	Bindet einen Exchange an einen anderen Exchange.
unbindExchange()	Löst die Bindung eines Exchange zu einem Exchange.
publish()	Sendet eine Nachricht an einen Exchange.
sendToQueue()	Sendet eine Nachricht direkt an eine Queue.
consume()	Registriert einen Consumer an einer Queue.
cancel()	Stoppt das Senden von Nachrichten an einen bestimmten Consumer.
get()	Ruft eine Nachricht von einer Queue ab.
ack()	Bestätigt den Empfang einer Nachricht.
ackAll()	Bestätigt den Empfang aller auf dem Channel ausstehenden Nachrichten.
nack()	Weist eine Nachricht ab.
nackAll()	Weist alle auf dem Channel ausstehenden Nachrichten ab.
reject()	Weist eine Nachricht ab. Äquivalent zu *nack()*, funktioniert aber auch in älteren Versionen von RabbitMQ.
prefetch()	Setzt die Anzahl an über einen Channel abzurufenden Nachrichten.
recover()	Reiht Nachrichten, deren Empfang noch nicht bestätigt wurde, wieder in die Queue zurück.
close()	Schließt einen Channel.

Tabelle 5: Methoden des STOMP-Client-API

Methode	Beschreibung
Stomp.over()	Erzeugt einen STOMP-Client, der sich über WebSockets zu dem Message Broker verbindet.
Stomp.overTCP()	Erzeugt einen STOMP-Client, der sich über eine TCP-Verbindung zu dem Message Broker verbindet.
Stomp.overWS()	Erzeugt einen STOMP-Client, der sich über WebSockets zu dem Message Broker verbindet.
client.connect()	Stellt die Verbindung zu dem Message Broker her.
client.disconnect()	Beendet die Verbindung zu dem Message Broker.
client.send()	Sendet eine Nachricht an den Message Broker.
client.subscribe()	Registriert den Client an einer Queue.
subscription.unsubscribe()	Trennt die Verbindung zu einer Queue.
client.begin()	Startet eine Transaktion.
transaction.commit()	Löst die Transaktion aus.
transaction.abort()	Bricht die Transaktion ab.

und erzeugt einen Exchange nur neu, falls es noch keinen entsprechenden Exchange im System gibt. Weiterhin wird über *assertQueue()* die Existenz zweier Queues sichergestellt *(example-queue* und *example-queue-2)* und diese jeweils über *bindQueue()* an den Exchange gebunden (existieren Exchanges, Queues und die Bindings zwischen beiden bereits, sind die Aufrufe *assertExchange(), assertQueue()* und *bindQueue()* natürlich überflüssig).

Das eigentliche Versenden einer Nachricht an den Exchange geschieht über die Methode *publish()*. Ihr übergibt man als ersten Parameter den Namen des Exchange, an den die Nachricht gesendet werden soll, als zweiten Parameter optional einen Routing Key (im Fall von Fanout Exchanges bleibt dieser leer, da hierbei der Routing Key ohnehin keine Rolle spielt und daher von RabbitMQ ignoriert wird), und schließlich als dritten Parameter den Inhalt der Nachricht in Form eines Buffers.

Optional kann als vierter Parameter ein Konfigurationsobjekt übergeben werden, über das man beispielsweise darauf Einfluss nehmen kann, wie lange eine Nachricht in einer Queue vorgehalten wird, bevor sie – bei Nichtbearbeitung – aus dem System gelöscht wird, oder beispielsweise, ob Nachrichten persistiert werden sollen und somit auch einen Neustart des Message Brokers überdauern (Tabelle 3). Weiterhin lassen sich unter anderem MIME-Typ und Encoding des Nachrichteninhalts, ein anwendungsspezifischer Nachrichtentyp, ein Timestamp sowie Header definieren, die das Routing beim Header Exchange beeinflussen.

Bei Verwendung eines Direct Exchange ist der Code prinzipiell sehr ähnlich zu dem Code aus dem vorigen Listing, die Unterschiede liegen im Detail: Zum einen übergibt man der Methode *assertExchange()* als zweiten Parameter den Wert *direct*, zum anderen übergibt man der Methode *publish()* als zweiten Parameter einen Routing Key, anhand dessen das Routing vollzogen wird (Tabelle 4).

Möchte man aus einem Browser heraus auf RabbitMQ zugreifen, eignet sich das Protokoll STOMP (Streaming Text Oriented Message Protocol), ein textbasiertes, HTTP-ähnliches Protokoll, das für den Einsatz bei Messaging-Systemen konzipiert wurde und über ein entsprechendes Plug-in (https://www.rabbitmq.com/stomp.html) auch von RabbitMQ unterstützt wird.

Bibliotheken wie stomp-websocket (https://github.com/jmesnil/stomp-websocket) oder das aktuellere webstomp-client (https://github.com/JSteunou/webstomp-client) stellen Clients zur Verfügung, um STOMP über WebSockets (STOMP over WebSockets) zu nutzen. Dazu muss allerdings auf Server-Seite (das heißt, unter RabbitMQ) neben dem STOMP-Plug-in zusätzlich das Web-STOMP-Plug-in installiert werden (https://www.rabbitmq.com/web-stomp.html).

Nachdem man die Bibliothek im HTML-Code eingebunden hat, steht das globale Objekt *webstomp* zur Verfügung. Über die Methode *over()* lässt sich ein STOMP-Client erzeugen, wobei als Parameter eine WebSocket-Objektinstanz zu übergeben ist (alternativ dazu lässt sich über die Methode *overTCP()* ein STOMP-Client auch auf Basis einer TCP-Verbindung erstellen).

Um sich zu dem Message Broker zu verbinden, ruft man anschließend die Methode *connect()* auf, wobei hier der Nutzername, Passwort, ein Callback-Handler für die erfolgreiche Verbindungsherstellung, ein Callback-Handler für den Fehlerfall und optional ein virtueller Host anzugeben sind. Über die Methode *subscribe()* wiederum erfolgt dann die Registrierung an einer Queue, wobei man hier den Namen der Queue und einen entsprechenden Callback-Handler übergibt. Eine Übersicht der zur Verfügung stehenden Methoden des STOMP-Client-API zeigt Tabelle 5.

Fazit

Messaging-Systeme wie das in diesem Artikel vorgestellte RabbitMQ können zur Entkopplung von Anwendungen beitragen. Über entsprechende Client-Bibliotheken ist auch die Integration von Node.js-Anwendungen oder Webanwendungen ohne größeren Aufwand möglich. ■

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

Bild: shutterstock / Wright Studio

DAS SELECTION API

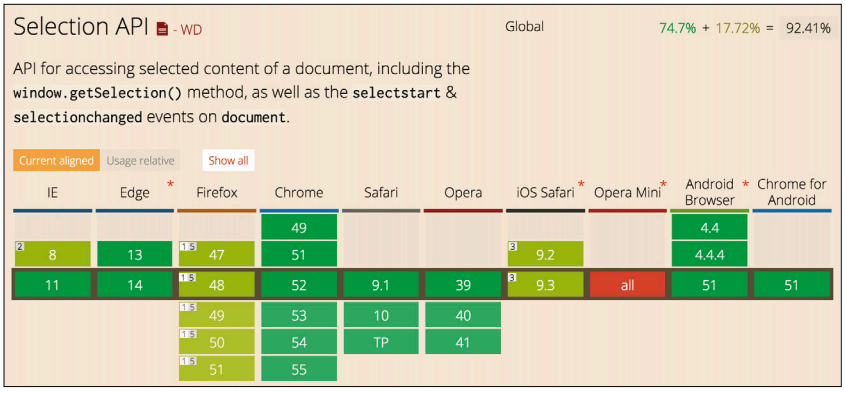
Textinhalte auswählen

Über das Selection API ist es möglich, Textinhalte auf einer Webseite per JavaScript auszuwählen und zu bearbeiten.

Das Selection API (https://www.w3.org/TR/selection-api) befindet sich momentan beim W3C im Status Working Draft, wird mittlerweile aber von so gut wie allen modernen Browsern unterstützt (Bild 1). In älteren Browsern, die keinen Support anbieten, kann man dagegen auf Polyfills wie rangy (https://github.com/timdown/rangy) oder selection-polyfill (https://github.com/luwes/selection-polyfill) zurückgreifen.

Das API definiert das Interface *selection*, welches die aktuelle Auswahl auf einer Webseite repräsentiert, und erweitert die Interfaces *document* und *window* jeweils um die Methode *getSelection()*, welche genau ein Objekt vom Typ *Selection* zurückgibt.

Um über Auswahlen, die der Nutzer trifft, informiert zu werden, stellt das Selection API darüber hinaus die beiden Events *selectstart* und *selectionchange* zur Verfügung. Ersterer wird ausgelöst, wenn der Nutzer mit der Auswahl beginnt, Letzterer dann, wenn sich die aktuelle Auswahl ändert. Entsprechende Event Listener können wie in Listing 1 gezeigt über die



http://caniuse.com

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

Methode *addEventListener()* am *document*-Objekt registriert werden.

Der Typ *selection* verfügt über die in Tabelle 1 gezeigten Eigenschaften und stellt die in Tabelle 2 gezeigten Methoden zur Verfügung. Die Eigenschaft *anchorNode* beispielsweise bezeichnet den Knoten, von dem die Auswahl ausgeht, die Eigenschaft *focusNode* den Knoten, bis zu dem sie reicht. Analog stellt *anchorOffset* die Position beziehungsweise den Offset dar, von dem die Auswahl ausgeht, und *focusOffset* die Position beziehungsweise den Offset, bis zu dem sie reicht. Hierbei ist zu beachten, dass der Wert von *focusOffset* nicht zwangsläufig größer als *anchorOffset* ist: Nur wenn der entsprechende Inhalt von links nach rechts selektiert wurde, ist dies der Fall. Wenn der Nutzer einen Text dagegen von rechts nach links selektiert, ist der Wert von *focusOffset* kleiner als der von *anchorOffset*. Gleiches gilt für das Verhältnis von *anchorNode* und *focusNode*. Auch hier spielt die Richtung der Auswahl eine Rolle. ▶

Listing 1: Zugriff auf Selektionsinformationen

```
'use strict';
function handleStartedSelection() {
  console.log('handleStartedSelection()');
}

function handleChangedSelection() {
  console.log('handleChangedSelection()');
  let selection = window.getSelection();
  // Startknoten der Selektion
  console.log(selection.anchorNode);
  // Startindex der Selektion
  console.log(selection.anchorOffset);
  // Endknoten der Selektion
  console.log(selection.focusNode);
  // Endindex der Selektion
  console.log(selection.focusOffset);
  // Gibt an, ob Startindex gleich Endindex
  console.log(selection.isCollapsed);
  // Anzahl der Ranges
  console.log(selection.rangeCount);
  // Typ der Selektion
  console.log(selection.type);
  let range = selection.getRangeAt(0);
  // Gibt an, ob Startindex gleich Endindex
  console.log(range.collapsed);
  // Startknoten der Selektion
  console.log(range.endContainer);
  // Startindex der Selektion
  console.log(range.endOffset);
  // Endknoten der Selektion
  console.log(range.startContainer);
  // Endindex der Selektion
  console.log(range.startIndex);
}

document.addEventListener('selectstart',
handleStartedSelection);
document.addEventListener('selectionchange',
handleChangedSelection);
```

Tabelle 1: Eigenschaften von Selection

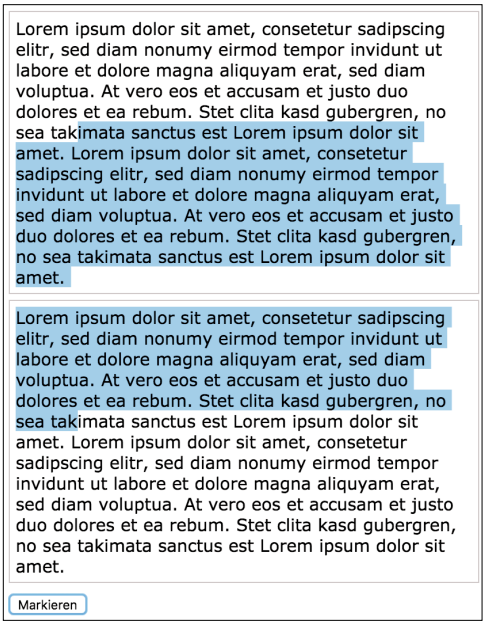
Eigenschaft	Beschreibung
anchorNode	Enthält den Knoten, ab dem die Auswahl beginnt.
anchorOffset	Enthält die Startposition der Auswahl in Form eines Offsets.
focusNode	Enthält den Knoten, an dem die Auswahl endet.
focusOffset	Enthält die Endposition der Selektion in Form eines Offsets.
isCollapsed	Gibt an, ob die Auswahl zusammengeklappt ist, sprich ob Anfangsposition und Endposition gleich sind.
rangeCount	Enthält die Anzahl an Bereichen in einer Auswahl.
type	Enthält den Typ der Auswahl.

Tabelle 2: Übersicht über die Methoden von Selection

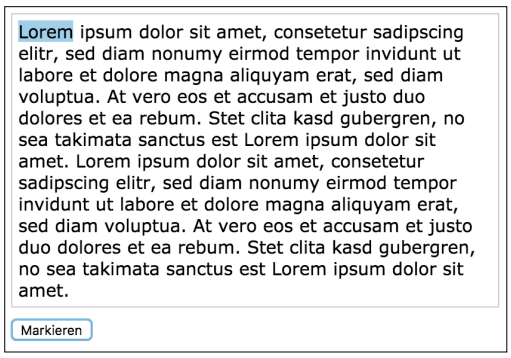
Methode	Beschreibung
addRange()	Fügt einen Bereich zu einer Auswahl hinzu.
collapse()	Deselektiert die Auswahl.
collapseToEnd()	Deselektiert die Auswahl zum Ende des letzten Bereichs.
containsNode()	Prüft, ob der übergebene Knoten in der aktuellen Auswahl enthalten ist.
collapseToStart()	Deselektiert die Auswahl zum Anfang des ersten Bereichs.
deleteFromDocument()	Löscht die Auswahl beziehungsweise deren Inhalt vom Dokument.
empty()	Leert die Auswahl.
extend()	Erweitert die Auswahl.
getRangeAt()	Ermöglicht den indexbasierten Zugriff auf einzelne Bereiche der Auswahl.
removeRange()	Entfernt einen Bereich von der Auswahl.
removeAllRanges()	Entfernt alle Bereiche von der Auswahl.
selectAllChildren()	Fügt alle Kindknoten beziehungsweise deren Inhalte zu der Auswahl hinzu.
setBaseAndExtent ()	Erzeugt eine neuen Bereich und fügt diesen der Auswahl hinzu.
setPosition()	Alias für die Methode *collapse()*.

Um einen Text auf einer Webseite mit Hilfe des Selection API zu selektieren, geht man wie in Listing 2 vor. Zunächst muss über *document.createRange()* ein Objekt vom Typ *Range* erzeugt werden. Dieser Typ repräsentiert einen einzelnen Bereich innerhalb einer Webseite und ist bereits im DOM-Standard definiert (https://dom.spec.whatwg.org/#interface-range). Über die Methoden *setStart()* und *setEnd()* lässt sich definieren, an welcher Stelle im Dokument der Bereich beginnen und an welcher Stelle er enden soll. Dabei wird als erster Parameter der Knoten im DOM-Baum angegeben, als zweiter Parameter der Offset innerhalb dieses Knotens.

Auf diese Weise lassen sich Bereiche innerhalb eines Knotens definieren, aber auch Bereiche, die sich über mehrere DOM-Knoten erstrecken. Der in Listing 2 definierte Bereich bezieht sich beispielsweise auf den Knoten mit der ID *content* und reicht von Position *0* bis Position *5* des darin enthaltenen Textes (Bild 2), der in Listing 3 definierte Bereich dagegen reicht von Position *250* des Knotens mit der ID *content* bis zu Position *250* des Knotens mit der ID *another-content* (Bild 3).



Selektion über mehrere Knoten hinweg (Bild 3)



Selection: So lassen sich Bereiche innerhalb eines Knotens definieren (Bild 2)

Listing 2: Selektion eines Textes per JavaScript

```
let button = document.getElementById('select');
button.addEventListener('click', () => {
  let selection = window.getSelection();
  let content = document.getElementById('content').
  firstChild;
  let range = document.createRange();
  range.setStart(content, 0);
  range.setEnd(content, 5);
  selection.removeAllRanges();
  selection.addRange(range);
});
```

Nach der Definition eines Bereichs kann dieser über den Aufruf der Methode *addRange()* der aktuellen Auswahl hinzugefügt werden. Möchte man den gesamten Inhalt (inklusive Kindknoten) eines Knotens zu der Auswahl hinzufügen, so gelingt dies auch ohne Range-Objekt über die Methode *selectAllChildren()*, die auf dem Selection-Objekt aufzurufen ist.

Neben der Selektion von Inhalten ist es auch möglich, selektierte Inhalte wieder zu deselektieren. Dazu kann man auf die Methoden *collapse(), collapseToStart()* und *collapseToEnd()* zurückgreifen. Die Methode *collapse()* erwartet zwei Parameter, auf deren Basis die Position definiert wird, zu der deselektiert werden soll: Der erste Parameter gibt dabei den Knoten an, der zweite Parameter den Offset. Die Methoden *collapseToStart()* und *collapseToEnd()* kommen dagegen ohne Parameter aus und deselektieren den Inhalt zum Anfang beziehungsweise zum Ende hin. Intern werden in allen Fällen Range-Objekte erzeugt, deren Start- und Endposition sowie Start- und Endknoten jeweils gleich sind. Im Fall von *collapse()* sind dies die übergebenen Parameter, im Fall von *collapseToStart()* der vorige Startknoten und der vorige Start-Offset der Selektion und im Fall von *collapseToEnd()* der vorigen Endknoten und der vorige End-Offset der Selektion.

Mehrfachselektion

Einige Browser wie Firefox erlauben die Auswahl mehrerer Bereiche auf einer Webseite (unter Mac OS X beispielsweise, indem man die Befehl-Taste während der Auswahl gedrückt hält). In solchen Browsern ist dies auch über das Selection API möglich, und zwar, indem man einfach mehrere Bereiche definiert und über *addRange()* der Auswahl hinzufügt.

In Listing 4 beispielsweise ist gezeigt, wie sich auf diese Weise alle Vorkommen des Wortes *Lorem* im Text selektieren lassen. Zugegeben ein nicht wirklich praxisnahes Beispiel, aber mit etwas Fantasie lässt sich ein etwas praxisrelevanterer Anwendungsfall finden: Beispielsweise könnte man, ähnlich wie im Listing gezeigt, mit Hilfe eines regulären Ausdrucks alle auf einer Webseite enthaltenen Kontaktdaten (E-Mail-Adressen, Telefonnummern et cetera) suchen und anschließend selektieren, sodass Nutzer direkt über Copy and Paste die gefundenen Daten aus der Webseite herauskopieren können (wobei dies prinzipiell natürlich auch auf anderen Wegen umsetzbar wäre, etwa unter Verwendung des Clipboard API, das ein späterer Artikel vorstellen wird).

Fazit

Das Selection API stellt verschiedene Funktionalitäten für die Auswahl von Inhalten auf einer Webseite zur Verfügung. Zum einen ist es möglich, Inhalte zu selektieren und zu de-

Listing 3: Selektion über mehrere Knoten

```
let button = document.getElementById('select');
button.addEventListener('click', () => {
  let selection = window.getSelection();
  selection.removeAllRanges();
  let content = document.getElementById
  ('content').firstChild;
  let anotherContent = document.getElementById
  ('another-content').firstChild;
  let range = document.createRange();
  range.setStart(content, 250);
  range.setEnd(anotherContent, 250);
  selection.addRange(range);
});
```

Listing 4: Mehrfachselektion per JavaScript

```
var button = document.getElementById('select');
button.addEventListener('click', () => {
  var selection = window.getSelection();
  selection.removeAllRanges();
  var content = document.getElementById
  ('content').firstChild;
  var text = content.textContent;
  var regExp = /Lorem/g;
  var result;
  var positions = [];
  while ((result = regExp.exec(text)) !== null) {
  positions.push(result.index) }
  positions.forEach(position => {
    var range = document.createRange();
    range.setStart(content, position);
    range.setEnd(content, position + 5);
  selection.addRange(range);});
});
```

selektieren, zum anderen wird man mittels entsprechender Events über Änderungen an der aktuellen Auswahl informiert. Prinzipiell wird das API von allen modernen Browsern unterstützt, wobei die gleichzeitige Auswahl mehrerer Bereiche nicht in allen Browsern funktioniert. ■

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

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

IOS: EIGENE EXTENSIONS FÜR NACHRICHTEN-APP

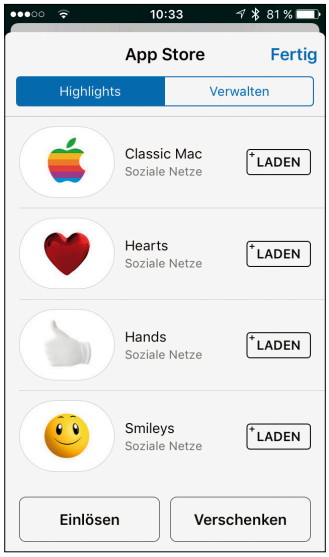
Apps für Nachrichten

Wie sich eigene Extensions für die Nachrichten-App in iOS 10 entwickeln lassen.

Für Apple ist es wohl mit das größte Highlight in der neuen Version 10 seines iOS-Betriebssystems: die stark überarbeitete Nachrichten-App. Sie ist bunter, verspielter und vielseitiger, erlaubt mit sogenannten Stickern neue Interaktionsmöglichkeiten und bietet interessante Features wie versteckte Nachrichten oder das Übertragen von Stimmungen durch aufdringliches Aufploppen oder vorsichtiges Einfliegen von Nachrichten beim Empfänger.

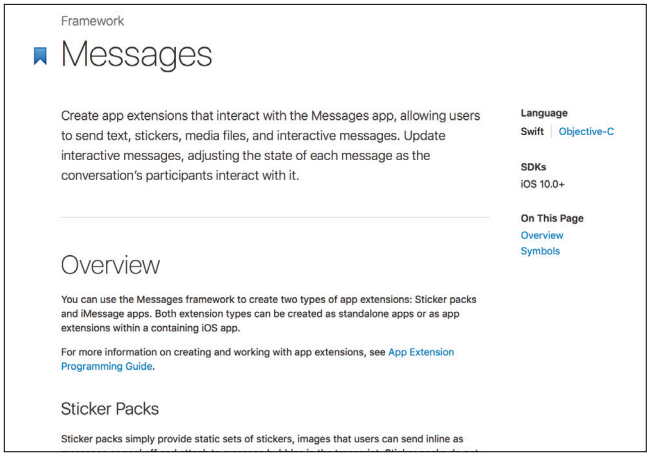
Dass die Nachrichten-App selbst mit iOS 10 deutlich an Funktionen gewonnen hat, ist aber nur das eine Highlight. Für Entwickler gibt es noch ein weitaus größeres: Mit iOS 10 ist es nämlich möglich, eigene Erweiterungen (sogenannte Extensions) für die Nachrichten-App zu entwickeln. Es handelt sich dabei um vollwertige Programme, die entweder anstelle der Bildschirmtastatur oder im Vollbild aus der Nachrichten-App heraus angezeigt werden und damit ganz neue und vielfältige Interaktionsmöglichkeiten erlauben. Erstmals können so eigene Anwendungen zum Versenden von Nachrichten für die Nachrichten-App selbst entwickelt und angeboten werden.

Zu diesem Zweck stellt Apple das neue Messages-Framework bereit, das alle Klassen und Funktionen mitbringt, um eigene Extensions für die Nachrichten-App zu entwickeln (Bild 1). Darüber hinaus gibt es innerhalb der Nachrichten-App in iOS 10 einen eigenen App Store, über den Apps, die eine Erweiterung für die Nachrichten-App enthalten, direkt angeboten werden und von den Nutzern darüber gekauft und heruntergeladen werden können, ohne separat in den App Store wechseln zu müssen (Bild 2). Das geht sogar so weit, dass sich derartige Erweiterungen als Standalone-Apps umsetzen lassen, es also keine zugehörige iOS-App braucht; ein Novum bei der Arbeit mit iOS-Extensions.



Über einen eigenen App Store können Erweiterungen direkt aus der Nachrichten-App heraus heruntergeladen und installiert werden (Bild 2)

Bei den Extensions für die Nachrichten-App wird zunächst einmal zwischen zwei verschiedenen Varianten entschieden: iMessage Apps und Sticker Packs. iMessage Apps stellen die volle Bandbreite des Möglichen dar und sind darauf ausgelegt, eine vollwertige und einzigartige Erweiterung für die Nachrichten-App zu erstellen, um darüber mit anderen Nutzern zu interagieren. Limitierungen liegen hier lediglich im Messages-Framework mit den verfügbaren Funktionen und in der eigenen Fantasie.

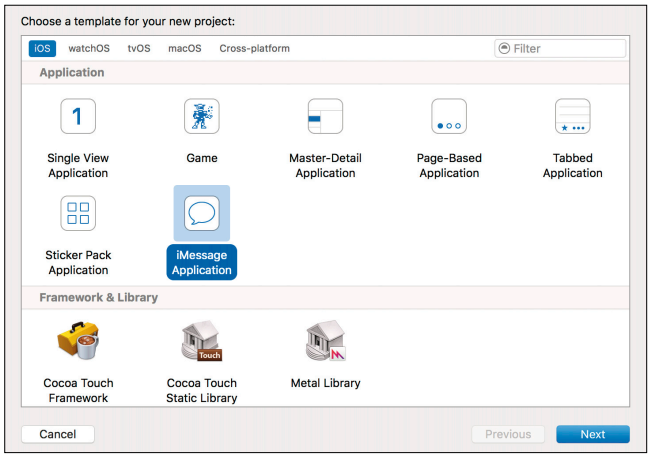


Mit Hilfe des neuen Messages-Frameworks können eigene Extensions für die Nachrichten-App entwickelt werden (Bild 1)

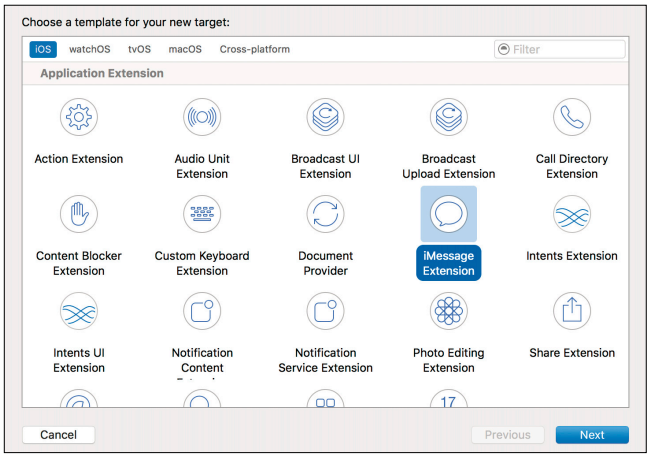
Sticker Packs hingegen sind deutlich einfacher. Wie der Name andeutet, handelt es sich dabei um Erweiterungen der Nachrichten-App, um Grafiken (die sogenannten Sticker) in einer Kommunikation hinzuzufügen. Aufgabe eines Sticker Packs ist es also, ein Set an Grafiken anzubieten, die dann von den Nutzern verwendet werden können. Eine solche Erweiterung lässt sich mit Hilfe des Messages-Frameworks deutlich einfacher umsetzen und benötigt in der Regel weniger eigene Logik als vollwertige iMessage Apps. Es ist sogar möglich, ein Sticker Pack zu erstellen, ohne eine einzige Zeile Code zu schreiben. Sticker Packs sind daher ideal für Erweiterungen, die lediglich ein genanntes Set an Grafiken anbieten sollen und ansonsten über keine tiefergehende Logik und Implementierung verfügen.

iMessage App und iMessage Extension

Apples hauseigene Entwicklungsumgebung Xcode bringt mit Version 8 eine neue Vorlage mit, um direkt ein neues Standalone-Projekt mit einer iMessage Extension zu erstellen (Bild 3). Wählt man diese *iMessage Application* getaufte Vorlage aus und klickt auf *Next*, braucht es im Prinzip nur noch einen gewünschten App-Namen, und das Projekt kann anschließend direkt erstellt werden (Bild 4). Ein solches Projekt setzt sich aus zwei Targets zusammen: einem einfachen App-



Mittels der Xcode-Vorlage *iMessage Application* lässt sich eine neue Erweiterung für die Nachrichten-App erstellen (Bild 3)



Mit der Vorlage iMessage Extension kann ein iOS-Projekt um eine Erweiterung für die Nachrichten-App ergänzt werden (Bild 5)

Target sowie der iMessage Extension. Letztere ist das eigentliche Herzstück einer iMessage App und enthält die komplette Logik sowie alle Dateien, die für die korrekte Funktionsweise der Erweiterung für die Nachrichten-App benötigt werden. Das App-Target hingegen ist im Fall einer iMessage Application sehr rudimentär gehalten und verfügt lediglich über eine *Info.plist*-Datei sowie ein Assets-Bundle, in dem das App-Icon für den App Store Platz findet. Hier wird bereits deutlich, dass eine iMessage Application wie beschrieben gänzlich autark funktionieren kann und keine zugehörige iOS-App benötigt, denn die fehlt in diesem Fall gänzlich.

Trotzdem lässt sich natürlich auch eine bereits bestehende iOS-Anwendung zusätzlich um eine Erweiterung für die Nachrichten-App ergänzen. Dazu muss in einem entsprechenden Projekt ein neues Target hinzugefügt und dort im Bereich *Application Extension* die Vorlage *iMessage Extension* ausgewählt werden (Bild 5). Nach einem anschließenden Klick auf *Next* gilt es noch, alle grundlegenden Informationen wie den Product Name und das zugehörige Projekt festzulegen, und anschließend kann die Extension einem bestehenden Projekt hinzugefügt werden.

Der MSMessagesAppViewController

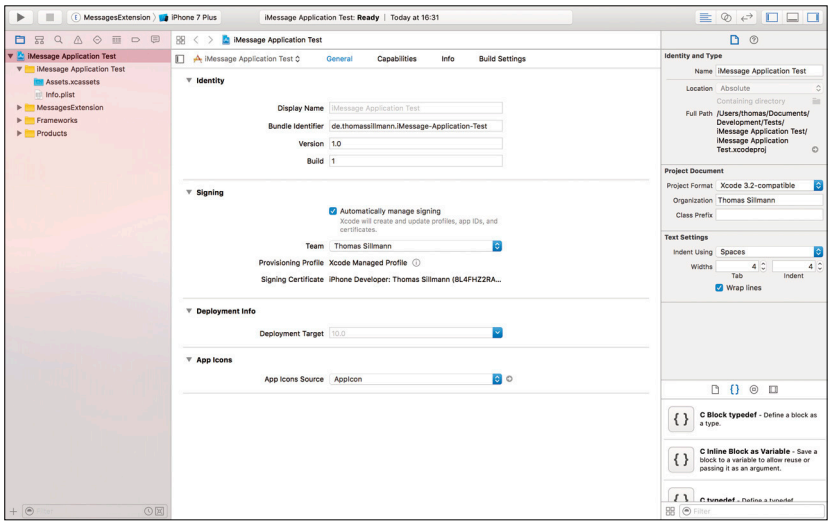
Herzstück einer jeden iMessage App (egal ob nun standalone oder als Teil einer bestehenden iOS-App) ist die Klasse *MSMessagesAppViewController*. Eine Subklasse davon wird automatisch beim Erstellen einer iMessage App erzeugt und hört auf den schlichten Namen *MessagesAppViewController*.

Die Klasse *MSMessagesAppViewController* selbst ist eine von *UIViewController* abgeleitete Klasse. Das, was der Nutzer während der Verwendung einer iMessage App in der Nachrichten-App sieht, ist jene Ansicht eines MSMessagesAppViewControllers. Dieser verfügt über alle grundlegenden und essenziellen Eigenschaften und Methoden, um Nachrichten zu versenden und zu empfangen und auf sonstige Aktionen während einer Konversation zu reagieren.

Eine der wichtigsten und zentralsten Eigenschaften ist dabei die Read-only-Property *activeConversation*. Es handelt sich hier um ein Optional vom Typ *MSConversation*, das einen Zugriff auf die vom Nutzer aktuell ausgewählte Konversation in der Nachrichten-App zur Verfügung stellt, unabhängig davon, ob es sich um einen Chat mit nur einer Person oder um eine ganze Gruppe handelt.

Die Klasse *MSConversation* wird auch dazu verwendet, Nachrichten und Text in das Textfeld der Nachrichten-App einzufügen. Sobald also ein Nutzer Ihrer Extension eine Aktion ausführt, die dazu führen soll, eine Nachricht zum Versenden aufzubereiten, dann übergeben Sie jene Nachricht an das *activeConversation*-Objekt.

Für diese Übergabe stehen Ihnen mehrere verschiedene Methoden zur Verfügung. Welche für Sie die richtige ist, ist davon abhängig, welche Art von Information Sie versenden möchten. ▶



Eine iMessage Application bringt alles mit, um mit der Entwicklung einer Erweiterung für die Nachrichten-App zu beginnen (Bild 4)

Mit Hilfe vier verschiedener Methoden der Klasse *MSConversation* ist es Ihnen möglich, eine Nachricht aufzubereiten, die dann vom Nutzer über die Nachrichten-App versendet werden kann. Die Deklaration dieser Methoden sehen Sie in Listing 1.

Betrachten wir dabei für den Anfang zunächst einmal die einfachste Methode. Mit Hilfe von *insertText(_:completionHandler:)* können Sie in das Textfeld der Nachrichten-App einen beliebigen String einfügen, der als erster Paramter *text* der Methode übergeben wird. Der zweite Parameter *completionHandler* ist optional und bei allen vier gezeigten Methoden in seiner Funktion identisch. Er übergibt Ihnen ein Error-Objekt, wenn es möglicherweise beim Durchführen der gewünschten Aktion zu einem Fehler kommen sollte, und Sie können diesen dann auf Wunsch auslesen und entsprechend reagieren.

Die Methode *insertAttachment(_:withAlternateFilename: completionHandler:)* erlaubt Ihnen das Hinzufügen einer Multimedida-Datei (wie Bilder oder Videos) zum Versand als Nachricht über die Nachrichten-App. Der erste Parameter *URL* erwartet dabei den Pfad zu der gewünschten Datei. Mit Hilfe des zweiten Parameters *filename* können Sie einen alternativen Dateinamen übergeben, der dann im Zusammenspiel mit der zu versendenden Datei angezeigt wird. Wird dieser Parameter nicht gesetzt, parst das System automatisch den Namen aus dem zuvor übergebenen URL. Zu guter Letzt findet sich dann der eben bereits vorgestellte *completionHandler*-Parameter, der im Fall eines möglichen Fehlers ein Error-Objekt zur Auswertung übergibt.

Diese zwei grundlegenden Methoden können somit direkt verwendet werden, ohne weitere Informationen und Eigenschaften aus dem Messages-Framework zu benötigen. Anders sieht es hingegen bei den noch übrigen zwei Methoden aus. Sie hören auf denselben Namen – *insert(_:completionHandler:)* – erwarten als ersten Parameter aber einmal ein Objekt vom Typ *MSSticker* und ein anderes Mal ein Objekt vom Typ *MSMessage*. Bei beiden Typen handelt es sich – wie das Präfix *MS* bereits andeutet – um Klassen aus dem Messages-Framework. Sie dienen ebenfalls dazu, eine Nachricht zu generieren, die anschließend über ein MSConversation-Objekt verschickt werden kann. Beide Typen betrachten wir nun einmal im Detail.

Der Aufbau der Klasse *MSMessage-TemplateLayout* (Bild 6)

MSSticker

Die sogenannten Sticker sind eine Besonderheit der in iOS 10 stark überarbeiteten Nachrichten-App. Sie können innerhalb einer Konversation platziert und wie Sticker an beliebigen Stellen aufgeklebt werden. Aus Systemsicht werden derartige Sticker über die Klasse *MSSticker* abgebildet.

Dabei ist eine solche MSSticker-Instanz sehr simpel aufgebaut und besteht aus zwei Teilen: Da ist einmal ein URL auf eine Bilddatei, der auf das Bild verweist, das als Sticker dargestellt werden soll, und zum anderen ein String, der kurz und knapp einen übersetzten Hinweis zum jeweiligen Sticker liefert. Ein neues Objekt der Klasse *MSSticker* wird so nun mit Hilfe eines passenden Initializers erstellt, deren Deklaration so aussieht:

```
init(contentsOfFileURL fileURL: URL,
localizedDescription: String) throws
```

Dabei kann die Methode fehlschlagen, wie das Schlüsselwort *throws* am Ende der Deklaration deutlich macht. Das ist immer dann der Fall, wenn keine passende Bilddatei über den übergebenen URL gefunden oder geladen werden konnte; entsprechend sollten Sie auf diesen Fall reagieren.

Ist eine neue MSSticker-Instanz mit Hilfe des gezeigten Initializers erstellt, können Sie die gesetzten Informationen über die beiden Read-only-Properties *fileURL* und *localizedDescription* jederzeit auslesen. Ebenso kann eine solche Instanz nun über die Methode *insert(_:completionHandler:)* der Klasse *MSConversation* in einen Nachrichtentext eingefügt werden, womit der Sticker mit dem gesetzten Bild versendet werden kann.

MSMessage

Deutlich komplexer und gleichzeitig mächtiger gestaltet sich die Klasse *MSMessage*. Sie verfügt über alle zugänglichen Eigenschaften, die für Nachrichten in der neuen Nachrichten-App verwendet werden können, und stellt zu diesem Zweck verschiedenste Properties zur Verfügung. Initialisiert wird ein Objekt der Klasse *MSMessage* über den Standard-Initializer *init()*.

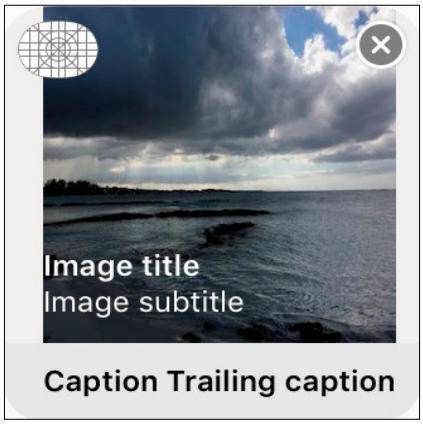
Die wichtigste Property zur Erstellung einer MSMessage-Instanz hört dabei auf den Namen *layout* und ist vom Typ *MSMessageLayout*. Dabei stellt *MSMessageLayout* eine abstrakte Klasse dar, von der selbst keine Instanzen erstellt werden, und die dazu dient, das Aussehen und die anzuzeigenden Elemente einer MSMessage zu definieren. Bisher bietet Apple in seinem Messages-Framework nur eine einzige Sub-

Listing 1: Methoden zum Versenden von Nachrichten

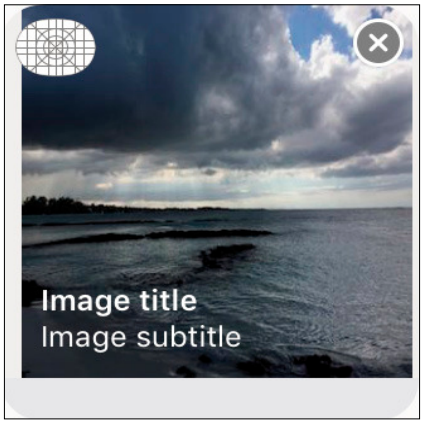
```
func insertAttachment(_ URL: URL,
withAlternateFilename filename: String?,
completionHandler: ((Error?) -> Void)? = nil)
insertAttachment
(_:withAlternateFilename:completionHandler:)
func insert(_ message: MSMessage, completionHandler:
((Error?) -> Void)? = nil)
func insert(_ sticker: MSSticker, completionHandler:
((Error?) -> Void)? = nil)
func insertText(_ text: String, completionHandler:
((Error?) -> Void)? = nil)
```

klasse an, die somit aktuell als Layout für alle MSMessage-Instanzen zu verwenden ist: *MSMessageTemplateLayout*. Diese Klasse definiert eine Nachricht mit einer Multimedia-Datei sowie verschieden platzierten Texten. Den grundlegenden Aufbau dieses Layouts sehen Sie in Bild 6. Im Folgenden liste ich noch einmal alle Bestandteile von oben nach unten hin auf, mitsamt deren Bezeichnung sowie darauffolgend der zugehörigen Property der Klasse, mit der Sie auf jenes Element des Layouts zugreifen und dieses setzen können:

- Bild, Audio oder Video (Property *image* (im Fall eines Bildes) oder *mediaFileURL* (im Fall eines URL-Verweises auf eine Multimedia-Datei),
- Bildtitel (Property *imageTitle),*
- Bilduntertitel (Property *imageSubtitle),*
- Bildunterschrift (Property *caption),*
- Ergänzung zur Bildunterschrift (Property *subcaption),*
- Ergänzende Bildunterschrift (Property *trailingCaption),*
- Ergänzung zur ergänzenden Bildunterschrift (Property *trailingSubcaption).*



Ein mögliches Layout für eine MSMessage-Instanz (Bild 7)



Der angezeigte Bereich eines MSMessage-TemplateLayoutsw passt sich dynamisch anhand der gesetzten und nicht gesetzten Properties an (Bild 8)

Bei allen genannten Properties handelt es sich um Optionals, Sie setzen also nur Werte für all jene Eigenschaften, die Sie auch tatsächlich in Ihrer Nachricht verwenden möchten. Das Layout ist dabei so dynamisch ausgelegt, dass Bereiche, die über keinerlei gesetzte Properties verfügen, gar nicht angezeigt werden. Fehlen beispielsweise jegliche Bildunterschriften im unteren grauen Bereich, so wird dieser Bereich für die Nachricht gar nicht erst angezeigt.

Das MSMessageLayout spielt somit eine essenzielle Rolle bei der Arbeit mit MSMessage und stellt das Herzstück Ihrer eigentlichen Nachricht dar. Neben einer Instanz von *MSMessage* selbst braucht es also auch eine Instanz einer Subklasse von *MSMessageLayout*, von der es – wie beschrieben – aktuell lediglich *MSMessageTemplateLayout* gibt. Ein Objekt dieser Klasse ist dann der Property *layout* von *MSMessage* zuzuweisen, um damit diese beiden Elemente miteinander zu verknüpfen.

Listing 2 zeigt einmal die beispielhafte Erstellung einer Instanz von *MSMessageTemplateLayout*, gefolgt von der Erstellung eines *MSMessage*-Objekts mitsamt Zuweisung des Layouts zur *layout*-Property. In Bild 7 ist zu sehen, wie eine solche Nachricht dann im Anschluss in der Nachrichten-App aussehen kann.

Darüber hinaus ist in Bild 8 zu sehen, wie sich das Layout minimal dynamisch verändert, wenn die Werte der Properties *caption* und *trailingCatpion* des *messageLayout*-Objekts nicht gesetzt werden. Dann nimmt dieser untere Bereich entsprechend weniger Platz weg, ohne dass wir selbst dafür noch ergänzend etwas tun müssten.

Eine so erstellte MSMessage-Instanz kann nun in das Nachrichtenfeld der Nachrichten-App eingefügt werden, indem die Methode *insert (_:completionHandler:)* auf die passende MSConversation-Instanz aufgerufen und ihr das *MSMessage*-Objekt als erster Parameter übergeben wird.

Listing 2: MSMessage-Instanz mit Layout

```
// Create message layout
let messageLayout = MSMessageTemplateLayout()
messageLayout.image = UIImage(named: "MyImage")
messageLayout.imageTitle = "Image title"
messageLayout.imageSubtitle = "Image subtitle"
messageLayout.caption = "Caption"
messageLayout.trailingCaption = "Trailing caption"
// Create message
let message = MSMessage()
message.layout = messageLayout
```

iMessage Extension Status

Mit *MSConversation*, *MSMessage*, *MSMessageLayout* und *MSSticker* wurden die zentralen Klassen zum Erstellen und Versenden von Nachrichten vorgestellt. Das Herzstück einer iMessage App bildet aber noch immer der eingangs bereits vorgestellte *MSMessagesAppViewController*. Die Property *activeConversation*, die auf die aktuelle Konversation verweist und dazu genutzt werden kann, um neue Nachrichten zu erstellen und zu versenden, haben wir bereits kennengelernt. Daneben bringt diese Klasse aber noch einige weitere Eigenschaften mit, um über den aktuellen Status der iMessage Extension selbst zu informieren:

```
func willBecomeActive(with conversation: MSConversation)
func didBecomeActive(with conversation: MSConversation)
func willResignActive(with conversation: MSConversation)
func didResignActive(with conversation: MSConversation)
func dismiss()
```

Die Methoden *willBecomeActive(with:)* und *didBecome Active(with:)* werden nacheinander aufgerufen, sobald die ▶

eigene Extension innerhalb einer Konversation in der Nachrichten-App aktiv wird. Dabei erhalten sie als Parameter das zugehörige *MSConversation*-Objekt für die zugrunde liegende Konversation. Äquivalent dazu verhalten sich die Methoden *willResignActive(with:)* und *didResignActive(with:)*, nur mit dem Unterschied, dass diese nacheinander aufgerufen werden, wenn die eigene Extension wieder inaktiv wird. Die Methode *dismiss()* schließlich wird vor der endgültigen Beendigung der eigenen iMessage Extension aufgerufen.

All diese Methoden können innerhalb der eigenen Subklasse von MSMessagesAppViewController überschrieben und um eigene passende Logik ergänzt werden.

Ebenfalls überschreiben lassen sich weitere Methoden von MSMessagesAppViewController, mit deren Hilfe sich der Status beim Versenden und Empfangen von Nachrichten verfolgen lässt. Die Deklarationen dieser zur Verfügung stehenden Methoden sehen Sie in Listing 3.

Die Methoden *willSelect(_:conversation:)* und *didSelect(_:conversation:)* weisen darauf hin, wenn eine Nachricht innerhalb einer Konversation per Tap ausgewählt wird. Dabei werden sowohl die Nachricht in Form eines *MSMessage*-Objekts als auch die zugehörige Konversation in Gestalt der passenden *MSConversation*-Instanz als Parameter übergeben. Eine wichtige Rolle spielen diese Methoden im Zusammenhang mit Updatable Messages, die gleich noch im Detail vorgestellt werden.

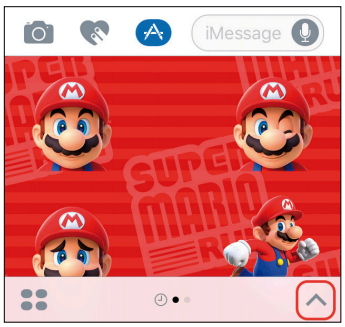
Eingang einer neuen Nachricht

Mit Hilfe von *didReceive(_:conversation:)* werden Sie darüber informiert, dass eine neue Nachricht eingegangen ist. Sobald eine Nachricht über Ihre Extension versendet wird, wird die Methode *didStartSending(_:conversation:)* aufgerufen, um Sie über den Sendevorgang zu informieren.

Das ist insofern wichtig, als dass dieser letzte Schritt – das Senden der mit Hilfe Ihrer iMessage Extension erstellten Nachricht – nicht aus dem Code der Extension heraus angestoßen werden kann. Dieser Schritt muss immer vom Nutzer selbst durchgeführt werden, weshalb Sie keinerlei Einfluss darauf haben und aus dem Code heraus nicht direkt Nachrichten versenden können. Stattdessen informiert Sie diese Methode über einen vom Nutzer gestarteten Sendevorgang einer Nachricht.

Listing 3: MSMessagesAppViewController

```
func willSelect(_ message: MSMessage, conversation:
MSConversation)
func didSelect(_ message: MSMessage, conversation:
MSConversation)
func didReceive(_ message: MSMessage, conversation:
MSConversation)
func didStartSending(_ message: MSMessage,
conversation: MSConversation)
func didCancelSending(_ message: MSMessage,
conversation: MSConversation)
```



Über die Pfeilschaltfläche unten rechts können iMessage Extensions auch im Vollbild angezeigt werden (Bild 9)

Sollte der Nutzer hingegen eine erstellte Nachricht wieder aus dem Nachrichtenfeld löschen, wird in diesem Zuge auch die Methode *didCancelSending(_:conversation:)* aufgerufen und informiert Sie somit über diesen Umstand.

Presentation Style

Standardmäßig wird die Oberfläche einer iMessage Extension am unteren Bildschirmrand der Nachrichten-App unterhalb des zugehörigen Nachrichtenfelds angezeigt und entspricht damit der Größe der Bildschirmtastatur. Mit Hilfe des Storyboards der Extension können Sie diese Oberfläche nach eigenem Belieben gestalten und die zugehörige Logik in Ihrer *MSMessagesAppViewController*-Subklasse unterbringen. Allerdings besitzen iMessage Extensions noch eine zweite Form der Präsentation, nämlich die im Vollbild. Dazu steht automatisch am unteren rechten Bildschirmrand eine Pfeil-Schaltfläche bereit, über die eine Extension den gesamten Bildschirm einnehmen kann (Bild 9).

Dieser Vollbildmodus kann anschließend umgekehrt wieder über eine Pfeilschaltfläche am oberen rechten Bildschirmrand verlassen werden.

Diese beiden Darstellungsarten von iMessage Extensions werden im Messages-Framework über die Enumeration *MSMessagesAppPresentationStyle* definiert. Diese besitzt passend dazu zwei Werte, um zwischen diesen beiden Darstellungen zu unterscheiden:

- *compact*: Die Extension wird klein unterhalb einer Konversation angezeigt.
- *expanded:* Die Extension wird im Vollbild angezeigt und legt sich damit über die Ansicht der Konversation.

Welcher dieser sogenannten Presentation Styles gerade bei Ihrer iMessage Extension aktiv ist, können Sie mit Hilfe der Read-only-Property *presentationStyle* des MSMessagesAppViewControllers in Erfahrung bringen. Entspricht der Wert *compact*, wird die Extension in kleinem Format am unteren Bildschirmrand angezeigt, im Fall von *expanded* nimmt sie hingegen den gesamten Bildschirmplatz ein.

Diese zwei Darstellungsarten einer iMessage Extension müssen im zugehörigen Interface Ihrer Extension berücksichtigt werden, und Sie müssen sicherstellen, dass Ihre Extension sowohl in der Compact- als auch in der Expanded-Ansicht vollständig funktional ist.

Dabei bedeutet das nicht, dass die Expanded-Ansicht einfach nur eine aufgeblähte Version von Compact sein muss; Sie können durchaus zwei grundverschiedene Ansichten in den beiden Fällen anbieten und damit die Funktionalität Ih-

rer Extension womöglich für beide Ansichten optimieren, solange eben nur beide Ansichten tatsächlich ausreichend unterstützt werden.

Um darüber informiert zu werden, dass der Nutzer Ihrer Extension einen Wechsel von Compact zu Expanded und umgekehrt vornimmt, bietet der MSMessagesAppViewController wieder passende Methoden, die Sie in Ihrer eigenen Subklasse überschreiben können, um damit bei einem Wechsel Ihre Ansicht in jeweils geeigneter Weise vom Code aus anpassen zu können. Die Deklaration dieser Methoden sieht so aus:

```
func willTransition(to presentationStyle:
MSMessagesAppPresentationStyle)
func didTransition(to presentationStyle:
MSMessagesAppPresentationStyle)
func requestPresentationStyle(_ presentationStyle:
MSMessagesAppPresentationStyle)
```

Die Methoden *willTransition(to:)* und *didTransition(to:)* werden nacheinander aufgerufen, sobald der Wechsel von einem Presentation Style zum anderen erfolgt. Als Parameter erhalten Sie jeweils den neuen Presentation Style.

Wenn Sie selbst aus Ihrer Extension heraus den Presentation Style anstoßen möchten (zum Beispiel durch eine Schaltfläche in Ihrem Interface, durch die Sie wie der Nutzer über die Pfeilschaltfläche die erweiterte Vollbildansicht aufrufen können), können Sie zu diesem Zweck die ebenfalls deklarierte Methode *requestPresentationStyle(_:)* verwenden. Damit stoßen Sie den Wechsel zu dem als Parameter übergebenen Presentation Style selbst an.

Updatable Messages

Eine Besonderheit von Messages Extensions sind die sogenannten Updatable Messages. Dabei handelt es sich um Nachrichten, die von einem Nutzer begonnen werden und anschließend von anderen Teilnehmern der zugehörigen Konversation bearbeitet (und somit aktualisiert) werden. Zu diesem Zweck kommt eine sogenannte Session zum Einsatz, die Sie als eigenes Objekt konfigurieren und einer MSMessage zuweisen können. Diese Session erlaubt es sodann, eine Nachricht aufgrund ihrer Session-Zugehörigkeit zu aktualisieren und zu verändern.

Zu diesem Zweck verfügt das Messages-Framework über eine Klasse namens *MSSession*. Sie ist direkt von *NSObject* abgeleitet und verfügt selbst über keinerlei zusätzliche Eigenschaften und Methoden. Instanzen dieser Klasse dienen lediglich dazu, eine aktualisierbare MSMessage auszumachen und eindeutig zuzuordnen.

Die Grundlage für eine Updatable Message besteht somit darin, eine Instanz der Klasse *MSSession* zu erstellen und einem neuen Objekt der Klasse *MSMessage* zuzuweisen. Für letztere Aktion bringt MSMessage einen passenden Initializer mit, der als Parameter ein *MSSession*-Objekt erwartet. Das nachfolgende Listing zeigt einmal ein einfaches Beispiel zum Erstellen einer MSSession und eines *MSMessage*-Objekts, dem diese Session zugewiesen wird:

```
let session = MSSession()
let message = MSMessage(session: session)
// Further message configuration ...
```

Wird eine so konfigurierte Nachricht nun über Ihre iMessage Extension versendet, kann beim Empfang der Nachricht die Property *session* des erhaltenen *MSMessage*-Objekts ausgewertet werden, um so festzustellen, ob dieses Teil einer Session ist oder nicht. Möchten Sie dann die erhaltene Nachricht überarbeiten, erstellen Sie lediglich eine neue MSMessage mit Hilfe des Initializers *init(session:)* und weisen diesem eben jenes erhaltene *MSSession*-Objekt zu, damit auch diese neue Message weiterhin über dieselbe Session verfügt. Wird sodann diese neue Message mit der bekannten Session verschickt, dann verschwindet die vorherige Nachricht derselben Session aus der Konversation und es bleibt nur noch die neueste zurück.

Damit sind Updatable Messages in den Fällen sinnvoll, in denen eine bestimmte Nachricht von anderen Teilnehmern einer Konversation bearbeitet und somit aktualisiert werden soll, woraufhin die ursprüngliche Nachricht keine Rolle mehr spielt. Wenn Sie solch einen Anwendungsfall besitzen, kann die Arbeit mit MSSession zum Erreichen dieses Ziels sehr sinnvoll sein.

In diesem Zuge können Sie zudem zwei weitere Properties der Klasse *MSMessage* nutzen, um die Arbeit mit einer Session noch weiter zu verfeinern und zu optimieren. Zum einen ist dies die Property *summaryText*. Dieser String wird unterhalb der versendeten MSMessage angezeigt, wenn diese Teil einer Session ist. Somit kann diese Property im Verlauf der Bearbeitung einer Nachricht dazu genutzt werden, über den aktuellen Fortschritt und die nächsten Schritte der Aktualisierung der Nachricht zu informieren. Des Weiteren können Sie die Property *url* nutzen, um weitere zusätzliche Informationen für eine Nachricht, die Teil einer Session ist, zu übertragen. Dazu bauen Sie ein solches URL-Objekt idealerweise mit Hilfe der Klasse *NSURLComponents* auf und können so – neben der eigentlichen Nachricht – zusätzliche Informationen übertragen, die womöglich für die Aktualisierung und Bearbeitung einer Nachricht wichtig sind.

Apples IceCreamBuilder-Projekt behandelt alle wichtigen Aspekten von iMessage Extensions (Bild 10)

IceCreamBuilder-Projekt

Wenn Sie die Umsetzung einer kompletten iMessage Extension einmal praktisch nachvollziehen möchten, sollten Sie unbedingt einen Blick auf Apples Beispielprojekt na- ▶

mens *IceCreamBuilder* werfen. Es handelt sich dabei um eine iMessage Extension, über die die Nutzer vorgefertige Bilder von verschiedenen Eisvariationen verschicken können oder sogar anhand vorgefertigter Kombinationen eigene Kreationen mit anderen Teilnehmern einer Konversation erzeugen können (Bild 10). Das Projekt behandelt alle wichtigen Aspekte von iMessage Extensions wie das Versenden und Aktualisieren von Nachrichten, den Zugriff auf eine Konversation und das Anpassen der Ansicht beim Wechsel der Presentation Styles.

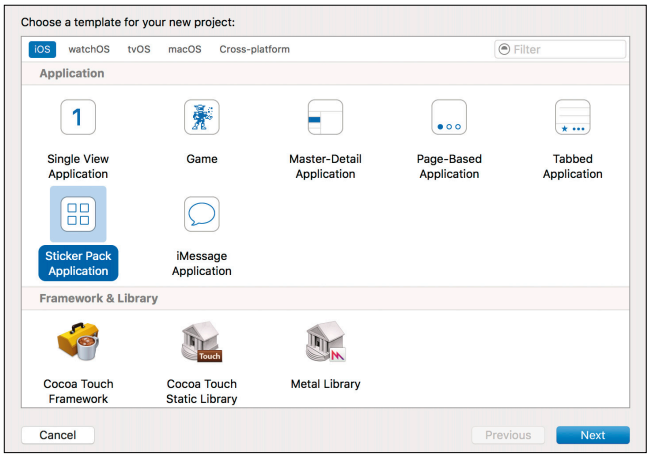
Das Projekt kann kostenfrei von Apples Developer-Website heruntergeladen werden. Zur Ausführung benötigen Sie mindestens Xcode 8.

Sticker Pack Application und Extension

Wie eingangs beschrieben, sind mit dem neuen Messages-Framework sowie der aktualisierten Nachrichten-App in iOS 10 nicht nur iMessage Apps und Extensions möglich, sondern auch sogenannte Sticker Pack Apps und Extensions. Diese Form der Erweiterung für die Nachrichten-App ist einzig und allein für die Verwendung von Stickern gedacht, wie wir sie bereits kurz bei Vorstellung der Klasse *MSSticker* kennengelernt haben. Solche Apps stellen entsprechend Sticker-Grafiken bereit, die die Nutzer dann in ihre Konversationen einbauen können.

Naturgemäß sind damit Sticker Pack Apps und Extensions deutlich weniger komplex als iMessage Apps. Zwar lassen sich auch hier ein eigenes User Interface sowie eigene Code-Logik unterbringen, die Verwendung der Nachrichten-App beschränkt sich am Ende aber eben immer nur auf den Versand von Stickern, sonst nichts.

Dafür gibt es einen immensen Vorteil in der Umsetzung: Für eine reine Sticker Pack App oder Extension müssen Sie nämlich keine einzige Zeile Code schreiben. Das wird spätestens dann deutlich, wenn Sie über Xcode ein neues Projekt erstellen und dort als Vorlage eine Sticker Pack Application auswählen (Bild 11). Das neu erstellte Projekt setzt sich nämlich sodann einzig und allein aus einem Assets-Bundle zusammen, das Sie mit den gewünschten Grafiken für Ihre Sticker Pack App befüllen können (Bild 12).

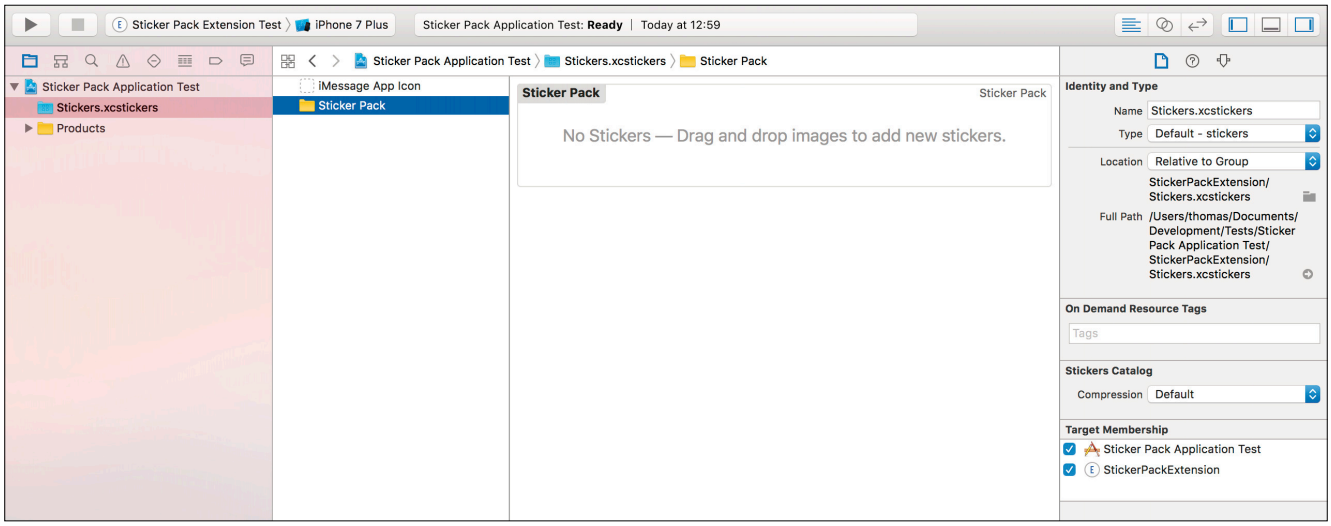


Mit Hilfe der Vorlage *Sticker Pack Application* erstellen Sie eine reine Sticker-Erweiterung für die Nachrichten-App (Bild 11)

Sie brauchen sich also um nichts weiter zu kümmern. Anhand der Grafiken wird automatisch eine passende Sticker Pack App erstellt, in der die Sticker in einer Art Collection-View automatisch angezeigt werden. Dazu ist es auch nicht notwendig, ein eigenes Interface zu erstellen; es reicht aus, das vorgegebene Asset-Bundle passend zu befüllen. Dasselbe gilt auch für eine Sticker Pack Extension, die ebenfalls nur ein solches Asset-Bundle mit sich bringt.

Erweiterung von Sticker Pack Apps

Wenn Sie mehr Konfigurationsspielraum für eine Sticker Pack App oder Extension wünschen, dann besteht der einzige Weg darin, stattdessen eine iMessage App beziehungsweise Extension zu erstellen und auf die beschriebene Art und Weise umzusetzen und damit zu arbeiten. Sie können sich dann lediglich eine weitere Klasse aus dem Messages-



Eine in Xcode neu erstellte Sticker Pack Application verfügt lediglich über ein Assets-Bundle, das mit den gewünschten Stickern zu befüllen ist (Bild 12)

Framework zunutze machen, die es Ihnen erlaubt, die bei einer Sticker Pack App automatisch generierte CollectionView-Ansicht direkt umzusetzen: *MSStickerBrowserViewController*.

Diese Klasse bildet die Standardansicht einer Sticker Pack App innerhalb einer iMessage App nach und stellt somit den idealen Spagat zwischen der Einfachheit einer Sticker Pack App und der Flexibilität einer iMessage App dar. Dabei setzen sich Instanzen der Klasse *MSStickerBrowserViewController* stets aus zwei Bestandteilen zusammen. Einerseits ist da die Größe, in der die Sticker dargestellt werden sollen. Diese Größe wird mittels der Enumeration *MSStickerSize* definiert und verfügt aktuell über drei mögliche Werte:

- *small:* Sticker werden in einer Größe von 100 x 100 Punkten dargestellt.
- *regular:* Sticker werden in einer Größe von 136 x 136 Punkten dargestellt.
- *large:* Sticker werden in einer Größe von 206 x 206 Punkten dargestellt.

Der Initializer der Klasse namens *init(stickerSize:)* fragt genau diesen gewünschten Wert für die Stickergröße bereits bei der Erstellung eines neuen *MSStickerBrowserViewController*-Objekts ab und speichert diesen in der Read-only-Property *stickerSize*.

Die zweite Read-only-Property (und zugleich noch letzte Eigenschaft) der Klasse lautet *stickerBrowserView* und verweist auf eine Instanz der *UIView*-Subklasse *MSStickerBrowserView*. Diese Klasse kümmert sich letzten Endes um die Darstellung der Sticker in der CollectionView-ähnlichen Darstellung. Damit das aber funktioniert, muss die View wissen, welche Sticker sie überhaupt darstellen soll. Zu diesem Zweck verfügt die Klasse *MSStickerBrowserView* über eine *dataSource*-Property vom Typ *MSStickerBrowserViewDataSource*. Sofern nicht explizit anders konfiguriert, ist eine MSStickerBrowserViewController-Instanz standardmäßig als solche Data Source für die MSStickerBrowserView festgelegt.

Dabei bringt das MSStickerBrowserViewDataSource-Protokoll zwei Methoden mit, die beide zwingend implementiert werden müssen:

```
func numberOfStickers(in stickerBrowserView:
MSStickerBrowserView) -> Int
func stickerBrowserView(_ stickerBrowserView:
MSStickerBrowserView, stickerAt index: Int) -> MSSticker
```

Über die Methode *numberOfStickers(in:)* geben Sie die Anzahl der Sticker zurück, die innerhalb der MSStickerBrowserView dargestellt und angezeigt werden sollen. Die zweite Methode – *stickerBrowserView(_:stickerAt:)* – erwartet als Rückgabewert die MSSticker-Instanz für den übermittelten Index. Wenn Sie diese beiden Methoden entsprechend implementieren und damit die Informationen zu Ihren Stickern übergeben, können Sie auch in einer iMessage App mit relativ geringem Aufwand ein Sticker Pack umsetzen, das Sie überdies frei im Code konfigurieren und anpassen können.

Links zum Thema

- Messages-Framework-Referenz
  https://developer.apple.com/reference/messages
- IceCreamBuilder-Beispielprojekt
  https://developer.apple.com/library/content/samplecode/IceCreamBuilder/Introduction/Intro.html

Wichtig ist dabei nur, die zu erstellende Instanz des MSStickerBrowserViewControllers als Subview zu Ihrem MSMessagesAppViewController hinzuzufügen, diesen benötigen Sie nämlich weiterhin.

Am einfachsten gelingt Ihnen diese Umsetzung, indem Sie eine Container-View auf dem Interface des MSMessagesAppViewControllers im Storyboard platzieren, diese als Outlet mit Ihrem Code verknüpfen und anschließend dort die neu erstellte MSStickerBrowserViewController-Instanz hinzufügen.

Fazit

Zum ersten Mal in der Geschichte von iOS ist es mit dem neuen Messages-Framework möglich, Apples hauseigene Nachrichten-App um eigene Erweiterungen zu ergänzen. Neben den vorgestellten einfachen Sticker Packs lassen sich auch komplexe Anwendungen erzeugen, die auf Basis des Mitteilungsdienstes funktionieren.

Dabei sind sogar erstmals Standalone-Apps möglich, die ausschließlich aus der entsprechenden Erweiterung ohne zugrunde liegende iOS-App bestehen, auch wenn es selbstverständlich möglich ist, eine iOS-App zusätzlich um eine solche Erweiterung zu ergänzen. Erste Entwickler sind bereits auf den Zug aufgesprungen, und der Nachrichten-eigene App Store bot bereits vom Start weg eine Vielzahl an neuen Apps und Erweiterungen für die Nachrichten-App.

Abzuwarten bleibt, inwieweit diese spannenden neuen Möglichkeiten auch von den Nutzern angenommen werden und inwieweit sich damit ein weiteres Geschäftsfeld für App-Entwickler öffnet.

Die Umsetzung und Integration in das bestehende System ist Apple in jedem Fall gelungen, und auch das Messages-Framework ist für Entwickler sehr gut umgesetzt. Dank seines übersichtlichen und gut strukturieren API findet man sich schnell zurecht, und Code-Beispiele erleichtern den Einstieg in die verschiedenen Klassen. ■

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

IOS: PROPERTY LISTS

Liste mit Eigenschaften

Konfigurationsinformationen einer App werden in der Info.plist-Datei verwaltet.

Sie befindet sich in jedem Projekt – die *Info.plist*-Datei. Oft beachtet man sie überhaupt nicht, und trotzdem ist sie für ein Projekt von essenzieller Bedeutung. In einer *Info.plist*-Datei werden projektbezogen Informationen gespeichert. Der für ein Projekt gewählte BundleIdentifier kann beispielsweise in dieser gesichert werden, oder zumindest eine Referenz auf den verwendeten Wert. Auch weitere Konfigurationsinformationen wie zum Beispiel die Namen der verwendeten Storyboard-Dateien sind dort zu finden. Weitere App-bezogene Daten sind zum Beispiel die durch die App unterstützten Bildschirmausrichtungen oder die Versionsnummer.

Ansichten einer PropertyList

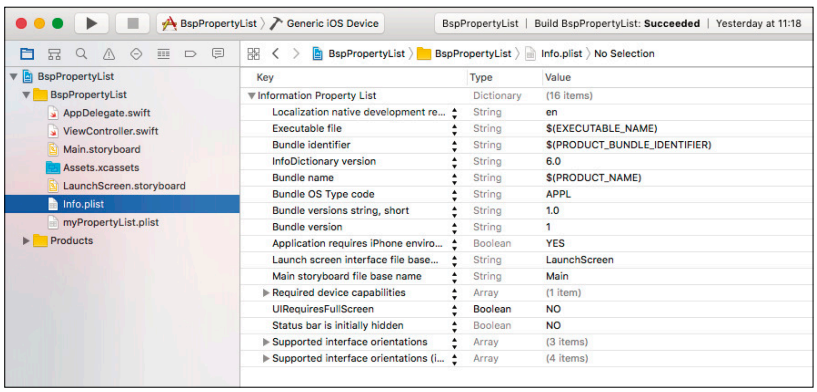
Bild 1 zeigt die grafische Darstellung der *Info.plist*-Datei eines Projekts. Diese Ansicht kennen Sie sicherlich und haben an dieser Stelle vielleicht auch schon einmal die eine oder andere Änderung vorgenommen. Eine *Info.plist*-Datei ist eine sogenannte PropertyList. Im Prinzip handelt es sich hierbei um ein XML-Dokument, das als Grundlage eine von Apple vorgegebene DTD (Document Type Definition) verwendet.

Das kann man sich sehr einfach ansehen, indem man die Darstellung der PropertyList in Xcode ändert. Problemlos ist das möglich, indem man die PropertyList markiert und anschließend das zugehörige Kontextmenü aufruft. Hierzu wählt man zuerst den Menüpunkt *Open As* und dann *Source Code*. Möchte man von der XML-Anzeige wieder zurück in den Übersichtsmodus, so muss man einfach den Eintrag *Open As, Property List* wählen. Den Inhalt einer *Info.plist*-Datei können Sie Listing 1 entnehmen.

Neben den zu Beginn bereits erwähnten Elementen enthält eine *Info.plist*-Datei natürlich noch viele weitere Einträge. In den ersten Zeilen befinden sich, wie vom XML-Standard her bekannt, Informationen zum Urheber des Dokuments. Eingeleitet wird die eigentliche Plist-Datei durch die Verwendung des Tags *<plist version="1.0">*. Geschlossen wird die *Info.plist* durch ein entsprechendes schließendes Tag. Alle in der Liste enthaltenen Werte sind in einem Dictionary (wird durch das Tag *<dict>* eingeleitet) abgelegt.

Tabelle 1: Verfügbare Typen

Swift-Typ	Beschreibung	Tag
Array [AnyType]	Wird zur Definition von Arrays verwendet	<array> </array>
Dictionary	[String:AnyType]	Wird zum Beispiel zur Definition von PropertyList verwendet
String	Für Zeichenketten	<string> </string>
NSData	Für objektbezogene Daten	<data> </data>
NSDate	Zur Speicherung von Datumswerten	<date> </date>
Int	Zur Darstellung von ganzen Zahlen	<int> </int>
Float, Double	Für Fließkommawerte	<real> </real>
Bool	Für Ja / Nein Werte	<true/> , <false/>
Schlüssel	Etwa in Dictionaries	<key> </key>
Eigenschaftenliste	Tag für PropertyList	<plist version="1.0"> </plist>



PropertyList eines Projekts (Bild 1)

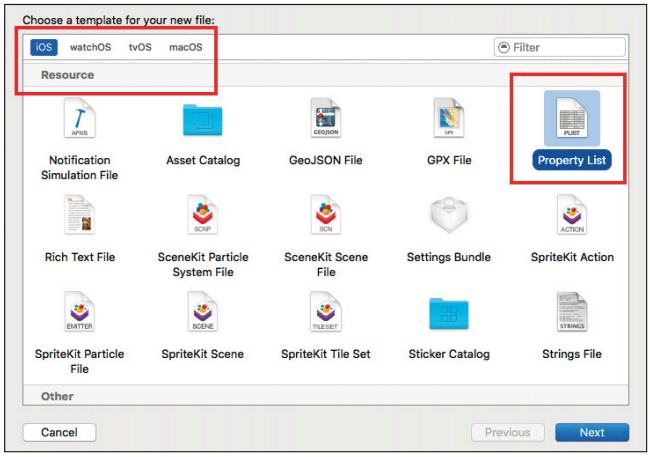
Typen in XML

Die verfügbaren Typen können Sie Tabelle 1 entnehmen. Die beschriebenen Typen sind Ihnen bekannt: Sie entsprechen natürlich den in Swift verfügbaren Typen. So gibt es etwa ein Tag, mit dem ein Array ausgewiesen werden kann. Aber auch für einfache Typen, zum Beispiel String oder Integer, findet sich in der Tabelle ein XML-Gegenstück. Bei der in Xcode-Projekten verwendeten PropertyList handelt es sich um ein Dictionary, das Key-Value-Paare enthält. Eine neue PropertyList können Sie jederzeit einem Projekt hinzufügen. Hierzu muss lediglich die entsprechende Vorlage in

Listing 1: Inhalt der Info.plist-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
    <key>CFBundleExecutable</key>
    <string>$(EXECUTABLE_NAME)</string>
    <key>CFBundleIdentifier</key>
    <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>$(PRODUCT_NAME)</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>1.0</string>
    <key>CFBundleVersion</key>
    <string>1</string>
    <key>LSRequiresIPhoneOS</key>
    <true/>
    <key>UILaunchStoryboardName</key>
    <string>LaunchScreen</string>
    <key>UIMainStoryboardFile</key>
    <string>Main</string>
    <key>UIRequiredDeviceCapabilities</key>
    <array>
      <string>armv7</string>
    </array>

    <key>UIRequiresFullScreen</key>
    <false/>
    <key>UIStatusBarHidden</key>
    <false/>
    <key>UISupportedInterfaceOrientations</key>
    <array>
      <string>UIInterfaceOrientationPortrait
      </string>
      <string>UIInterfaceOrientationLandscapeLeft
      </string>
      <string>UIInterfaceOrientationLandscapeRight
      </string>
    </array>
    <key>UISupportedInterfaceOrientations~ipad</key>
    <array>
      <string>UIInterfaceOrientationPortrait
      </string>
      <string>UIInterfaceOrientationPortraitUpsideDown
      </string>
      <string>UIInterfaceOrientationLandscapeLeft
      </string>
      <string>UIInterfaceOrientationLandscapeRight
      </string>
    </array>
  </dict>
</plist>
```

Xcode ausgewählt werden (Bild 2). Nach der Vergabe einer Bezeichnung wird die neue PropertyList erzeugt. Man kann diese entweder in der üblichen Ansicht bearbeiten oder aber direkt den Quellcode aufrufen (Listing 2). Gut im Code erkennbar ist die Struktur einer PropertyList. Für den Einsatz muss diese jetzt einfach nur noch mit Werten befüllt werden.



Eine neue PropertyList anlegen (Bild 2)

In einem kleinen Beispielprojekt soll eine PropertyList als Datenquelle für eine App dienen. Der Inhalt der PropertyList soll in einer TableView publiziert werden. Hierzu muss zunächst ein entsprechendes Projekt angelegt werden. Legen Sie eine neue Single View Application an, löschen Sie die vorhandene View im Storyboard und ersetzen Sie diese durch eine *TableViewController*-Komponente aus der Object Library von Xcode. ▶

Listing 2: Eine leere PropertyList

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD
PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">

<plist version="1.0">
  <dict>

  </dict>
</plist>
```

Listing 3: Standarisierte Vorgehensweise

```
import UIKit
class ViewController: UITableViewController {
  var languageData = [String]()
  var color1 = UIColor(
    red: 0.0,
    green: 0.0,
    blue: 0.8,
    alpha: 0.5)
  let color2 = UIColor(
    white: 0.2,
    alpha: 0.5)
  override func viewDidLoad() {
    super.viewDidLoad()
    let path = Bundle.main.path(forResource:
    "myPropertyList", ofType: "plist")
    let dict = NSDictionary(contentsOfFile: path!)
    languageData = dict!.object(forKey:
    "Programmiersprachen") as! [String]
  }
  override func numberOfSections(in tableView:
  UITableView) -> Int {
    return 1
  }
  override func tableView(_ tableView: UITableView,
  numberOfRowsInSection section: Int) -> Int {
    return languageData.count
  }
  override func tableView(_ tableView: UITableView,
  titleForHeaderInSection section: Int) -> String? {
    return "Programmiersprachen:"
  }
  override func tableView(_ tableView: UITableView,
  cellForRowAt indexPath: IndexPath) ->
  UITableViewCell {
    let cell = tableView.dequeueReusableCell
    (withIdentifier: "myCell", for: indexPath)
    // Configure the cell...
    cell.textLabel!.text = languageData
    [(indexPath as NSIndexPath).row]
    cell.backgroundColor = color2
    cell.textLabel?.textColor = color1
    return cell
  }
}
```

Im Anschluss sollten Sie zuerst in der View die Eigenschaft *Is Initial View Controller* aktivieren, damit die View als Einstiegspunkt in die App verwendet wird. Konfigurieren Sie dann die Zelle in der TableView mittels *Style: Basic* und *Identifier: myCell*. Der Rahmen für die Anzeige ist damit gebaut. Als Nächstes muss, wie bereits beschrieben, eine neue PropertyList in das Projekt eingefügt und konfiguriert werden. In der PropertyList sollen einige Programmiersprachen eingefügt werden. Das kann man entweder über den Editor in Xcode oder aber direkt im Quelltext erledigen.

Links zum Thema

- Framework Reference
  https://developer.apple.com
- Info.plist Apple Developer
  https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html

Für jede Sprache wurden im Dictionary ein Key- sowie ein Value-Feld erzeugt. Die Liste selbst wird in einem Array gehalten, das Bestandteil des Dictionarys ist. Für die Verwendung der PropertyList als Datenquelle für die TableView schlägt man dann die standardisierte Vorgehensweise ein (Listing 3).

Das Listing beginnt mit der Definition eines String-Arrays, das später die Werte der PropertyList enthält. Zu Beginn werden dann noch zwei Farbräume definiert, damit die Zeilen in der TableView unterschiedlich hervorgehoben werden. Los geht es dann in der Methode *viewDidLoad*. Zuerst wird der Standort der PropertyList ermittelt, ausgelesen und in der Variablen *path* gespeichert.

Im folgenden Schritt wird ein neues Dictionary vom Typ *NSDictionary* angelegt und die Variable *path* wird als Parameter übergeben. Danach wird im letzten Schritt innerhalb der Methode *viewDidLoad* die PropertyList ausgelesen und deren Inhalt der Variablen *languageData* zugewiesen. Im Code folgen die üblichen Methoden zum Anzeigen des Inhalts der Variablen innerhalb der TableView. So wird in der Methode *numberOfSection* die entsprechende Anzahl zur Konfiguration der TableView angegeben.

Wie viele Zeilen in der TableView anzuzeigen sind, gibt die Methode *numberOfRowsInSection* durch Auslesen der Eigenschaft *count* zurück. Mittels der Methode *titleForHeaderInSection* wird eine Überschrift eingefügt. Der Aufbau der Zelle(n) geschieht dann in der Methode *cellForRowAt indexPath*. Über den Identifier *(myCell)* wird eine Zelle angesprochen. Anschließend wird die Referenz auf die Zelle *(cell)* verwendet, um über die Eigenschaften den Inhalt *(text)* und das Aussehen *(BackgroundColor, TextColor)* der Zelle festzulegen. Zuletzt wird die fertig formatierte Zelle aus der Methode zurückgegeben. ■

Christian Bleske
ist Autor, Trainer und Entwickler mit dem Schwerpunkt Client/Server und mobile Technologien. Erreichbar ist er unter:
cb.2000@hotmail.de

developer media

developer media Trainings

Updates für Ihr Know-How

„Entwickler sollten Ihr Framework so gut kennen wie der Handwerker seinen Werkzeugkoffer – auswendig!“

Christian Giesswein
.NET-Experte, Consultant und Trainer

Test Driven Development

Trainer: Hendrik Lösch

2 Tage, 28.-29.11.2016, Köln
Ab 1.799,- EUR zzgl. MwSt.

Java für Einsteiger

Trainer: Alexander Salvanos

3 Tage, 21.-23.11.2016, Köln
Ab 2.199,- EUR zzgl. MwSt.

Modulare WPF-Anwendungen mit PRISM

Trainer: Christian Giesswein

3 Tage, 30.11.-02.12.2016, Köln
Ab 2.199,- EUR zzgl. MwSt.

Java für Fortgeschrittene

Trainer: Alexander Salvanos

3 Tage, 28.-30.11.2016, 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

USABILITY-RICHTLINIEN

Schwierige Aufgabe

Gute Usability für mobile Apps zählt zu den schwierigeren Aufgaben eines Entwicklers.

Bild: shutterstock / MaxxiGo

Durch die Vielfalt der am Markt vorhandenen mobilen Geräte entsteht für die Entwicklung einer mobilen App eine besondere Situation. Alle Geräte mit Android als Betriebssystem unterscheiden sich hinsichtlich CPU, Bildschirmgröße, Bildschirmauflösung und verfügbarem Arbeitsspeicher. Apple hingegen bietet eine kleinere Anzahl von Geräten für sein Betriebssystem iOS an, aber ebenfalls mit unterschiedlichen Merkmalen.

Besondere Rahmenbedingungen von mobilen Geräten

Die nachfolgenden Richtlinien sollen helfen, eine bessere Usability für mobile Geräte zu schaffen und interne Standards für die Entwicklung von Apps zu erstellen.

Portieren Sie keine Bedienoberflächen von einem Designmodell in ein neues Designmodell. Dieser Grundsatz ist sicherlich nicht nur für mobile Applikationen gültig, sondern auch generell ein Grundsatz bei der Software-Entwicklung von Desktop-Applikationen. Bei der nativen App-Entwicklung müssen meistens mehrere App Stores – gemäß ihrer Marktanteile – berücksichtigt werden. Bei jeder App sollten auf jeden Fall die Eigenschaften und Merkmale des jeweiligen Zielsystems berücksichtigt werden. Auch Auftraggebern gegenüber sollte man sich immer durchsetzen, da es kaum einen Anwender geben wird, der beide Systeme bedient. Der Anwender der App kennt nur die Merkmale seines verwendeten Systems – und weiß dieses auch zu schätzen.

Chrome-Elemente

Chrome-Elemente sind sichtbare Designelemente, die Informationen über den Bildschirminhalt liefern, beziehungsweise Steuerelemente, die für das Handling der Inhalte wichtig sind. Ein Beispiel hierfür sind Suchleisten und Symbolleisten innerhalb einer App. Chrome-Elemente haben die Eigenschaft, dass sie auf einem kleinen Display viel Platz benötigen und somit wenig Raum für den eigentlichen Inhalt einer App bieten. Deshalb sollte man sehr sparsam mit der Platzierung von Chrome-Elementen sein und diese auch als Bedienelement deutlich machen. Sicherlich gibt es Apps, deren Chrome-Elemente wichtig für den Kontext der App sind. Diese stellen aber eine Ausnahme dar. Wenn man Chrome-Elemente in die Oberfläche einbindet, dann kann das Anzeigen der Elemente auch beeinflusst werden.

Jedes mobile Betriebssystem bringt eine Anzahl von Standard-Apps mit (Bild 1)

Jedoch gibt es Alternativen zu Chrome-Elementen. Eine gängige Alternative ist die Gestensteuerung, die auf allen mobilen Betriebssystemen vorhanden ist.

Jedoch sollten diese Gesten auch nur in bekanntem Kontext genutzt werden, da ansonsten die individuelle Geste dem Anwender nicht bekannt sein wird. Die Folge ist ein weiteres Chrome-Element, das die Gestensteuerung deutlich macht und erklärt.

Bekannte Eingabeelemente wiederverwenden

Jedes mobile Betriebssystem bringt eine Anzahl von Standard-Apps mit. Diese sind somit kostenlos und werden vom Anwender stärker genutzt als gekaufte Apps.

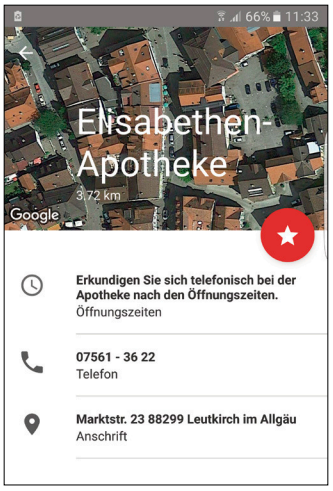
Deshalb sollte man darauf achten, dass man die aus diesen Apps bekannten Ein-

gabeelemente immer auch in den eigenen Apps wiederverwendet (Bild 1).

Eingabeelemente wie Suchleisten und Suchfelder sollten auch in der eigenen App wiederverwendet werden, sofern es in den Kontext der App passt. Eigene Varianten eines Suchfelds sollten vermieden werden.

Auch hier bringt jedes mobile Betriebssystem immer einen Satz von Icons mit, die jeder Anwender kennt und hinter denen er eine bestimmte Funktion erwartet.

Ein Fehler besteht dann darin, hinter einem Standard-Icon eine andere Funktion zu hinterlegen als die Funktion, die der Anwender erwartet. Ein weiterer Fehler besteht darin, für eine Standardfunktion kein Standard-Icon zu verwenden. Deshalb sollten Sie bei der Verwendung von Icons immer die Notwendigkeit eines Standard-Icons hinterfragen und gegebenenfalls prüfen, ob das jeweilige SDK hierfür eine Möglichkeit bietet.



Call-To-Action-Elemente sollten als solche gut erkennbar sein (Bild 2)

Den vorhandenen Platz ausnutzen

Da die Anzeigemöglichkeiten auf mobilen Geräten begrenzt sind, sollte die Anzeige auch möglichst ausgenutzt werden. Versuchen Sie, die Oberfläche nicht zu überladen, aber dennoch die vorhandene Anzeigefläche auszunutzen.

Eine App, die dauerhaft im App Store bestehen will, muss eine leistungsfähige und umfangreiche Funktionalität anbieten. Aber nicht jede Funktionalität muss den direkten Weg in das GUI finden. Hier ist es wichtig, dass man zwischen Funktionen unterscheidet, die immer benötigt werden, und solchen, die man nur in einem bestimmten Kontext nutzen möchte.

Deshalb sollte jede App immer einen Button in der Symbolleiste anbieten, um weitere Optionen anzuzeigen. Hier sollten dann Funktionen erscheinen, die der Anwender nicht immer benötigt, sondern nur in einem bestimmten Kontext. Typische Funktionen hierfür sind:

- Per E-Mail versenden,
- Drucken,
- Teilen (Facebook, Twitter, …),
- Exportieren,
- Speichern in der Fotogalerie,
- Öffnen mit …

Wenn eine App entwickelt wird, dann wird meistens erst zum Schluss der Entwicklung auf einem Tablet oder Smartphone getestet. Bis dahin wird die App im jeweiligen Simulator der Entwicklungsumgebung getestet und das GUI wird mit der Maus bedient und gesteuert. Hierbei wird oft vergessen, dass ein Finger etwas dicker ist, und die Eingabeelemente werden dann zu nah an anderen Elementen platziert. Versucht der Anwender, ein Eingabeelement zu bedienen, besteht die Gefahr, ein anderes Eingabeelement zu erwischen. Wenn dieses Element dann eine ganz andere Funktionalität besitzt, ist der Ärger beim Anwender verständlich.

Deshalb muss man bei der Platzierung von Oberflächenelementen immer diesen Effekt berücksichtigen und das komplette GUI auch auf realen Geräten (Tablet, Smartphone) testen.

Call-To-Action-Elemente sichtbar machen

Ein Call-To-Action-Element ist ein Bestandteil einer Oberfläche, der den Anwender zu einer Aktion auffordert. Beispiele hierfür sind Buttons oder Links zu Webseiten, die der Anwender antippen kann und die dann eine Aktion auslösen.

Jedes SDK bietet einem Entwickler die Möglichkeit, auch eigene Buttons zu erzeugen, nur leider sind diese nicht immer auch als Call-To-Action-Element ersichtlich. Manchmal wirken diese in Kombination mit einen Bild eher wie ein einfaches Icon. Auch eigene Links sollten entsprechend sichtbar gemacht werden. Typischerweise sind diese blau unterstrichen. In Bild 2 ist eine Info-Seite sichtbar, die einen Button mit einem Telefonsymbol neben der Telefonnummer beinhaltet. Der Anwender hat hier die Möglichkeit auf diesen Telefon-Button zu tippen, um die nebenstehende Telefonnummer anzurufen.

Bevor eine App in den jeweiligen App Store übertragen wird, ist es selbstverständlich, diese ausgiebig zu testen. Jedoch testet meistens der Entwickler der App selbst und nicht eine an der Entwicklung unbeteiligte Person.

Aber genau das wird der Anwender der App sein. Deshalb sollte man die App noch von einer unbeteiligten Person testen lassen, die dann die eigenen Erfahrungen in die Entwicklung mit einbringen kann. Ein Entwickler testet meistens nur in einem begrenzten Umfang, da er noch andere Entwicklungsaufgaben wahrnehmen muss.

Fazit

Neben der Vielfalt von Geräten und den damit verbundenen unterschiedlichen Auflösungen ist der Nutzungskontext entscheidend. Erfolgreiche Apps im App Store zeichnen sich durch eine gute und durchdachte Usability aus. Und genau darum geht es vor der Veröffentlichung: Die Usability sollte gut durchdacht und dem Kontext angepasst werden. Dabei sollten Standardelemente des jeweiligen Betriebssystems berücksichtigt werden und mit in die Oberfläche einfließen. ■

Dipl.-Ing. Andreas Sommer
ist Geschäftsinhaber von Engenious – Business Solutions & Engineering.
www.engenious.de

ANDROID STUDIO 2.2

Interessante Funktionen

Die Version 2.2 von Android Studio bietet vielfältig nutzbare neue Funktionen an.

Wer noch mit Eclipse arbeitet, sollte schleunigst eine Umstellung ins Auge fassen: Viele neue Features werden für die alte IDE schlichtweg nicht angeboten. Dieser Artikel stellt die interessantesten davon vor – vielleicht motiviert dies ja zum Wechsel.

Wer sich dagegen schon mit Android Studio auskennt, findet hier interessante Hinweise zur Nutzung der neuen Funktionen. Einige davon sind insbesondere beim Handling von Fragmentierung Gold wert.

Update auf die neue Version

Wer Android Studio noch nicht auf seiner Workstation installiert hat, findet hierzu unter https://developer.android.com/studio/install.html Informationen. Wer die IDE – wie die meisten Entwickler – schon hat, kommt am einfachsten über das in IntelliJ integrierte Update-Feature zur neuen Version 2.2.

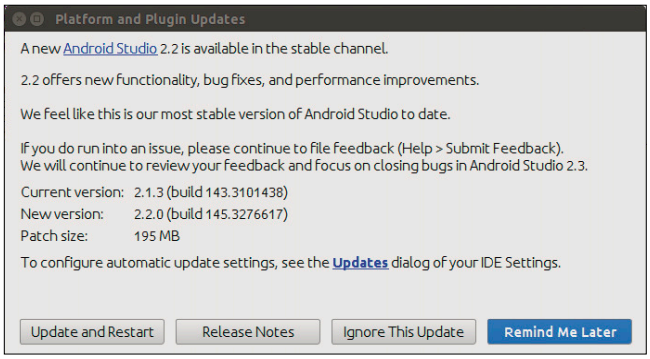
Öffnen Sie dazu Android Studio. Je nach Konfiguration erscheint entweder ein Fly-in oder sogar ein Fenster, das Sie zum Herunterladen der neuen Version animiert. Wer dieses Fenster aus irgendeinem Grund nicht sieht, kann auch unter *Help* und *Check for updates* eine Aktualisierung anfordern.

Die IDE reagiert darauf mit dem Einblenden des in Bild 1 gezeigten Fensters. Klicken Sie auf *Update and Restart*, um die Änderungen herunterzuladen. Geben Sie Android Studio etwas Zeit: Die Aktualisierung erfolgt durch ein Patchfile, das auf die schon vorhandene Installation angewendet werden muss und einiges an Rechenleistung in Anspruch nimmt.

Android Studio wird sich während der Installation manchmal über das Schon-Vorhandensein von Dateien beklagen, die sich auf Konfigurationseinstellungen der virtuellen Maschine beziehen. Der im Installationswerkzeug enthaltene Assistent bietet an, diese zu überschreiben. Nehmen Sie diese Option ohne Bedenken an, in der Praxis kommt es nicht zu Problemen. Wer ganz auf Nummer sicher gehen will, kann zudem noch ein Backup anlegen.

Beim nächsten Start fragt die IDE, ob Sie die vorhandenen Einstellungen übernehmen wollen. War Ihre Installation von Android Studio zuvor weitestgehend sinnvoll konfiguriert, so können Sie dies ohne Bedenken annehmen. Nach dem Bestätigen dieses Dialogs startet Android Studio wie gewohnt durch. Im Normalfall finden Sie sich dort wieder, wo Sie schon vorher waren. Ein eventuell geöffnetes Projekt muss in das neue Dateiformat konvertiert werden – aufgrund von Änderungen in der Projektstruktur sind die älteren Dateien nicht kompatibel.

Bei dieser Gelegenheit sollten Sie auch die Plattform-Tools, die Plattform und alle SDKs aktualisieren. Google nimmt hier immer wieder Änderungen vor, die Fehler beheben und die Performance verbessern.



Google hilft Entwicklern gerne beim Aktualisieren (Bild 1)

Erweiterung von IntelliSense

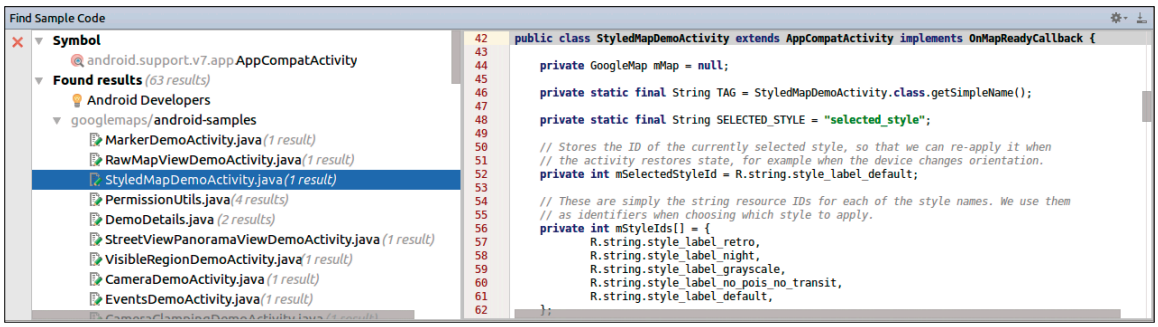
Android gilt als nicht sonderlich gut dokumentierte Plattform. Wer mehr über das Betriebssystem erfahren möchte, muss mitunter in den Quellcode blicken. Die von Google zahlreich bereitgestellten Beispielprogramme waren bisher insofern uninteressant, als man sie nur schwer nach Interessantem durchsuchen konnte.

Android Studio 2.2 löst dieses Problem durch eine kleine Erweiterung von IntelliSense, die nun auch die Android-Beispielrepositories nach interessanten Codepassagen durchsucht. Wer diese Funktion nutzen möchte, legt den Mauszeiger auf ein beliebiges Symbol und klickt dieses rechts an. Im daraufhin erscheinenden Pop-up-Menü gibt es eine Option namens *Find Sample Code*, die sich auch durch Drücken von [Alt F8] aktualisieren lässt.

Die IDE macht sich daraufhin – bei bestehender Internetverbindung – an die Arbeit und blendet nach einigen Sekunden das in Bild 2 gezeigte Fenster ein. Das doppelte Anklicken eines Treffers öffnet die jeweilige Codedatei im Browser Ihrer Workstation. Durch einfaches Anklicken wird der auf der linken Seite gezeigte Codeviewer eingeblendet, der die direkte Umgebung des jeweiligen Treffers anzeigt.

Sensoren nutzen

Wer umgebungssensitive Applikationen realisieren möchte, handelt sich beim Testen zwangsweise Ärger ein. Nutzt man den Temperatursensor eines Telefons, so sollte man einen Kühlschrank und eine Heizkammer in der Nähe haben oder eine Mockingklasse schreiben. Beides ist in der Praxis nur wenig befriedigend.



Android Studio 2.2 durchsucht Googles Beispielcode automatisch nach Interessantem (Bild 2)

Android Studio 2.2 begegnet diesem Problem durch die Einführung von Erweiterungen im Emulator, die wir an dieser Stelle kurz antesten wollen.

Emulatoren genießen wegen ihrer Langsamkeit eine schlechte Reputation. Googles Emulator ist hier eine absolute Ausnahme. Wer ein x86-Image verwendet und auf seiner Workstation Hardwarevirtualisierung aktiviert, erreicht eine mehr als akzeptable Performance.

Zum Zeitpunkt der Drucklegung kennt Android ein gutes Dutzend Sensoren, die allerdings nicht in allen API-Levels zur Verfügung stehen. Tabelle 1 zeigt, welche Sensoren ab welcher Betriebssystemversion einsetzbar sind.

Wir wollen für die folgenden Schritte auf den Temperatursensor zurückgreifen, der je nach Implementierung des Herstellers entweder die Umgebungstemperatur oder die der Smartphone-Hardware misst.

Öffnen Sie die IDE und erstellen Sie ein neues Projekt namens *NMGTemperature*. Zur Nutzung des Sensor-API sind drei Erweiterungen in *MainActivity* erforderlich. Erstens müssen wir das Interface *SensorEventListener* implementieren, um über Informationen, die im Sensor anfallen, informiert zu werden. Zweitens brauchen wir lokale Variablen vom Typ *SensorManager* und *Sensor*. Erstere ist für die Instanz beziehungsweise die Referenz auf den Sensormanager des Betriebssystems verantwortlich, während das zweite Objekt den eigentlichen Sensor repräsentiert:

```
public class MainActivity extends AppCompatActivity
implements SensorEventListener{
  SensorManager mySensorManager;
  Sensor mySensor;
```

Im Konstruktor findet sich gewohnte Kost. Wir nutzen im ersten Schritt die Methode *getSystemService*, um einen Verweis auf den vom System bereitgestellten Sensormanager zu erhalten. Dieser wird im nächsten Schritt nach seinem empfohlenen Temperatursensor befragt, der einen Listener eingeschrieben bekommt:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  mySensorManager=(SensorManager)
  getSystemService(Context.SENSOR_SERVICE);
  if ((mySensor=mySensorManager.getDefaultSensor
  (Sensor.TYPE_AMBIENT_TEMPERATURE)) != null){
    mySensorManager.registerListener(this, mySensor,
    SensorManager.SENSOR_DELAY_NORMAL);
  }
  else {  }
}
```

Manche Smartphones enthalten mehrere identische Sensoren. Wer eine Liste aller Sensoren eines bestimmten Typs erhalten möchte, erreicht dies durch Aufrufen des folgenden Codesnippets:

```
List<Sensor> deviceSensors = mySensorManager.
getSensorList(Sensor.TYPE_AMBIENT_TEMPERATURE);
```

Die Methode zur Beschaffung der Sensorlisten ist übrigens nicht auf das Anliefern von Listen bestimmter Sensorarten beschränkt. Wer als Parameter *XXX* übergibt, erhält eine Liste aller auf der Zielhardware vorhandenen Sensoren. ▶

Tabelle 1: Sensoren/Betriebssystemversion

Sensor	Version
TYPE_ACCELEROMETER	1.5
TYPE_AMBIENT_TEMPERATURE	4.0
TYPE_GRAVITY	2.3
TYPE_GYROSCOPE	2.3
TYPE_LIGHT	1.5
TYPE_LINEAR_ACCELERATION	2.3
TYPE_MAGNETIC_FIELD	1.5
TYPE_ORIENTATION	1.5, deprecated
TYPE_PRESSURE	2.3
TYPE_PROXIMITY	1.5
TYPE_RELATIVE_HUMIDITY	4.0
TYPE_ROTATION_VECTOR	2.3
TYPE_TEMPERATURE	4.0, deprecated

Das Interface zwingt uns nun zur Implementierung der folgenden beiden Funktionen:

```
@Override
public void onSensorChanged(SensorEvent event) {
}
@Override
public void onAccuracyChanged(Sensor sensor,
int accuracy) {
}
```

onSensorChanged ist dabei für das Entgegennehmen von Sensorereignissen erforderlich, während *onAccuracyChanged* immer dann zum Einsatz kommt, wenn sich die Genauigkeit des zugrunde liegenden Sensors geändert hat.

Ein im Hintergrund laufender Sensor kann eine große Menge von Messdaten erzeugen. Da deren Verarbeitung Rechenleistung verbraucht, wirkt sich dies im Gesamtbild sehr negativ auf die Systemperformance beziehungsweise die Batterielaufzeit aus.

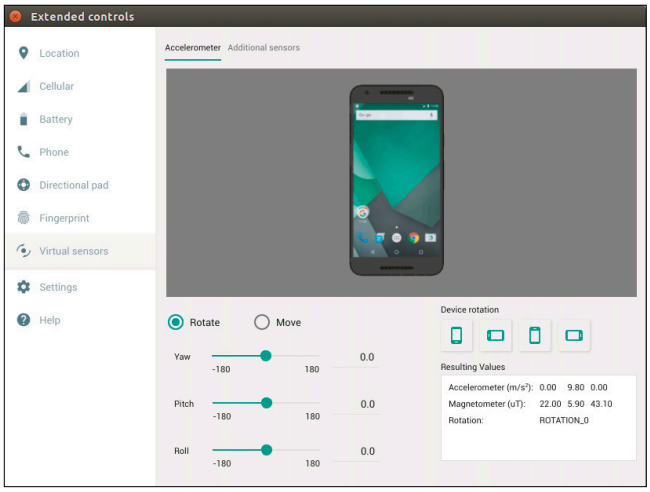
Aus diesem Grund wäre es grob fahrlässig, Ihnen folgendes Codestück nicht ans Herz zu legen. Es deaktiviert den Sensor durch Abmeldung eines Listeners, wenn die Activity pausiert wird:

```
protected void onPause() {
  super.onPause();
  mySensorManager.unregisterListener(this);
}
```

An dieser Stelle stehen wir einer neuen Aufgabe gegenüber: Die vom Sensor angelieferten Informationen müssen angezeigt werden. Wer die Datei *activity_main.xml* öffnet, erlebte in Version 2.2 ein kleines Überraschungserlebnis: Der Layout-Editor präsentiert sich nun wie in Bild 3 gezeigt

Die aus optischer Sicht am stärksten hervortretende Änderung ist mit Sicherheit das Blaupausenfeld, das normalerweise rechts vom schon bekannten Rendering angezeigt wird. Es ist insofern interessant, als es eine auf die Bedürfnisse von Designern optimierte Darstellung des Formulars anbietet. Ein Blick auf die TextView genügt, um sich von den Vorteilen der neuen Darstellungsform zu überzeugen.

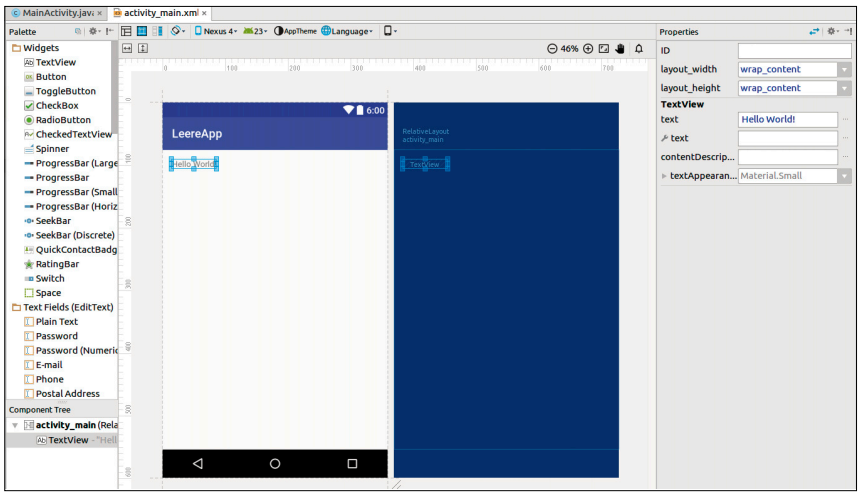
Diese Information suchen Sie in der grafischen Ansicht vergebens (Bild 4)



Accelerometer-Gesten lassen sich durch Verschieben des virtuellen Telefons eingeben (Bild 5)

Ein weiteres angenehmes Attribut ist das auf der rechten Seite des Bildschirms eingeblendete Eigenschaftsfenster, in dem Sie – ganz im Stil von Visual Studio – die Eigenschaften des gerade selektierten Steuerelements an Ihre Bedürfnisse anpassen können.

Für unsere Bedürfnisse reicht es fürs Erste aus, wenn Sie die Breite der Textbox maximieren und ihr eine ID zuweisen. Achten Sie dabei auf das Verhalten der Blaupausenansicht. Der Name des Steuerelements erscheint, wie in Bild 4 gezeigt, in der Mitte des jeweiligen Widgets.



Google hat den Layout-Editor stark überarbeitet (Bild 3)

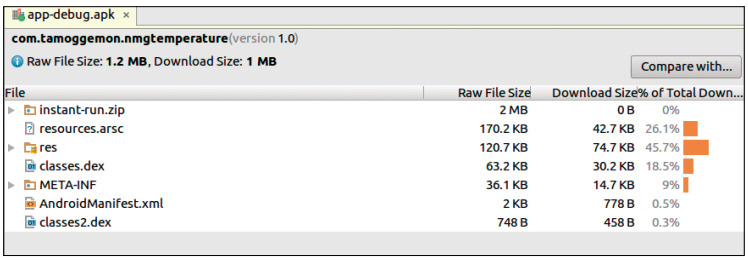
Blaupausenansicht

Die Blaupausenansicht wird ihre Stärken im nächsten Schritt weiter ausspielen. Die als Constraint bezeichnete neue Layoutgattung hilft bei der Realisierung von Formularen, bei denen die Steuerelemente ineinander verschachtelt sind.

Wir wollen an dieser Stelle aber mit unserem Test weitermachen. Kehren Sie zur Funktion *onSensorChanged* zurück und erweitern Sie sie um folgenden Event Handler:

```
@Override
public void
onSensorChanged(SensorEvent event) {
```

```
String aString="";
for(int i=0;i<event.values.length;i++) {
  aString+=(new Float(event.values[i])).
  toString() + "\n";
}
((TextView)findViewById(R.id.
TxtSensorData)).setText(aString);
... }
```



Das Vorhandensein der Datei *instant-run.zip* verheißt Böses (Bild 7)

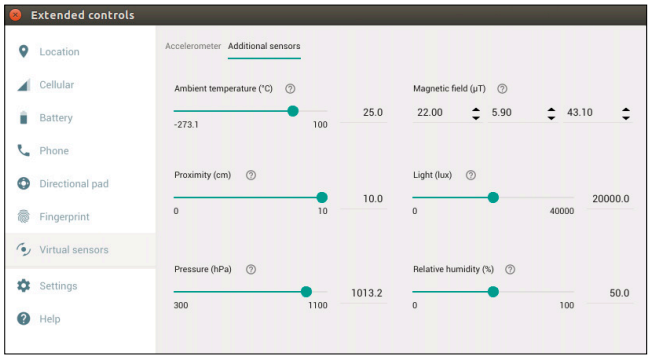
Der hier verwendete Code ist nur insofern interessant, als er das von *findViewById* zurückgelieferte Objekt über einen komplexen Cast direkt weiterverwendet. Viel relevanter ist, wie sich das Programm zur Laufzeit verhält. Erstellen Sie im ersten Schritt unter Nutzung des AvD-Managers eine neue virtuelle Maschine. Eine vergleichsweise untergeordnete Rolle spielt hierbei das Betriebssystem. Starten Sie diese danach und schicken Sie die Applikation in Richtung des virtuellen Geräts.

Die Applikation wird sofort nach dem Start eine Serie von drei Werten anzeigen. Klicken Sie auf die drei nebeneinanderliegenden Punkte neben dem Emulatorfenster, um die erweiterten Einstellungen auf den Bildschirm zu holen. In der Rubrik *Virtual Sensors* finden sich die in Bild 5 und Bild 6 gezeigten Fenster. Sie erlauben das Einstellen von diversen Sensoreigenschaften.

Analysiere das APK

Googles Binärformat ist eine klassische Fehleinschätzung der Weiterentwicklung von Technologien. Als *.dex*-Files erstmals spezifiziert wurden, ging man bei Google davon aus, dass praktische Handcomputer-Applikationen nie mehr als 65.000 Methodeneinträge enthalten könnten. Das rapide Zunehmen an Funktionen und das Verfügbarwerden vorgefertigter SDKs haben einen Verfettungsprozess in Gang gesetzt. Wer einige Bibliotheken von Drittanbietern einbindet und fleißig draufloscodet, donnert bald in die Begrenzung.

Die praktische Erfahrung von Generationen von Entwicklern lehrt, dass die Korrelation zwischen Code und erzeugter APK-Datei oft nur sehr uneindeutig ist. Zwischen Kompilation, Optimierung und automatischer Codegenerierung ist reichlich Platz für Schlupf.



Die Werte der anderen Sensoren lassen sich durch Slider beeinflussen: 25 Grad werden beim Temperaturmanager als 0 retourniert (Bild 6)

Android Studio 2.2 löst dieses Problem durch ein neues Analysewerkzeug, das seine Arbeit ausschließlich anhand fertiger APK-Dateien bewerkstelligt.

Googles Boris Farber entwickelt mit dem unter http://classyshark.com bereitstehenden ClassyShark seit langer Zeit ein Alternativprogramm, das dem in Android Studio integrierten APK-Analysator insbesondere im Bereich der Visualisierung und des GUI wesentlich überlegen ist.

Wer ein schon vorhandenes APK mit der neuen Funktion auf seinen Inhalt überprüfen möchte, legt es einfach per Drag and Drop in einem Editorfenster ab.

Die in einem für Instant Run freigegebenen Codeprojekt vorliegenden APKs – normalerweise handelt es sich hierbei um die Projektausgabe – sind für eine derartige Analyse auf keinen Fall brauchbar. Aufgrund der Nutzung von Instant Run sind sie nur partiell und würden bei der Analyse falsche Werte ergeben. Nutzen Sie stattdessen die Menüoption *Build* und *Build APK*, die eine vollwertige APK-Datei erzeugt. Übrigens lassen sich Instant-Run-APKs anhand des in Bild 7 gezeigten Archivs zuverlässig erkennen.

Beachten Sie, dass Android Studio zum Zeitpunkt der Drucklegung einen kleinen Bug aufweist: Wenn die APK-Analyse ein APK mit einem bestimmten Namen geladen hat, führt das Öffnen eines gleichnamigen APK nicht zu einer Aktualisierung des Bildschirminhalts. Es ist aus diesem Grund empfehlenswert, die Analyseansicht immer zu schließen, bevor man eine neue Datei lädt.

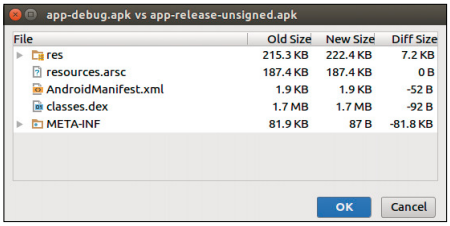
Der eigentliche APK-Editor ist in zwei Teile geteilt: Auf der Oberseite des Bildschirms findet sich eine gewöhnliche Dateiliste, die die diversen in der APK-Datei enthaltenen Ressourcen auflistet. Das Anklicken einer DEX-Datei blendet darunter ein weiteres Panel ein, das die Methoden nach Namespace aufsplittet und so grundlegende Möglichkeiten zur Reflexion in die jeweilige Klasse bietet.

Nutzer von ClassyShark vermissen an dieser Stelle allerdings die Möglichkeit zum automatischen Generieren von Headern. Auf der Habenseite von Android Studio steht hingegen die Möglichkeit, zwei APKs zu öffnen. Dazu ist ein kleiner Kniff erforderlich: Öffnen Sie im ersten Schritt ein APK und laden Sie danach das zweite durch Anklicken des *Compare with ...*-Buttons.

Ein sehr schöner Weg zur Nutzung des Vergleichsfeatures ist das Feststellen der Größenreduktion, die durch das Aktivieren des Release-Buildmodus erreicht wird. Im Fall unseres Sensorbeispiels sieht das erreichbare Ergebnis wie in Bild 8 aus. ▶

Verschiedene Menschen haben verschiedene Bedürfnisse an ihre Smartphones. Mittlerweile hat sogar Apple dieses einfache und doch weltbewegende Phänomen erkannt. Da das leidig-debile Thema der Fragmentierung von Android nicht in der Versenkung verschwinden möchte, bringt Google Entwicklern immer wieder neue Layout-Helferlein.

Mit dem in Android Studio 2.2 eingeführten ConstraintLayout betritt man insofern Neuland, als es sich hierbei um ein Steuerelement handelt, das sich an den von Qt bekannten Regeln orientiert. Dazu eine kurze Erklärung: In Qt bestehen Formulare aus Steuerelementen, die auf Bahnen fahren und



Die Nutzung von Release 2.2 führt zu wesentlich kleineren *.apk*-Dateien (Bild 8)

von Federn auf Distanz gehalten werden. Der englische Begriff Constraint bedeutet soviel wie Einschränkung oder Verbindung: Das Layout basiert auf dem Prinzip, dass Steuerelemente Beziehungen zueinander aufbauen und anhand dieser am Bildschirm ausgerichtet werden.

Realisierung eines numerischen Eingabefelds

Wir wollen uns dies anhand eines kleinen Beispiels näher ansehen, das ein numerisches Eingabefeld realisiert. Erzeugen Sie im ersten Schritt ein neues Projekt namens *NMGConstraint1* – Google unterstützt ConstraintLayouts ab Android 2.3. Das Layout wird von Google in Form einer Erweiterungsbibliothek ausgeliefert, die jedoch nicht von Haus aus auf Ihre Workstation wandert. Öffnen Sie deshalb nach dem Erstellen des Projekts den unter *Tools* und *Android* bereitstehenden SDK-Manager und installieren Sie die Module *Support Repository ConstraintLayout for Android* und *Support Repository Solver for ConstraintLayout*.

Nach dem erfolgreichen Herunterladen blendet Android Studio eine Versionsnummer an ein. Beachten Sie auch, dass das Paket im aus der Kommandozeile aufrufbaren SDK-Manager nicht zur Verfügung steht. Es kann nur über das in Android Studio enthaltene Downloadwerkzeug bezogen werden. Öffnen Sie im nächsten Schritt die *build.gradle*-Datei und fügen Sie das Modul nach folgendem Schema ein:

```
dependencies {
  compile 'com.android.support:appcompat-v7:23.4.0'
  compile 'com.android.support.constraint:
  constraint-layout:1.0.0-alpha9'
  testCompile 'junit:junit:4.12'
}
```

Öffnen Sie im nächsten Schritt die Datei *activity_main.xml* und klicken Sie das von Google von Haus aus angelegte *RelativeLayout* im Komponentenbaum an. Er befindet sich nun auf der unteren linken Seite des Bildschirms. Wählen Sie im daraufhin erscheinenden Pop-up-Menü die Option *Convert RelativeLayout to ConstraintLayout* aus und lassen Sie die beiden Häkchen vor *Voreinstellungen* aktiviert.

Nach dem Anklicken von *OK* verändert sich das Aussehen der Blaupausenansicht. Löschen Sie die TextView und ziehen Sie stattdessen neun Buttons in das Steuerelement. Klicken Sie im nächsten Schritt einen der Buttons an. Im Vergleich zu klassischen Layouts erscheinen nun sowohl in der Rendering- als auch in der Blaupausen-Ansicht vier weitere runde Punkte in der Mitte der jeweiligen Begrenzungslinie. Es handelt sich dabei um die Steuerelemente, die zum Einrichten der Constraint-Beziehungen vorgesehen sind.

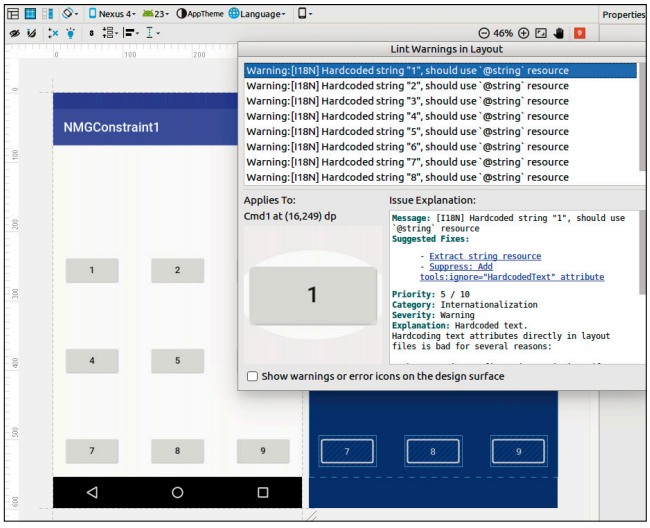
Steuerelemente in einem ConstraintLayout

Wir beginnen unsere Experimente mit dem Knopf, der am Ende ganz links oben sein wird. Markieren Sie ihn in der Blaupausenansicht und klicken Sie in den oberen Kreis. Ziehen Sie daraufhin bei gedrückt gehaltener Maustaste eine Linie zur oberen Bildschirmkante beziehungsweise in Richtung der Titelzeile. Android Studio blendet ein Hilfefenster ein, das Sie darüber informiert, dass Sie durch Loslassen der Maustaste ein Constraint anlegen können. Verfahren Sie danach mit der linken Kante nach demselben Schema.

In einem ConstraintLayout befindliche Steuerelemente müssen mindestens zwei Constraints aufweisen. Eines muss dabei die vertikale, eines die horizontale Positionierung beschreiben.

Dieses Kriterium ist insofern schwierig zu handhaben, als es erst zur Laufzeit umgesetzt wird. Die Renderingansicht von Android Studio zeigt die Steuerelemente am Ablageort an, um dem Entwickler das einfachere Anliegen von Constraints zu ermöglichen. Die Konsequenzen von vergessenen Beziehungen zeigen sich zur Laufzeit: Die betroffenen Steuerelemente schweben dann am linken oberen Bildschirmrand.

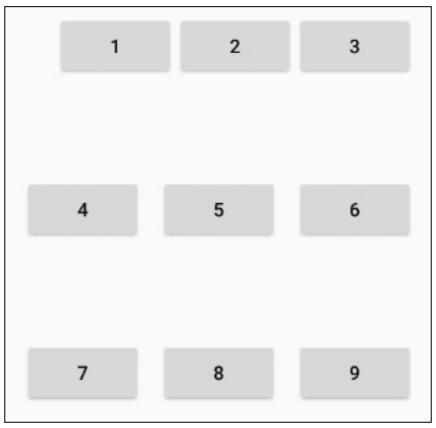
Der Editor zum Anlegen neuer Constraints ist zeitweise etwas sperrig. Ein Weg zum Prüfen des Layouts ist das rote



Lint findet auch Fehler in Layoutdateien (Bild 9)

Quadrat an der oberen rechten Bildschirmkante. Klicken Sie es an, um die in Bild 9 gezeigte Fehlerliste auf den Bildschirm zu holen.

Nach diesem kurzen Exkurs ist es an der Zeit, die restlichen Steuerelemente miteinander zu verbinden. Die mittleren Buttons werden sowohl links als auch rechts mit ihren Nachbarn verbunden. Berücksichtigen Sie dabei allerdings auch, dass ConstraintLayouts keine rekursiven Beziehungen abbilden können.



Hier stimmen die Margin-Einstellungen nicht (Bild 10)

Constraint-Beziehungen sind nur teilweise wechselseitig. In der Praxis ist es empfehlenswert, am Rand des Bildschirms befindliche Steuerelemente mit den jeweiligen Bildschirmkanten zu verbinden. Im Fall unseres Zahleneingabepads sollten Sie beispielsweise den Knopf Nummer eins mit links und oben verbinden, während der Knopf drei mit rechts und oben verdrahtet wird.

Anzumerken ist, dass ConstraintLayouts nur extrem schwer in XML zusammengebaut werden können. Das liegt daran, dass Google ein gutes Dutzend verschiedene Attributnamen spezifiziert. Als Beispiel dafür der Code von Knopf Nummer drei:

```
<Button
  android:id="@+id/Cmd3"
  app:layout_constraintRight_toRightOf="parent"
  app:layout_constraintBottom_toTopOf="@+id/Cmd6"
  app:layout_constraintTop_toTopOf="parent" ... />
```

Die mit *_toRightOf* endenden Deklarationen liegen auch als *_toLeftOf*-Variante vor, während die *_toTopOf*-Varianten auch als *_toBottomOf* angeboten werden. In der Praxis ist es empfehlenswert, die Struktur mit dem Layout-Editor anzulegen und erst später auf das XML zurückzugreifen, um Feineinstellungen vorzunehmen.

Ein weiteres Ärgernis ist der Button *Beziehungen entfernen*, der alle Beziehungen löscht. Wundern Sie sich also nicht, wenn es nach einer Bearbeitung zu seltsam aussehenden Steuerelement-Positionierungen kommt. Einzelne Beziehungen lassen sich zum Zeitpunkt der Drucklegung nur durch Anklicken des Ankerpunkts löschen – dies ist kontraintuitiv.

Immerhin funktioniert Instant Run in diesem Bereich sehr gut. Der Autor testet ConstraintLayouts auf seiner Zweischirmworkstation, indem er einen Emulator auf den Zweitschirm legt und nach jeder Änderung des Layouts einen Instant-Run-Deploymentprozess anstößt.

Im Idealfall sieht das Formular nach getaner Arbeit – also dann, wenn alle notwendigen Constraints am Platz sind – symmetrisch aus. In der Praxis kommt es jedoch beim Platzieren immer wieder zu Abweichungen – Bild 10 zeigt ein Beispiel.

ConstraintLayouts stellen Entwicklern eine Vielzahl von Eigenschaften zur Verfügung, mit denen sie das Darstellungsverhalten an ihre Bedürfnisse anpassen können.

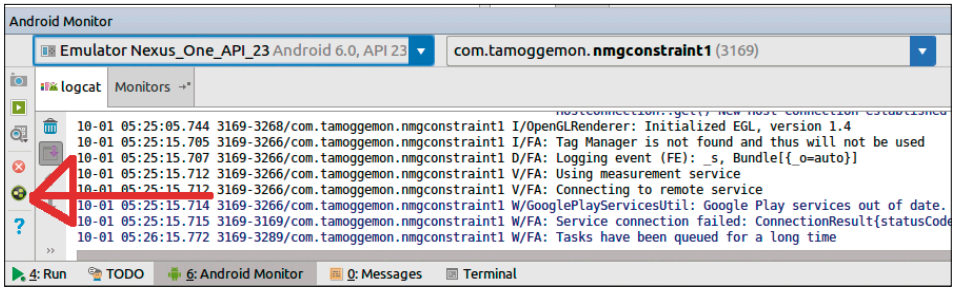
Angesichts der vergleichsweise hohen Komplexität größerer Layouts wollen wir an dieser Stelle ein Werkzeug vorstellen, das mit Android Studio 2.2 erstmals zur Verfügung steht: den Layoutinspektor. Er lässt sich durch Anklicken des in Bild 11 hervorgehobenen Knopfs auf den Bildschirm bringen und zeigt dann detaillierte Informationen über das gerade im Emulator beziehungsweise am Telefon verwendete Layout an. Das ist insbesondere bei der Suche von fehlplatzierten Steuerelementen hilfreich, die am oberen Rand des Bildschirms kleben.

Das einfachste Werkzeug zum Anpassen des Verhaltens eines ConstraintLayouts ist die Nutzung der *Margin*-Eigenschaft. Google spezifiziert hier mehrere Attribute, die den Abstand in Richtung einer bestimmten Achse beschreiben. Als Beispiel dafür hier ein Knopf, der drei derartige Attribute aufweist:

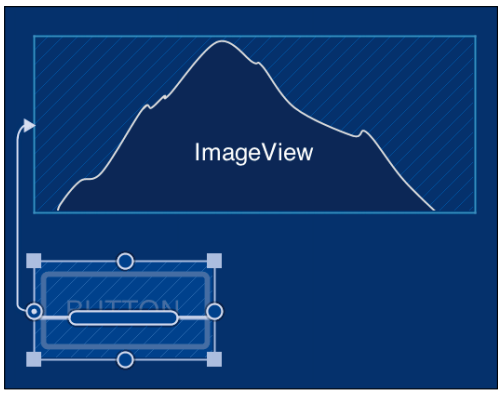
```
<Button
  ...
  android:layout_marginStart="8dp"
  android:layout_marginEnd="8dp"
  android:layout_marginBottom="16dp" />
```

Zur Quantifizierung des Abstands wird ein Wert übergeben, die ein Vielfaches von 8 dp sein sollte. Google weist in der Dokumentation explizit darauf hin, dass nicht durch acht teilbare Werte zu diversen seltsamen Verhalten führen können.

Eine weitere interessante Möglichkeit betrifft die Art der Bindung. Sie sind nicht auf das Anbinden von gegenüberliegenden Kanten beschränkt. Unsere Tabelle zeigt die von Google angebotenen Variationen – positionieren Sie die Steuerelemente bei Bedarf am Bildschirm um, um die betreffende Beziehung herzustellen (Bild 12, Bild 13). ▶



Die Aktivierung des Inspektors ist gut versteckt (Bild 11)



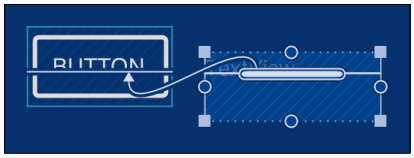
Ecke an Ecke: Die beiden Ecken werden auf einer gemeinsamen Hilfslinie angeordnet (Bild 12)

Mindestens ebenso interessant sind die Hilfslinien. Es handelt sich dabei um für den Benutzer unsichtbare Geraden, die über das Formular gelegt werden und die zum Andocken von Steuerelementen geeignet sind. Der zum Anlegen neuer Hilfslinien vorgesehene Knopf findet sich in der Toolbar des Ressourceneditors und ist etwas versteckt – Bild 14 zeigt ihn samt dem dazugehörigen Kontextmenü.

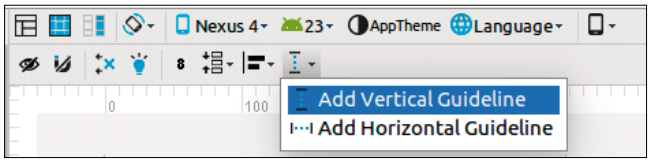
Die Positionierung der Führungslinien im Formular kann entweder in dp oder durch Prozentangaben erfolgen. Die Umstellung der verschiedenen Positionierungsmodi erfolgt normalerweise über den Kreis, der außerhalb des Steuerelements angezeigt wird. Leider verschwindet er in der Previewversion vom Bildschirm. Ein Weg, um dieses Verhalten zu umgehen, besteht darin, den Mauszeiger auf seinen Platz zu legen und einfach draufloszuklicken. Alternativ können Sie den gewünschten Rendering-Modus auch durch XML festlegen (Listing 1).

Wir wollen dies im nächsten Schritt nutzen, um die Knöpfe ein wenig nach unten zu schieben. Dazu ist eine horizontale Guideline notwendig, die per Drag and Drop eine Höhe von 33 Prozent eingeschrieben bekommt. Das Anklicken von Guidelines im Ressourceneditor beziehungsweise in der Blaupause ist wegen ihrer geringen Dimensionen schwierig. Erfreulicherweise erscheinen die Steuerelemente auch in der Baumansicht.

Dank der Flexibilität des ConstraintLayouts bedeutet das eigentliche Anpassen der Buttonreihen keinen großen Aufwand. Klicken Sie in den Header des ersten, des zweiten und des dritten Buttons und ziehen Sie den oberen Anker statt auf den oberen Rand des Displays auf die neu errichtete Hilfslinie. Die restlichen Steuerelemente wandern daraufhin automatisch mit ihrem Elterelement mit.



Baseline: richtet das Steuerelement an der Textbasis des anderen Steuerelements aus (Bild 13)



ConstraintLayouts kennen sowohl horizontale als auch vertikale Hilfslinien (Bild 14)

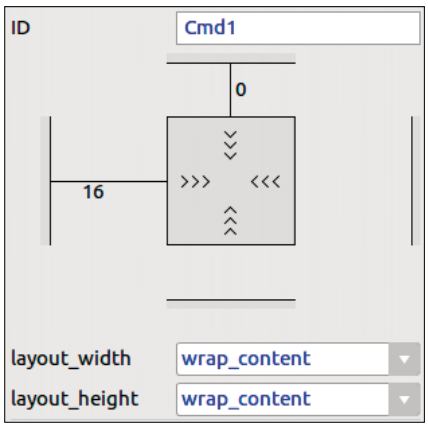
Kreative Größenanpassung

Damit sind wir zum Einpflegen eines TextView-Steuerelements bereit. Hängen Sie es einfach wie vorher den Knopf Nummer eins an der oberen Bildschirmkante und an der linken Bildschirmkante an – ConstraintLayout erledigt den Rest.

Es wäre in der Praxis wünschenswert, dass das Steuerelement mitwachsen würde. Dieses in Qt über das Konzept der Gewichte gelöste Problem wird in Android Studio auf eine ähnliche Art und Weise abgearbeitet. Wer den reduzierten Properties-Dialog aktiviert und ein Steuerelement in der Blaupausen-oder Render-Ansicht selektiert, sieht das in Bild 15 gezeigten Einstellungswerkzeug.

Listing 1: Rendering-Modus

```
// Prozentuale Positionierung: Hilfslinie ist an
// einem bestimmten Punkt des Bildschirms

<android.support.constraint.Guideline
  android:layout_height="0dp"
  android:id="@+id/guideline3"
  android:orientation="horizontal"
  tools:layout_editor_absoluteY="169dp"
  tools:layout_editor_absoluteX="0dp"
  android:layout_width="384dp"
  app:layout_constraintGuide_percent="0.33" />

// DP-Positionierung nach oben beziehungsweise
// links: Hilfslinie schwebt in bestimmtem Abstand
// vom linken beziehungsweise oberen Anker
<android.support.constraint.Guideline
  ...android:layout_height="0dp"
  android:id="@+id/guideline3"
  android:orientation="horizontal"
  tools:layout_editor_absoluteY="169dp"
  tools:layout_editor_absoluteX="0dp"
  android:layout_width="384dp"
  app:layout_constraintGuide_begin="169dp" />
// DP-Positionierung nach unten beziehungsweise
// rechts: Hilfslinie schwebt in bestimmtem Abstand
// vom unteren beziehungsweise rechten Anker
<android.support.constraint.Guideline
  android:layout_height="0dp"
  android:id="@+id/guideline3"
  android:orientation="horizontal"
  tools:layout_editor_absoluteY="169dp"
  tools:layout_editor_absoluteX="0dp"
  android:layout_width="384dp"
  app:layout_constraintGuide_end="342dp" />
```



Alle wichtigen Layout-Einstellungen sind hier übersichtlich zusammengefasst (Bild 15)

Das Anpassungsverhalten in die jeweilige Richtung lässt sich durch Anklicken der Symbole bearbeiten. Die drei nicht verbundenen Pfeile stehen dabei für automatische Größenanpassung. Das Steuerelement ist in diesen Betriebsmodus so groß, wie es für die Anzeige der Inhalte notwendig ist.

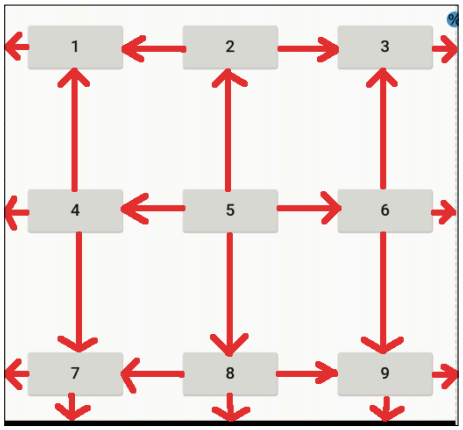
Eine unterbrochene Linie gibt an, dass die Größe ausschließlich durch die Konferenz festgelegt wird. Wer sich stattdessen für die gerade Linie entscheidet, stellt den zur Laufzeit nur programmatisch veränderbaren Wert in den beiden darunter eingeblendeten Textboxen ein.

Äquidistante Verteilung

Als nächste Aufgabe wollen wir eine äquidistante Verteilung erreichen. Dazu müssen wir die Ausrichtung der einzelnen Constraints anpassen. Bild 16 zeigt eine sinnvolle Vorgehensweise.

Wer das Programm in der vorliegenden Form ausführt, bemerkt, dass die Steuerelemente in der Mitte des ihnen zugewiesenen Bereichs schweben. Die Abstandszuteilung erfolgt dabei nach dem Prinzip der Fairness. Es werden im ersten Schritt die Margins abgezogen. Bleibt danach noch Platz frei beziehungsweise sind keine Margins vorhanden, so erfolgt die Zuteilung nach dem Prinzip der brüderlichen Teilung.

Diese Vorgehensweise ist in der Praxis nicht immer wünschenswert. Weist ein Steuerelement in beide Richtungen ausgehende Constraints auf, so blendet Android Studio den in Bild 17 gezeigten Slider ein. Das Anpassen des von 0 bis 100 laufenden Werts erlaubt das Festlegen eines Gewichts zwischen den beiden Achsen: Liegt der Wert beispielsweise auf 30, so schwebt das Steuerelement eher oben beziehungsweise eher links.

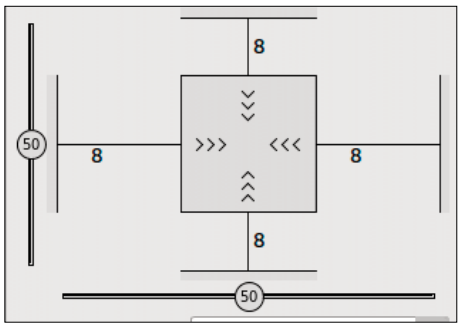


Die roten Pfeile beschreiben die Richtung der Beziehungen der Steuerelemente (Bild 16)

Nach diesen zugegebenermaßen langwierigen Arbeiten mit ConstraintLayouts ist es an der Zeit, sich einem weiteren interessanten Thema zuzuwenden: Unit Tests sind bei Entwicklern vor allem deshalb unbeliebt, weil ihre Erzeugung viel Zeit in Anspruch nimmt. Bei der Arbeit mit grafischen Frameworks wie Espresso ist der Aufwand noch größer.

Erstellung grafischer Testfälle

Android Studio 2.2 bringt ein von Systemen wie Selenium bekanntes Werkzeug mit, das die Erstellung grafischer Testfälle erleichtert. Zum Testen dieses Features müssen wir unser Eingabefeld im ersten Schritt um etwas Intelligenz erweitern. Verdrahten Sie die Knöpfe 1 bis 9 mit nach folgendem Schema aufgebauten Event Handlern, um Texte in die jeweiligen TextViews zu schreiben:



Die Slider scheinen nur dann auf, wenn die Constraint-umgebung die Berücksichtigung von Gewichten ermöglicht (Bild 17)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  ...
  ((Button)findViewById(R.id.Cmd1)).
  setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      addText("1");
    }
  });
}
```

An dieser Stelle ist die Art der Errichtung des Event Handlers relevant. Indem Sie das Anlegen einer Membervariable umgehen, ersparen Sie sich das mühevolle Auseinanderdividieren der Ereignisse in *onClick*.

Das eigentliche Anhängen der anzulegenden Zahl erfolgt sodann in einer Hilfsfunktion, deren Code folgendermaßen aussieht:

```
public void addText(String _what) {
  TextView myView=(TextView)findViewById
  (R.id.textView);
  String jetztWert=myView.getText().toString();
```

```
  jetztWert=jetztWert + _what;
  myView.setText(jetztWert);
}
```

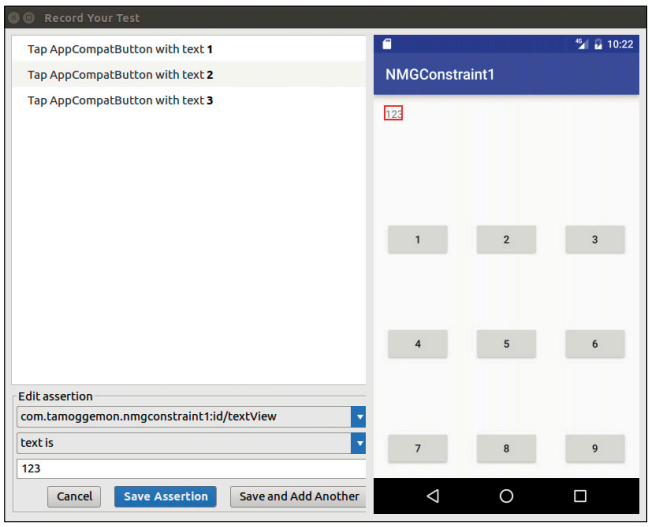
Wir haben die Bedienung von Espresso in der Ausgabe 11/2016 von **web & mobile developer** vorgestellt. Die dort präsentierten Hinweise (Stichwort Deaktivieren der Animationen) gelten auch dann, wenn ein Test mit Automatisationsunterstützung aufgezeichnet werden soll.

Der Assistent zum Aufnehmen von Testfällen lässt sich durch Anklicken von *Run, Record Espresso Test* aktivieren. Android Studio beginnt daraufhin mit einem normalen Launch der APK-Datei. Diese Vorgehensweise ist insofern sinnvoll, als jeder Espresso-Test aus dem Neutralzustand heraus erfolgt. Im Rahmen des Startprozesses der Applikation blendet Android Studio einen mit *Record your Test* betitelten Dialog ein, in dem die diversen Ereignisse aufgezeichnet werden.

Testbedingungen lassen sich grafisch anlegen. Klicken Sie dazu auf den Button *Add Assertion*. Android Studio reagiert darauf mit dem Anzeigen des in Bild 18 abgedruckten Dialogs, der eine Liste aller gerade am Bildschirm befindlichen Views samt den dazugehörenden Inhalten beziehungsweise Attributen enthält. Nach dem erfolgreichen Zusammenstellen der Bedingungen und der abzuarbeitenden Klicks fehlt nur noch der Button *Complete Recording*. Der Editor fragt sodann nach dem Namen der anzulegenden Klasse – die eigentliche Codegenerierung erfolgt automatisch.

In diesem Zusammenhang ist der Aufbau des erzeugten Codes interessant – Interaktionen mit Steuerelementen erfolgen über die Klasse *ViewInteraction*. Als Beispiel dafür der Code, der für das Anklicken des 1-Buttons zuständig ist:

```
@Test
public void nMGTestfall1() {
  ViewInteraction appCompatButton = onView(
    allOf(withId(R.id.Cmd1), withText("1"),
```



Wer Constraints mit *Record your Test* zusammenbaut, spart jede Menge Zeit (Bild 18)

```
    withParent(allOf(withId(R.id.activity_main),
    withParent(withId(android.R.id.content)))),
    isDisplayed()));
  appCompatButton.perform(click());
```

Dieselbe Klasse kommt auch beim Einlesen der in der Textbox befindlichen Inhalte zum Einsatz:

```
ViewInteraction textView = onView(allOf(withId(R.id.
textView), withText("123"),
  childAtPosition(
    allOf(withId(R.id.activity_main),
      childAtPosition(
        withId(android.R.id.content),
      0)),
    0),
  isDisplayed()));
textView.check(matches(withText("123")));
```

Googles Codegenerator unterscheidet sich von von Hand geschriebenen Testfällen insofern, als er vergleichsweise stringente Überprüfungen anlegt. Dies ist beim manuellen Anpassen der Cases ärgerlich – der Autor empfiehlt, den Testfall bei größeren Änderungen in der Applikationsstruktur einfach neu anzulegen.

Einbindung von Firebase

Microsofts SQL-Server verdankt seinen Erfolg – zumindest bis zu einem gewissen Grad – der großartigen Integration zwischen dem Produkt und Visual Studio. Wer einmal eine Datenbankapplikation auf diese Art und Weise zusammengebaut hat, codiert nur noch ungern von Hand. Azure und Co. bauen diese Kommunalität weiter aus, um Entwickler im Ökosystem zu halten.

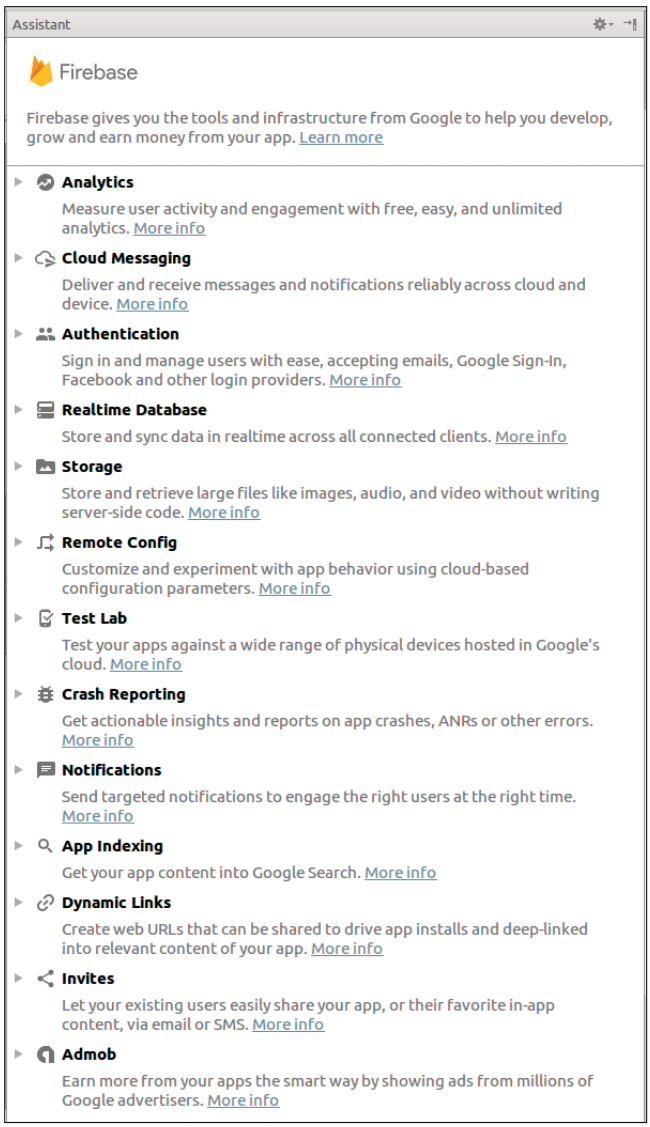
Google kaufte im Jahr 2014 Firebase, um seinen Klienten eine bessere Datenbank-Arbeitsumgebung anbieten zu können. Android Studio 2.2 bindet diese nun über ein vergleichsweise umfangreiches Plug-in ein, das Entwicklern diverse Cloud-Features bequem zur Verfügung stellt.

Wie im Fall des ConstraintLayouts gilt auch hier, dass Google die betreffenden Module nicht automatisch mit Android Studio ausliefert. Stattdessen ist etwas Handarbeit erforderlich. Klicken Sie auf *Tools, Android, SDK Manager*, wechseln Sie in die Rubrik *SDK Tools* und installieren Sie das Google-Repository.

Nach dem erfolgreichen Herunterladen der diversen Komponenten steht unter *Tools, Firebase* ein Assistent zur Verfügung, der nach der Aktivierung den in Bild 19 gezeigten Feature-Listendialog einblendet.

Wir wollen an dieser Stelle die in Firebase integrierte Analytics-Lösung antesten. Öffnen Sie dazu *MainActivity.java* und platzieren Sie den Cursor in einen der Knopf-Event-Handler. Klicken Sie danach auf den Pfeil neben der Rubrik *Analytics*, um den Link *Log an Analytics event* einzublenden.

Android Studio reagiert darauf mit dem Einblenden einer Anweisungsliste. Der erste Task verbindet die IDE mit einem Google-Konto, das für den Zugriff auf die Firebase-Daten zu-



FireBase dient als Sammelpunkt für diverse Funktionen (Bild 19)

ständig sein soll. Nach der im Browser erfolgenden Anmeldung präsentiert die IDE einen Dialog, in dem Sie ein neues Firebase-Projekt anlegen können.

Schritt zwei ist das Anklicken des Buttons *Add Analytics to your App*. Android Studio blendet daraufhin ein Fenster ein, das Sie über die in den diversen *build.gradle*-Dateien erfolgenden Änderungen informiert. Beachten Sie dabei, dass das Einpflegen des angezeigten Codes in die Activity von Hand erledigt werden muss.

Zum fertigen Logging-Programm fehlt dann nur noch das Einbinden folgendes Snippets in die diversen Event Handler – der Inhalt des Bundles kann je nach den Bedürfnissen ihrer Applikation mehr oder weniger umfangreich ausfallen:

```
Bundle bundle = new Bundle();
bundle.putString(FirebaseAnalytics.Param.ITEM_ID, id);
bundle.putString(FirebaseAnalytics.Param.ITEM_NAME,
name);
bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE,
"image");
mFirebaseAnalytics.logEvent
(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
```

Ganz gleich ob Symbian oder Bada: Das Ausliefern von Programmen auf Endgeräte ist seit jeher ein Ärgernis, das viel Zeit in Anspruch nimmt. Das mit Android Studio 2.0 erstmals an Entwickler ausgelieferte Instant-Run-Feature sollte an dieser Stelle Abhilfe schaffen.

Die dahinterstehende Idee ist einfach. Es handelt sich um eine Erweiterung der IDE, die auf einem kompatiblen Telefon Änderungen am Applikationscode ohne ein komplettes Redeployment des APK übernimmt.

In der Praxis funktioniert dieser diff-Prozess meist eher schlecht als recht. Viele Entwickler – der Autor dieses Artikels zählt dazu – deaktivierten es, weil die dadurch verursachten Fehler den durch das schnellere Deployment eingespielten Zeitgewinn in vielen Fällen auffraßen.

Google verspricht, dass Android Studio 2.2 – allerdings nur im Zusammenspiel mit komplett aktualisierten Plattform-Tools – an dieser Stelle wesentlich zuverlässiger arbeitet. In Versuchen des Autors zeigte sich die neue Version von Instant Run in der Praxis als brauchbar. Es handelt sich hierbei also um ein Feature, dem man zumindestens noch einmal eine Chance geben sollte.

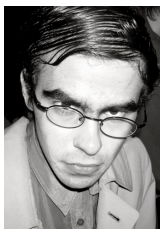
Ein weiteres Feature zur Beschleunigung des Entwicklungsprozesses ist ein globaler Build-Cache, der kompilierte Artefakte – auch zwischen Projekten – aufbewahrt. Das im Moment noch in der Betaphase befindliche Feature lässt sich durch Öffnen der Datei *gradle.properties* anpassen. Ergänzen Sie sie einfach um die folgende Passage, um den Cache zu aktivieren:

```
android.enableBuildCache=true
```

Berücksichtigen Sie, dass Google dieses Feature in der hauseigenen Dokumentation explizit als unausgereift bezeichnet – es ist durchaus möglich, dass es bei der Nutzung zu diversen seltsamen Fehlern kommt.

Fazit

Android Studio ist – unter anderem dank des im Hintergrund arbeitenden IntelliJ – seit langer Zeit eine durchaus ausgereifte IDE. Mit der neuen Version 2.2 betreibt Google erfolgreiche Produktpflege. Die neuen Features sparen an der ein oder anderen Stelle Mannminuten, ohne dabei unangenehm aufzufallen. ■



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

OBJEKTORIENTIERTE PROGRAMMIERUNG MIT PHP

Stichtag

Wie man im objektorientierten Umfeld Probleme modelliert und in PHP implementiert.

Mit fertigen Bibliotheken zu arbeiten, um bei der Auftragsbewältigung konkurrenzfähig zu bleiben, ist legitim und durchaus üblich. Wenn allerdings für das vorhandene Problem noch keine Lösung existiert, besteht die Notwendigkeit, die Aufgabe in Eigenregie zu bewältigen.

Wie bei einer solchen Herausforderung eine mögliche Lösung aussehen kann, zeigt der folgende Workshop zu einem einfachen Kalender. Da die Funktionsweise von Kalendern allgemein bekannt ist und die Fachlichkeit hinreichend komplex ist, habe ich meine Wahl diesbezüglich getroffen.

Planspiele

Nicht nur in großen Projekten steht eine ausführliche Planung der umzusetzenden Schritte am Anfang aller Aktivitäten. Auch kleinen Vorhaben eröffnet eine gründliche Überlegung mehr Gestaltungsmöglichkeiten, als ein reines Drauflosprogrammieren bieten kann. Um bei unserem Beispiel die Kirche im Dorf zu lassen, reduzieren wir den Planungsaufwand auf das Notwendigste. Die Beschreibung der Fachlichkeit ist in unserem Fall eine Aufzählung aller Funktionen, die als Resultat für uns eine überschaubare Liste produziert hat:

- tabellarische Darstellung eines Kalendermonats,
- der erste Tag eines Monats beginnt eingerückt zum entsprechenden Wochentag,
- Erkennen von Schaltjahren,
- Sonntage sollen hervorgehoben werden, zum Beispiel rot eingefärbt,
- Hervorheben von statischen Feiertagen wie Weihnachten oder 1. Mai,
- Berechnung des Ostersonntags (zur Anzeige dynamischer Feiertage),
- Anzeige des aktuellen Datums zum Beispiel durch Einrahmen,
- Anzeige des Monatsnamens,
- Anzeige des Kalenderjahrs (4-stellig).

Neben der reinen Funktionalität existieren zusätzliche, sogenannte nichtfunktionale Anforderungen, welche ebenfalls berücksichtigt werden müssen. Zu dieser Kategorie zählen üblicherweise Vereinbarungen über Performance, Qualität, Sicherheit und Benutzbarkeit.

Für den Kalender lässt sich das wie folgt formulieren: Die Software soll größtmögliche Flexibilität im Einsatz aufweisen, wobei die Komponente möglichst einfach zu handhaben sein soll. Es sind aktuelle Standards der Programmierung zur Gewährleistung der Applikationssicherheit einzuhalten. Wie diese speziellen Forderungen eingehalten werden können, zeigen die folgenden Abschnitte.

Resultat: So sieht der fertige Kalender aus (Bild 1)

In Bild 1 soll das zu erreichende Resultat schon einmal vorweggenommen werden. Die vollständigen Sourcen sind auf GitHub unter GPL zur freien Verwendung veröffentlicht. Das Setup der Entwicklungsumgebung ist Windows, NetBeans IDE, Git und XAMPP mit der PHP-Version 5.5.

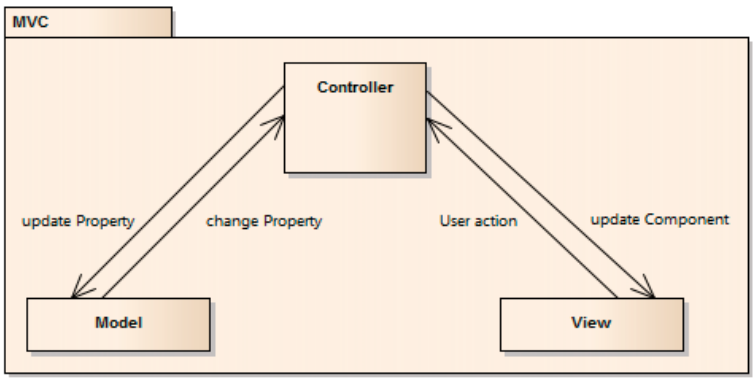
Eine Datenbank ist nicht erforderlich. Bevor ich einige Worte zum objektorientierten Paradigma (OOP) äußere, erläutere ich das Grundkonzept der Architektur. Dabei kommt das Muster Model-View-Controller (MVC) zur Verwendung, das in der Software-Entwicklung eine wichtige Position einnimmt.

Puzzleteile

Das wichtigste Ziel bei der professionellen Software-Entwicklung ist die Wiederverwendung von entwickelten Komponenten. Eine solche Wiederverwendung erreicht man vornehmlich durch sauberes Abgrenzen (Kapseln) der Verantwortlichkeiten. Um dies zu erzielen, bedienen wir uns des etablierten Musters (Design Patterns) MVC. In der Fachliteratur wird dieses Pattern meist als Architekturmuster deklariert, aber auch die Einordnung als Entwurfsmuster ist recht häufig anzutreffen.

Meiner Ansicht nach ist MVC bevorzugt als Architekturmuster zu katalogisieren, da es einen strukturierenden Charakter besitzt. In MVC werden Daten (Model) vom Layout (Design/Darstellung) der sogenannten View getrennt und über einen Controller verbunden. Das Ganze lässt sich wie folgt beschreiben: Der Controller befüllt die Datenobjekte, um diese dann in der View darzustellen. Ebenfalls zum Model zählt die Geschäftslogik, die in unserem Fall die Datumsberechnungen wie Schaltjahr oder Ostern sind. Bild 2 zeigt ein vereinfachtes UML-Diagramm zu MVC.

Der größte Mehrwert dieser Vorgehensweise ist die Möglichkeit, die View ohne viel Aufwand auszutauschen oder mehrere Views parallel zu betreiben. Mobile, Desktop und Browser könnten beispielsweise verschiedene Views sein. Aber auch Projektor oder Print sind als Views vorstellbar. Da dieses kleine Projekt ohne Datenbank auskommt, ist die Erstellung von Datenobjekten hinfällig.



MVC: Ein vereinfachtes UML-Diagramm zu MVC (Bild 2)

Eine spätere Erweiterung könnte das Abspeichern verschiedener Events sein. Das Model eines Events (Datenobjekt/Domainobjekt) könnte aus Eventname (Index), Datum, Uhrzeit, Ort und Beschreibung bestehen. Wie man sieht, lassen sich auf diese Weise mit überschaubarem Aufwand Anpassungen vornehmen. Damit erlauben wir auf längere Sicht eine Wiederverwendung unseres Projekts, ohne gleich zu Beginn sämtliche Möglichkeiten umzusetzen, und die Kosten einer Implementierung lassen sich gerechter zwischen den Auftraggebern verteilen.

Hexenküche

Ein Grund, weshalb PHP für den professionellen Einsatz geeignet ist, ist die Möglichkeit der objektorientierten Programmierung. Eine tiefergehende Behandlung dieser Thematik lässt die Kompaktheit dieses Artikels leider nicht zu, weshalb ich auf weiterführende Literatur verweisen möchte. Um die Codebeispiele nachvollziehen zu können, gehe ich an dieser Stelle kurz auf die verwendete Syntax ein. Die wichtigste Empfehlung zum Umgang mit Objekten ist den Java Coding Styles entnommen, die sich in unzähligen Projekten seit vielen Jahren als äußerst hilfreich bewährt haben. Die vereinfachten Regeln lauten wie folgt:

- Eine Datei darf nicht mehr als eine Klassendefinition enthalten.
- Der Name der Datei ist identisch mit der Klassendefinition.
- Ein Klassenname beginnt immer mit einem Großbuchstaben (CamelCase/Wiki Style).
- In Dateien, in denen Klassen definiert werden, sollte OOP nicht mit funktionaler Programmierung vermischt werden.

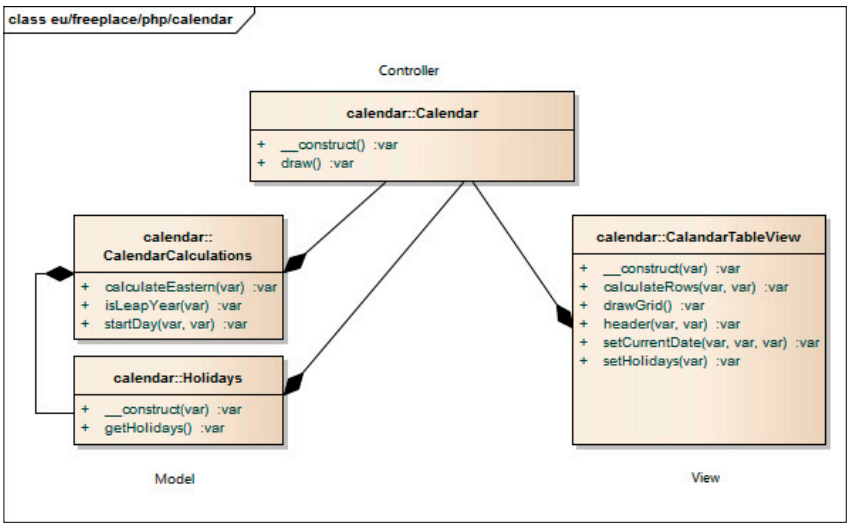
Das Erzeugen von konkreten Objekten (Instanzieren) erfolgt über den Operator *new*. Eine andere Bezeichnung für Objekt ist Klasse. Klassen können Eigenschaften von einer Elternklasse erben. Dieses Verhalten erlaubt die Spezialisierung der Kindklassen gegenüber der (generalisierten) Elternklasse. Eine Kindklasse kann nur eine Elternklasse besitzen. Eine Mehrfachvererbung ist also nicht möglich. Das Schlüsselwort für Vererbung lautet *extend*.

Weitere wichtige Relationen zwischen zwei Klassen/Objekten sind Aggregation und Komposition. Eine Aggregation bedeutet, dass Klasse A eine Klasse B aufruft und diese Klasse B so lange existent ist wie Klasse A selbst. Das hat zur Folge, dass Klasse B nicht ohne Klasse A existieren kann und gemeinsam mit Klasse A wieder freigegeben wird. Aggregationen sind Spezialfälle der *hat ein*-Beziehung – etwa: Ein Magazin hat n Abbonenten. Eine Komposition hingegen gestattet es, Klasse A von Klasse B zu entkoppeln. Das bedeutet, dass Klasse B nach dem Freigeben von Klasse A eigenständig weiterexistieren kann.

Ein typisches Beispiel für Aggregation ist, dass ein Objekt *Kurs* nicht ohne das Objekt *Schule* existieren kann. Eine Komposition von Objekten kann für ein *Auto* das Objekt *Fahrwerk* und das Objekt *Motor* sein, die sich in verschiedenen Modellen kombinieren lassen. Es ist nicht entscheidend, welches die natürliche Relation zwischen Objekten ist, sondern der in der Architektur festgelegten Verknüpfung ist stets Folge zu leisten. Natürlich sollten solche Entscheidungen immer begründet werden können. Eine umfassende Übersicht zwischen Objektrelationen und deren Darstel- ▶

Listing 1: Definitionen

```
01: class Object {
02:   const KLASSEN_KONSTANTE = "Never change me.";
03: }
04:
05: class ConcretObject extends Object {
06:   private $variable;
07:
08:   function __construct() {
09:     $this->variable = "";
10:     echo "Konstruktor ConcretObject";
11:   }
12:
13:   public function ausgabe() {
14:     echo "Methode ausgabe() Attribut: " .
        $this->variable;
15:   }
16:
17:   public function getVariable() {
18:     return $this->variable;
19:   }
20:
21:   public function setVariable($variable) {
22:     $this->variable = $variable;
23:   }
24:
25: }
26:
27: $klasse = new ConcretObject();
28: $klasse->setVariable("wert");
29: $klasse->ausgabe();
30:
31: echo ConcretObject::KLASSEN_KONSTANTE;
```



Die Struktur des vollständigen Klassendiagramms (Bild 3)

lung in UML findet sich in Wikipedia unter https://en.wikipedia.org/wiki/Class_diagram. In Listing 1 werden die vorangegangenen Definitionen etwas genauer beleuchtet. Zeile 1 definiert die Klasse *Object* mit der Konstante *KLASSEN_KONSTANTE*. Die Klasse *ConcretObject,* die von *Object* abgeleitet ist und alle Eigenschaften von *Object* erbt, wird in Zeile 5 definiert. Anschließend wird eine Variable deklariert, die in der gesamten Klasse sichtbar ist. Das Schlüsselwort *private* verhindert einen direkten Zugriff auf die Klassenvariable. Diese globalen Klassenvariablen werden im objektorientierten Umfeld auch als Attribut bezeichnet, da es sich im semantischen Kontext um eine Eigenschaft des Objekts handelt. Das Attribut ist noch nicht mit einem Wert initialisiert, dies geschieht über den Konstruktor, der ab Zeile 8 beginnt.

Der Konstruktor wird beim Instanzieren eines neuen Objekts automatisch ausgeführt. Über den in Zeile 17 definierten Getter und den in Zeile 21 definierten Setter kann auf *$variable* außerhalb der Klasse zugegriffen werden. Das Instanzieren von *ConcretObject* durch das Schlüsselwort *new* zeigt die Zeile 27. Den Vererbungsmechanismus am Beispiel der *KLASSEN_KONSTANTE* demonstriert abschließend Zeile 31, wobei der ::-Operator den statischen Zugriff auf die Konstante erlaubt.

Objektpuzzle

Kommen wir nun zu den konkreten Bestandteilen des Kalenders, der aus vier Objekten besteht. Die Klasse *Calendar* ist der Controller und zugleich der Programmeinstiegspunkt. Wie der Name schon verrät, handelt es sich bei *CalendarTableView* um die View, die für die Darstellung verantwortlich ist. *CalendarCalculation* wiederum enthält unsere Businesslogik, was, wie bereits erwähnt, auch zum Model zählt. Das Model umfasst zudem die Klasse *Holidays*, die eine Datenstruktur mit den entsprechenden Feiertagen beinhaltet.

Die Verknüpfung zwischen der TableView und der Kalkulation im Controller ist gleichberechtigt. Die Struktur des vollständigen Klassendiagramms kann man Bild 3 entnehmen. Die Verknüpfungen geben dabei Aufschluss, welche Objekte in einer Hauptklasse Verwendung finden. So nutzt *Holidays* beispielsweise die *Calculations*-Klasse, um über die Methode *calculateEastern()* den dynamischen Feiertag Karfreitag zu ermitteln und in der Datenstruktur abzulegen.

Eine direkte Verbindung zwischen Model und View ist nicht vorhanden, was auf eine gelungene Trennung der jeweiligen Bereiche hinweist.

Die Verwendung des Kalenders ist entsprechend der Anforderung unkompliziert. Drei Codezeilen genügen, um den Kalender in ein beliebiges Projekt zu integrieren:

```
include './Calendar.php';
$kalender = new \eu\freeplace\php\
calendar\Calendar();
$kalender->draw();
```

Die Verwendung von Namespaces stellt sicher, dass es zu keinen Namenskonflikten mit bereits vorhandenen Klassen kommt. Um eine einheitliche und übersichtliche Klassendefinition zu etablieren, folgen dem Namespace sämtliche importierte Klassen mit dem Statement *include_once*. Im Unterschied zu einem einfachen *include* wird so dem *redeclare*-Fehler vorgebeugt, der entsteht, wenn man versucht, Codefragmente mehrfach einzubinden. Zusätzlich entsteht so ein Katalog der inkludierten Klassen.

Schreibstube

Nun ist es an der Zeit, unser Klassenensemble mit Leben zu erfüllen. Die am leichtesten zu lösende Aufgabe ist das Ermitteln des aktuellen Datums. Genauer gesagt gibt die Funktion *getDate()* das Datum zurück, das auf dem Server eingestell ist, auf dem unser Skript läuft. Unter der Annahme, dass der Server korrekt auf unsere Bedürfnisse konfiguriert ist, soll uns

Tabelle 1: Auszug der Datumsformatierungen

Formatierung	Beschreibung	Beispiel
N	Numerische Repräsentation des Wochentags. 1 = Montag, 7 = Sonntag	1–7
mday	Numerische Darstellung des Tages, ohne führende Null	1–31
mon	Numerische Darstellung des Monats, ohne führende Null	1–12
year	4-stellige Darstellung des Jahres	2016
z	Numerische Darstellung des Tages im Jahr	0–365
W	Wochennummer im Jahr	42

dieser Ansatz als Basis für unser weiteres Vorgehen dienen. Tabelle 1 zeigt eine Übersicht der verwendeten Formatierungen für das Datum. Eine vollständige Übersicht dazu findet sich in der PHP-Dokumentation. Da das aktuelle Datum für uns eine zentrale Rolle einnimmt und für viele Berechnungen benötigt wird, platzieren wir dessen Ermittlung an den Programmeinstieg.

Andere wichtige Informationen zum aktuellen Datum sind Erkenntnisse über die Anzahl der Tage des Monats, den entsprechenden Wochentag und Ähnliches. All diese Informationen werden über den Konstruktor initialisiert. Die Monatsnamen und Wochentage sind über die Arrays *$months* und *$days* hinterlegt. Über dieses Vorgehen ist unser Programm weitgehend flexibel und kann später mit einer Option zur Internationalisierung erweitert werden.

Die Hilfsmethode *setMonthName()* passt den numerischen Wert des Monats an den Array-Index an. Die Anzahl der Tage innerhalb der Monate eines Jahres sind weitgehend statisch, mit der Ausnahme für den Februar bei Schaltjahren. Um diesem Umstand gerecht zu werden, ermittelt die Methode *isLeapYear()* in *CalenderCalculation,* ob das übergebene Jahr ein Schaltjahr ist.

Für den Fall, das der Algorithmus das aktuelle Jahr als ein Schaltjahr erkennt, wird der Wert der Anzahl der Tage für Februar im Konstruktor von 28 auf 29 angehoben. Die Feiertage werden ebenfalls über den Konstruktor des Controllers initialisiert und sind über die Klasse *Holidays* implementiert. Der Zugriff erfolgt über die Methode *getHolidays()*.

Feste feiern

Für Feiertage unterscheiden wir zwei Gruppen: dynamische und fixe Feiertage. Feststehende (fixe) Feiertage sind typischerweise Festlichkeiten wie Neujahr, Heiligabend, 1. Mai, et cetera und können demzufolge ohne Weiteres in der Datenstruktur hinterlegt werden. Der wichtigste dynamische Feiertag ist der Karfreitag (Ostern), an dem sich die meisten kirchlichen Feiertage ausrichten.

Die korrekte Ermittlung des Karfreitags ist eine etwas kompliziertere Berechnung, die von dem deutschen Mathematiker Gauß formuliert wurde. Der Algorithmus nach Gauß ist in der Methode *calculateEastern()* umgesetzt, die den Tag und Monat durch ein # getrennt zurückgibt. Dieser kleine Kunstgriff ist notwendig, da Karfreitag auf den März oder April fallen kann. Über die Funktion *explode()* wird anschließend die Zeichenkette wieder in Tag und Monat zerlegt.

Das dreidimensionale Array *$holidays* wird durch die beiden Hilfsmethoden *setStaticHolidays()* und *setDynamicHolidays()* befüllt. Die Dimensionen des Arrays sind weniger kompliziert, als sie im ersten Moment erscheinen. Der erste Parameter beinhaltet den Monat, der zweite Parameter repräsentiert den Tag des Monats und der dritte Parameter enthält die Optionen *0* und *1*. Im Feld *0* wird der Name des Feiertags hinterlegt und im Feld *1* der Typ des Feiertags:

```
// $holidays[monat][tag][option]:
$holidays[1][1][0] = "Neujahr";
$holiday[1][1][1] = 1;
```

Links zum Thema

- Sourcen auf GitHub
  https://github.com/ElmarDott/PHP-Calendar.git
- NetBeans IDE
  https://netbeans.org
- Git
  https://git-scm.com
- XAMPP
  https://www.apachefriends.org
- MVC Design Pattern
  https://de.wikipedia.org/wiki/Model_View_Controller
- Google Java Style Guide
  https://google.github.io/styleguide/javaguide.html
- Dokumentation von *getDate()*
  http://php.net/manual/en/function.date.php
- Objektrelationen
  https://en.wikipedia.org/wiki/Class_diagram

Das hier hinterlegte Datum ist der 1. Januar als Feiertag mit dem Namen *Neujahr* und dem Feiertagstyp *1.*

Neben der Unterscheidung, wie Feiertage ermittelt werden, kann man übrigens auch unterscheiden, um welchen Typ von Feiertag es sich handelt. Eine solche Differenzierung ist dann sinnvoll, wenn man Unterscheidungen zwischen staatlichen, kirchlichen, evangelischen und katholischen Feiertagen wünscht.

Diese Information ist notwendig, da in Deutschland die einzelnen Bundesländer verschiedene Regelungen für arbeitsfreie Tage haben. Die bisherige Implementierung für Feiertage kennt Arbeitsfrei = *1* und Festtag = *2*. Die Datenstruktur lässt sich natürlich flexibel erweitern.

Rahmenwerk

Nachdem der Kalender mit den notwendigen Parametern durch den Konstruktor versorgt wurde, können diese Informationen zum Erzeugen der View herangezogen werden. Die View wird über den Controller in der Methode *draw()* erzeugt.

Um für die erste Woche des Monats die Einrückung zum Wochentag zu ermöglichen (Offset), auf den der Monatsbeginn fällt, müssen die auszulassenden Tage ermittelt werden. Die übernimmt die Methode *startDay()*, die als Parameter den aktuellen Tag und den Wochentag als numerische Darstellung entgegennimmt. Das Ergebnis diese Berechnung ist die Anzahl der zu überspringenden Wochentage für den ersten Tag des Monats.

Die Berechnung erfolgt über eine Restklassenauswertung des aktuellen Tages zur Basis 7. Mit dem Wissen über den Offset und der maximalen Anzahl der Tage des aktuellen Monats kann das Herzstück der View, die Methode *calculateRows()* aufgerufen werden. Der dafür notwendige Code ist in Listing 2 dargestellt. ▶

Listing 2: Methode calculateRows()

```
01: public function calculateRows($start, $end) {
02:   $calendarGrid = null;
03:   $counter = 1;
04:   $space_cnt = 1;
05:   $holiday = $this->holidays;
06:   while ($counter <= $end) {
07:     $calendarGrid .= "\n<tr>";
08:     for ($j = 1; $j <= 7; $j++) {
09:       $style = ""; $class = ""; $title = "";
10:       if (($j == 7) && ($counter < $end) ||
          ($counter == $start)) {
11:         $style .= " color:red;";
12:         $class .= "sundays ";
13:       }
14:       if (isset($holiday[$this->month]
          [$counter][1])) {
15:         if ($holiday[$this->month][$counter]
            [1] == '1') {
16:           $style .= " color:red;";
17:         } elseif ($holiday[$this->month]
            [$counter][1] == '2') {
18:           $style .= " color:blue;";
19:         }
20:         $style .= " font-weight:bold;";
21:         $class .= "holidays ";
22:         $title = $holiday[$this->month]
            [$counter][0];
23:       }
24:       if ($this->day == $counter) {
25:         $style .= " border:1px solid;";
26:         $class .= " currentDay ";
27:       }
28:       if ($style != "") {
29:         $style = " style='" . $style . "'";
30:       }
31:       if ($class != "") {
32:         $class = " class='" . $class . "'";
33:       }
34:       $calendarGrid .= "\n\t<td" . $style .
          $class . ">";
35:       if ($space_cnt < $start) {
36:         $calendarGrid .= " ";
37:         $space_cnt++;
38:       } else if ($counter <= $end) {
39:         $calendarGrid .= $counter;
40:         $counter++;
41:       } else {
42:         $calendarGrid .= " "; }
43:       $calendarGrid .= "</td>";
44:     }//for
45:     $calendarGrid .= "\n</tr>";
46:   }//while
47:   $this->rows = $calendarGrid . "\n";
48: }
```

Die äußere *while*-Schleife ist für die Erzeugung der Zeilen *<tr>* zuständig, währen die innere *for*-Schleife die Spalten *<td>* generiert. Eine Zeile repräsentiert eine Woche mit sieben Tagen, weswegen die innere Schleife nach sieben Durchläufen beendet wird. Für jede Spalte werden drei Sequenzen nacheinander abgearbeitet. Die erste Sequenz, das Formatieren der Feiertage, beginnt bei Zeile 9 und endet mit der Zeile 27. Die erste *if*-Bedingung in Zeile 10 erkennt die Sonntage in allen Variationen. Der Test in Zeile 14 verhindert Fehlermeldungen über nicht vorhandene Indexe, die immer dann auftreten, wenn kein entsprechender Eintrag im Array *$holidays* vorhanden ist.

Dem nachgeordnet werden die verschiedenen Feiertagstypen abgefragt und die entsprechenden Formatierungen zugewiesen. In Zeile 29 wird der aktuelle Wochentag erkannt. Die Formatierungen erfolgen stufenweise, was durch den Operator .= erfolgt. Der Codeblock zwischen den Zeilen 28 und 33 überprüft, ob für die aktuelle Tabellenzelle eine Formatierung vorhanden ist. Wenn dies der Fall ist, werden die CSS-Attribute *class* und *style* eingefügt.

Die äußere *while*-Schleife iteriert, bis alle Tage eines Monats durchlaufen sind. In der Variablen *$counter* werden die Tage in der Zeile 40 hochgezählt um dann in den Tabellenzellen ausgegeben zu werden. Das Inkrementieren erfolgt nur dann, wenn es sich nicht um Leerzellen handelt. Die Funktion *drawGrid()* in der View baut anschließend sämtliche Fragment der Tabelle zusammen.

Nun erkennt man auch den Vorteil, die Klassenattribute vor externem Zugriff zu schützen. Diese innerhalb einer Klasse globalen Variablen können so nicht versehentlich mit ungültigen Werten befüllt werden.

Fazit

Wie man in diesem Workshop sehen konnte, ist Software-Entwicklung kein Hexenwerk. Mit ein wenig Basiswissen und gut überlegtem Vorgehen lassen sich so gut wie alle Probleme bewältigen. Einige Anregungen für Erweiterungen habe ich bereits gegeben.

Andere mögliche Funktionalitäten könnten beispielsweise verschiedene Layouts, Blättern zwischen den Monaten oder die Anzeige der Wochennummern sein.

Ich hoffe, mit diesem kleinen Projekt ein wenig Motivation dafür geschaffen zu haben, eigenständig zu experimentieren und eigene Funktionalitäten umzusetzen. Die Quelldateien unterliegen der GPL, was ihre freie Modifikation und Verwendung in kommerziellen Projekten gestattet. ■

Marco Schulz
studierte an der HS Merseburg Diplominformatik. Sein Arbeitsschwerpunkt liegt in der Automatisierung von Build-Prozessen und dem Softwarekonfigurations-Management. Seit über zehn Jahren entwickelt er auf unterschiedlichen Plattformen Webapplikationen.

Jetzt kostenlos testen!

Das Fachmagazin für IT-Entscheider

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

www.com-magazin.de/gratis

AMAZON ALEXA SKILLS KIT

Mehrstufige Architektur

Amazon spielt mit seinem Alexa genannten Service seit einiger Zeit im Bereich der Spracherkennung mit. Entwickler können nun eigene Dienste einbinden.

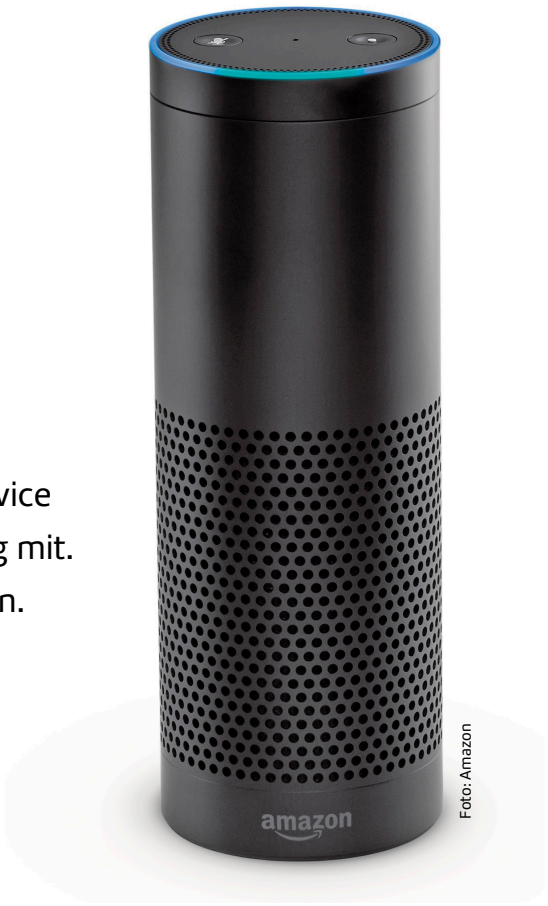


Foto: Amazon

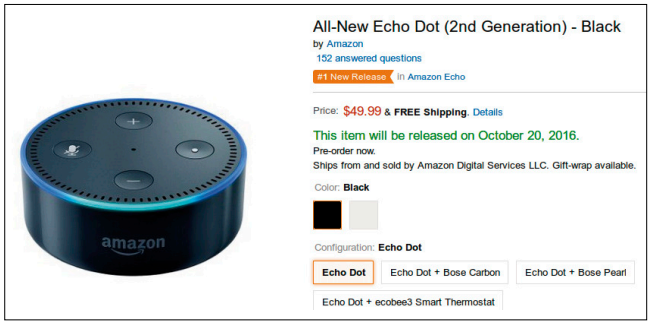
Wer bisher mit Dragon oder klassischen Transkriptionsprogrammen gearbeitet hat, muss bei Alexa umdenken. Die Amerikaner halten ihre Algorithmen auf einem zentralen Server, der von den Endgeräten mit digitalisierten Sprachdaten versorgt wird. Die Auswertung erfolgt in Amazons Rechenzentrum, Resultate wandern in Form von JSON-Resultaten zurück. Diese werden sodann von der Firmware in an die Enduser gerichtete Ausgaben umgewandelt und angezeigt beziehungsweise ausgegeben.

Diese auf den ersten Blick umständlich funktionierende Architektur ist insofern von Vorteil, als Amazon seine Hardware sehr preiswert gestalten kann. Während Sie dieses Heft in Händen halten, beginnt Amazon mit dem Verkauf der zweiten Generation des Echo Dot (Bild 1). Es handelt sich dabei um ein Assistenzsystem, das zum wohlfeilen Preis von 50 US-Dollar angeboten wird.

Programmlogik in der Cloud

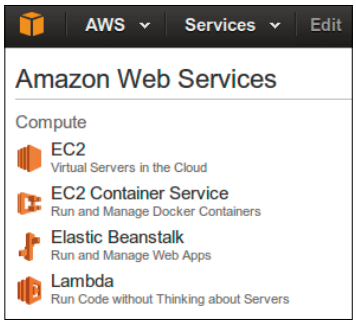
Wer Amazon-Verkaufsrepräsentanten auf Kongressen begegnet, kann der Werbung für AWS nur schwer entgehen. Der einfachste Weg zu einer Erweiterung der Amazon-Alexa-Plattform besteht – wie sollte es auch anders sein – in der Nutzung des AWS-Online-Dienstes.

Dieser bietet mit einem als Lambda bezeichneten Feature seit einiger Zeit die Möglichkeit an, unbürokratisch Programmlogik in die Cloud zu laden.



Dank Client-Server-Architektur sehr preiswert: der Amazon Echo Dot (Bild 1)

Öffnen Sie im ersten Schritt die unter https://aws.amazon.com bereitstehende Startseite von Amazon AWS. Wer noch kein Konto bei Amazon hat, kann sich unter Nutzung einer Kreditkarte anmelden und bekommt ein Jahr lang diverse kostenlose Dienstleistungen zur Verfügung gestellt. Es ist trotzdem ratsam, schon an dieser Stelle ein Kostenlimit festzulegen beziehungsweise eine Prepaid-Kreditkarte zu nutzen. Ein außer Kontrolle geratenes Initialisierungsskript kann sonst einige 100 Instanzen anfordern und immense Kosten verursachen.



Nomen est Omen – auch in der Management-Konsole von Amazon AWS (Bild 2)

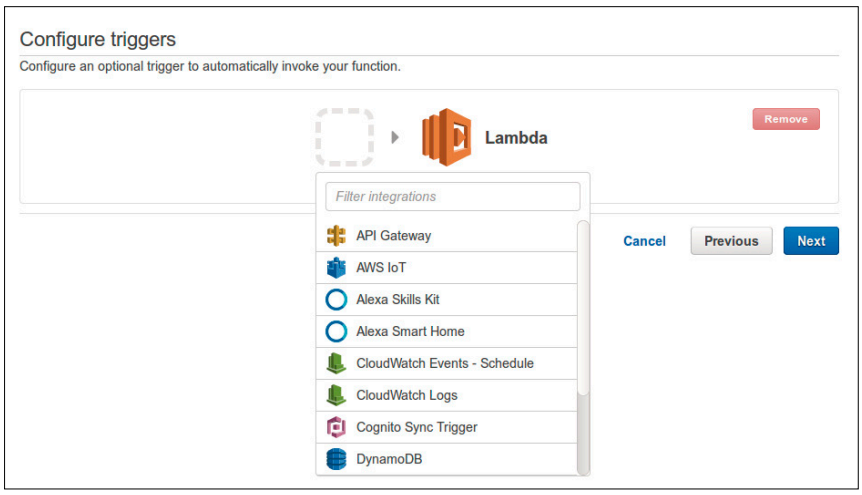
Die eigentliche Verwaltung der diversen Offerten erfolgt in der Management-Konsole, die Sie am einfachsten durch Eingabe des URL https://console.aws.amazon.com/console/home öffnen können. Amazon präsentiert Ihnen daraufhin eine Liste aller angebotenen Dienste. Der von uns benötigte Service findet sich in der Gruppe *Compute* und hört auf den Namen *Lambda* (Bild 2).

Da Alexa im Moment noch in der Testphase ist, legt Amazon Entwicklern eine vergleichsweise seltsame Beschränkung auf: Die zur Realisierung von Skills verwendeten Lambdafunktionen dürfen nur im Rechenzentrum in Nordvirginia beheimatet sein. Dies ist insofern ärgerlich, als die AWS-Steuerkonsole in diesem Bereich unübersichtlich aufgebaut

ist. Das ausgewählte Rechenzentrum wird nicht bei jedem Anlegen einer Ressource neu abgefragt, sondern wird auf die Sitzungsdauer festgelegt. Dazu dient das in Bild 3 hervorgehobene Steuerelement.

Der eigentliche Assistent zum Anlegen einer neuen Funktion fragt im ersten Schritt, ob Sie eine schon vorhandene Blaupause beziehungsweise Vorlage ins Projekt einbinden wollen. Dies ist in unserem Fall nicht notwendig – klicken Sie auf den *Skip*-Knopf, um zum zweiten Schritt zu gelangen.

Amazon fragt an dieser Stelle, welche Ereignisse die Abarbeitung der in der Lambda-Funktion enthaltenen Logik auslösen können. Auch hier ist das Benutzerinterface nicht unbedingt befriedigend. Klicken Sie in das in Bild 4 gezeigte Quadrat, um eine Liste der möglichen Trigger auf den Bildschirm zu holen. Wählen Sie danach die Option *Alexa Skills Kit* aus, um die AWS-Funktion mit Alexa zu verbinden.



Wer auf das leere Rechteck klickt, wird mit einer Liste möglicher Trigger belohnt (Bild 4)

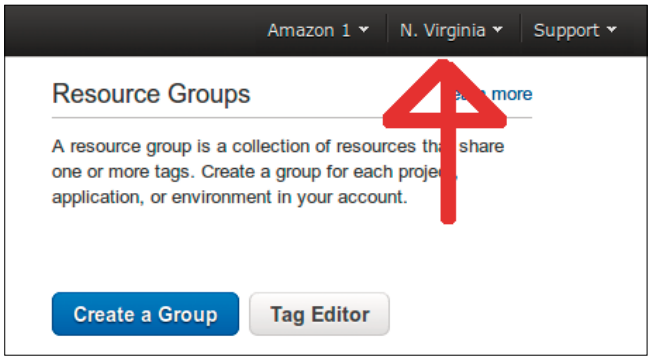
Damit sind wir zur Konfiguration der Logik bereit. Als Name wird in den folgenden Schritten *NMGFirstLambda* angenommen, der Inhalt des Beschreibungsfelds kann im Großen und Ganzen nach Belieben ausgewählt werden. Zum Zeitpunkt der Drucklegung unterstützt Lambda die folgenden vier Ausführungsumgebungen: Java 8, Node.js, Node.js 4.3 und Python 2.7.

Erzeugung von Node.js-Lambdas

Wir wollen fürs Erste – schon aus Gründen der Bequemlichkeit – auf Node.js 4.3 setzen. Wer die normale Node.js-Variante auswählt, erlaubt Amazon das beliebige Deployment aktualisierter Versionen der Ausführungsumgebung.

Der Assistent bietet bei der Erzeugung von Node.js-Lambdas ein Funktionsskelett an, das wir fürs Erste übernehmen wollen. Als nächstes Hindernis präsentiert sich die Einrichtung einer Zugriffsrolle. Es handelt sich dabei um eine Abstraktionsschicht, die die zum Zugriff auf eine Lambda-Funktion berechtigten Programme und Benutzer festlegt.

Klicken Sie im Feld *Role* auf die Option *Create a custom role*. AWS öffnet daraufhin ein Pop-up-Fenster, das Ihnen die Einrichtung einer neuen Rolle namens *lambda_basic_execution* anbietet. Klicken Sie auf *Allow*, um das Anlegen abzuschließen.



Steht hier nicht *N. Virginia,* so haben Sie ein Problem (Bild 3)

Nach dem Erzeugen einer neuen Rolle muss diese noch von Hand übernommen werden. Klicken Sie dazu abermals in das *Role*-Feld und wählen Sie nun die Option *Choose an existing role* aus. Das AWS-Backend blendet daraufhin eine weitere Textbox ein, in dem Sie die neu erstellte Rolle auswählen können.

Nach dem Anklicken von *Next* erscheint ein Übersichtsbildschirm, in dem Sie die AWS-Lambda-Funktion durch Anklicken des blauen Buttons in das System übernehmen und zur Ausführung freigeben.

Was ist ein Skill?

Amazon teilt Alexa und AWS auf: Entwickler, die mit dem Sprachsystem arbeiten, müssen – wahrscheinlich aus kartellrechtlichen Gründen – nicht unbedingt AWS-Nutzer sein. Für uns ist dies insofern ärgerlich, als wir im nächsten Schritt in die unter https://developer.amazon.com/edw/home.html bereitstehende Alexa-Entwicklerkonsole wechseln müssen.

Dabei ist ein abermaliges Anmelden erforderlich. Achten Sie darauf, das bei zur Anmeldung von AWS verwendete Konto auch an dieser Stelle einzusetzen, um den Zugriff auf die Skills beziehungsweise Lambda-Methoden zu ermöglichen.

Klicken Sie danach auf die Option *Get started*, die in der Rubrik *Alexa Skills Kit* bereitsteht. Die Konsole zeigt daraufhin eine Liste der schon vorhandenen Skills an. Falls Sie noch keinen haben, klicken Sie auf die Option *Add New Skill*. Alexa fragt im nächsten Schritt nach der Sprache des Skills. Wir wollen fürs Erste *Deutsch* auswählen. Im Feld *Skill Type* wählen wir das *Custom Interaction Model* aus, als Name wählen wir nun *NMGFirstSkill*.

An dieser Stelle findet sich eine kleine Besonderheit: Benutzer können Amazon Alexa dazu anweisen, bestimmte Anfragen an bestimmte Skills weiterzuleiten. Jeder Skill besitzt einen als Invocation Name bezeichneten Namen, der ihn eindeutig identifiziert und der vom Entwickler festgelegt wer- ▶

den muss. Amazon erlegt den Nutzern bei der Auswahl dieses Strings vergleichsweise strenge Regeln auf. Die unter https://developer.amazon.com/appsandservices/solutions/alexa/alexa-skills-kit/docs/choosing-the-invocation-name-for-an-alexa-skill#invocation-name-requirements zu findende Liste enthält eine Vielzahl von Einschränkungen, die beachtet werden müssen.

Wichtig ist, dass Skills aus zwei oder noch mehr Wörtern bestehen müssen. Einwortige Skill-Invokationen sind nur dann erlaubt, wenn es sich dabei um einen Markennamen des jeweiligen Unternehmens handelt.

Wir wollen hier stattdessen den String *Erster NMG* verwenden. Im Interesse maximaler Kompatibilität setzen wir das Feld *Audio Player* auf *False*. Es handelt sich dabei um eine Komponente, die im Moment von diverser Hardware – wie zum Beispiel dem Fire TV – nicht unterstützt wird. Klicken Sie im nächsten Schritt auf *Next*, um zum Eingabefeld für Intentschemata und Beispielausdücke (Sample Utterances) zu gelangen.

Spracherkennungsverfahren von Alexa

Hier müssen wir abermals reflektieren, um uns besser mit dem Funktionieren von Alexa und dem darin implementierten Spracherkennungsverfahren vertraut zu machen. In der Welt der Spracherkennung gibt es zwei Vorgehensweisen. Die von Systemen wie Dragon und Co. implementierte komplette Erkennung ist insofern kompliziert, als der Computer keinerlei Informationen über die Art der eingehenden Sprachinformationen hat.

Systeme wie Amazon Alexa nutzen ein von Microsoft mit dem Kinect 2 erstmalig einer breiteren Nutzerschaft vertraut gemachtes Verfahren, das mit Grammatiken arbeitet. Entwickler legen hier im ersten Schritt fest, mit welchen Spracheingaben zu rechnen ist. Nutzer sind während der Interaktion mit dem System auf diese Eingabemöglichkeiten beschränkt.

Die damit einhergehende Reduktion der Flexibilität wird in der Praxis durch eine wesentlich höhere Erkennungsgenauigkeit wettgemacht. Dies funktioniert in der Praxis und wird im militärischen Bereich schon lange genutzt. Ein schönes Beispiel dafür wäre das in der SFRJ entwickelte Kampfflugzeug Novi Avion, das – in den späten 1980er Jahren entwickelt – grammatikbasierte Sprachsteuerung für die Avionik anbot.

Analogien zu Android

Im Fall von Amazon Alexa ist die Struktur einfach. Die von einem Skill zu bearbeitenden Aufgabenstellungen werden – Analogien zu Android sind dabei rein zufällig – als Intents bezeichnet. Die einzelnen Intents können zudem einen oder mehrere Slots mitbringen. Es handelt sich dabei um Metaobjekte, die für die Verarbeitung von variablen Eingabedaten wie zum Beispiel Zahlen, Ortsnamen oder Ähnlichem vorgesehen sind.

Die eigentliche Konfiguration der Intelligenz erfolgt durch einen JSON-String, der nach dem unter https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/alexa-skills-kit-interaction-model-reference#intent-schema-syntax-json im Detail beschriebenen Format aufzubauen ist. Im Fall unseres Beispiels sieht er so aus:

Tabelle 1: Vordefinierte Datentypen

Schema	Erkennt
AMAZON.DATE	Datumsinformationen
AMAZON.DE_CITY	Namen größerer Städte in Deutschland
AMAZON.DE_FIRST_NAME	Häufige Vornamen von Personen; die Wissensdatenbank ist auf Deutschland bezogen
AMAZON.DURATION	Informationen über Zeitspannen, die vom Benutzer auf verschiedene Arten beschrieben werden können
AMAZON.EUROPE_CITY	Die Namen wichtiger Städte in Europa
AMAZON.FOUR_DIGIT_NUMBER	Vierstellige Zahlen mit optimierter Erkennungsleistung
AMAZON.NUMBER	Zahlen beliebiger Länge

```
{
  "intents": [
    {
      "intent": "InfoOnDate",
      "slots": [
        {
          "name": "Date",
          "type": "AMAZON.DATE"
        }
      ]
```

Wir deklarieren an dieser Stelle zwei Intents. Das Intents-Array dient dabei als Containerstruktur. Intent eins nimmt einen Parameter entgegen, der in Form eines Datums vorliegt.

Amazon unterstützt Entwickler hier insofern, als es eine Gruppe von vordefinierten Datentypen gibt, deren Erkennung vollautomatisiert abläuft. Tabelle 1 zeigt einige besonders interessante Datentypen.

Der zweite Slot ist vom Aufbau her einfacher. Es handelt sich hier um einen reinen Intent, der keinerlei Parameter entgegennimmt:

```
    },
    {
      "intent": "SayHello"
    }
  ]
}
```

Da wir im Moment keine selbst definierten Slots verwenden, bleibt dieser Teil des Formulars leer.

Als nächste Aufgabe müssen wir Alexa mit den Sätzen vertraut machen, mit denen die Nutzer die diversen Intents in Zukunft aktivieren sollen.

Amazon arbeitet mit dem Prinzip der Beispielausdrücke. Es handelt sich dabei um einen oder mehrere Sätze, die einen Teil der verbalen Eingabe abdecken. Zu ihrer Beschreibung kommt eine vergleichsweise einfache Syntax zum Einsatz. Ein Beispielbegriff beginnt mit dem Namen des Intents, der von ihm angesprochen wird. Darauf folgt der Rest des Satzes, ein Zeilenendzeichen leitet den nächsten Beispiel-Term ein. Slots werden dabei mit ihrem Slotnamen und geschwungenen Klammern beschrieben. Im Fall unseres Beispiels sieht die Aussagestruktur folgendermaßen aus:

```
InfoOnDate Was an {Date} passierte
InfoOnDate um Daten zu {Date}
InfoOnDate über Informationen zu {Date}
SayHello Hallo zu sagen
SayHello Freundlich zu sein
SayHello Willkommen zu rufen
```

Amazon empfiehlt in der Dokumentation, so viele Sample-Ausdrücke wie möglich anzulegen. Das Vorhandensein von drei Ausdrücken gilt dabei als Mindeststandard. Sie werden in der Beschreibung der jeweiligen Skills verwendet.

Interessant ist in diesem Zusammenhang, dass die eingegebenen Beispielfragen nicht gesprochen werden. Benutzer arbeiten mit einer speziellen Anfragesyntax, die Schlüsselbegriffe nutzt. Als Beispiel dafür bietet Amazon die folgenden sechs Sätze an, die die Nutzung eines virtuellen Horoskopdiensts demonstrieren:

- Alexa, frage bei Astro Dienst nach dem Tageshoroskop für Zwilling.
- Alexa, frage Astro Dienst nach dem Horoskop für Zwilling.
- Alexa, bitte Astro Dienst um das Horoskop für Zwilling.
- Alexa, bitte Astro Dienst um ein Horoskop für Zwilling.
- Alexa, bitte Astro Dienst um das Horoskop für Zwilling.
- Alexa, frage den Astro Dienst um das Horoskop für Zwilling.

Es ist von eminentester Bedeutung, dass die Beispielausdrücke auf eine natürliche Art und Weise mit der restlichen Anfragesyntax von Alexa kombiniert werden können. Die praktische Erfahrung des Autors lehrt, dass Lektoren, Germanisten und sonstige Textexperten meist mehr Schaden als Nutzen anrichten, wenn ihnen die Utterances vorgelegt werden.

Nach dem Anklicken von *Next* verifiziert das Alexa-Backend das eingegebene Sprachmodell. Wundern Sie sich nicht, wenn der grüne Haken neben dem betreffenden Punkt erst nach einiger Zeit erscheint. Tritt ein Fehler auf, so macht das Backend Sie darauf aufmerksam.

Erfreulicherweise können Sie während des Durchlaufens der Verifikation schon die nächsten Parameter einstellen. Alexa fragt Sie in der Rubrik *Konfiguration* danach, woher der auszuführende Lambda-String beziehungsweise die abzuarbeitende Logik bezogen werden soll.

Das Backend blendet daraufhin ein Textfeld ein, in dem Sie den Namen des jeweiligen Lambda-Programms eingeben. Amazon erwartet an dieser Stelle übrigens nicht den im Namensfeld eingegebenen String, sondern einen als ARN bezeichneten Identifyer. Er wird im Backend von Lambda auf der oberen rechten Seite eingeblendet (Bild 5) – im Fall unseres Beispiels lautet er *arn:*aws:*lambda:us-east-1:40236349 0054:function:NMGFirstLambda*.

Das Backend wechselt nach Eingabe des Namens in die Testansicht. Da wir im Moment noch keine Logik implementiert haben, lassen wir das Browserfenster geöffnet und kehren zur Lambda-Konsole zurück.

Lasst uns codieren

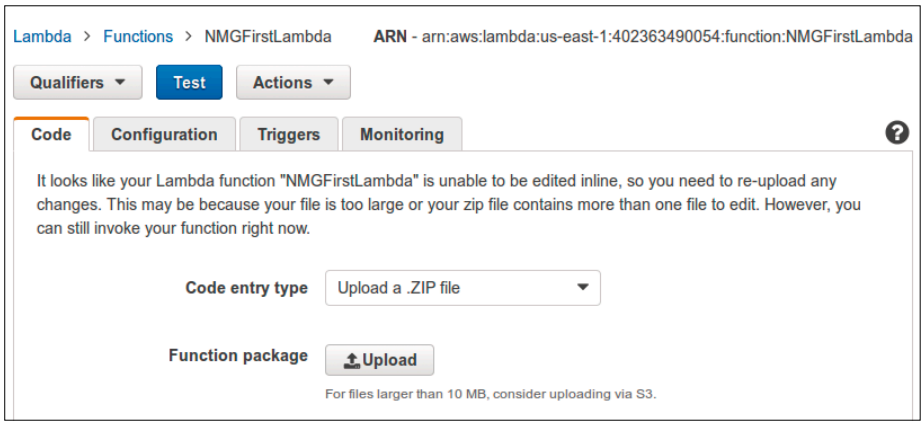
Wer seine Skills mit Javascript entwickeln möchte, sollte an dieser Stelle auf NPM setzen. Installieren Sie die Arbeitsumgebung auf Ihrer Workstation und wechseln Sie in ein beliebiges Verzeichnis, das fortan als Arbeitsverzeichnis dient.

Im nächsten Schritt muss im neu angelegten Verzeichnis ein Node.js-Projekt angelegt werden. Dies lässt sich am einfachsten über das Kommando *npm init* bewältigen, das im Rahmen der Erstellung einige Parameter abfragt:

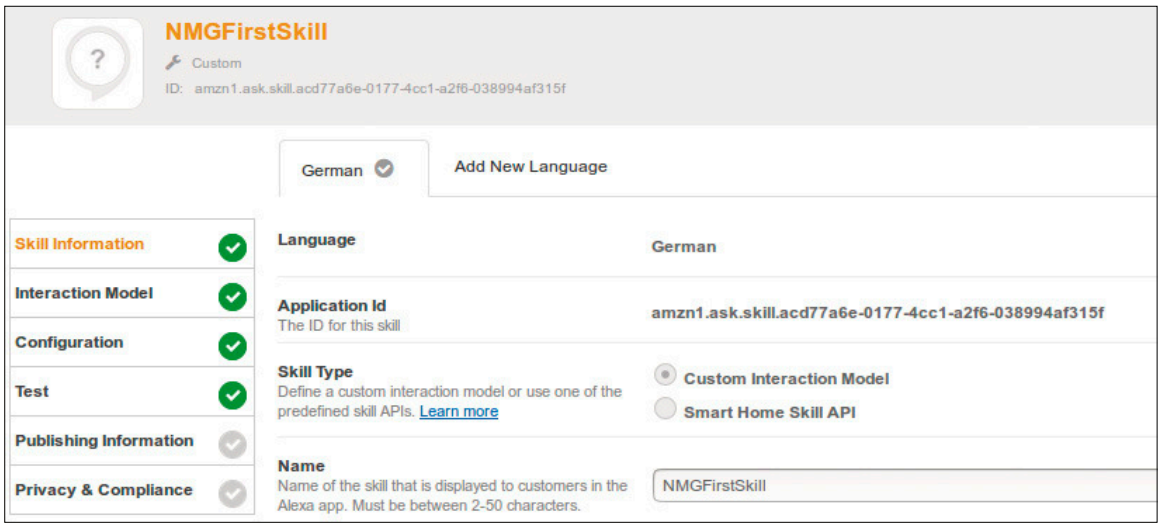
```
tamhan@TAMHAN14:~/Desktop/nmgskill$ npm init
This utility will walk you through creating a package.
json file.
...
Press ^C at any time to quit.
name: (nmgskill)
version: (1.0.0)
...
About to write to /home/tamhan/
Desktop/nmgskill/package.json:
```

Beim Erstellen von kleinen Beispielen ist es hilfreich, die von *npm init* vorgegebenen Basiswerte durch mehrfaches Drücken der Eingabetaste zu übernehmen. Nach getaner Arbeit entsteht eine *.package*-Datei, die das Projekt als Ganzes beschreibt.

npm init ist nicht in der Lage, eine Ordnerstruktur zu erzeugen. Erstellen Sie aus diesem Grund von ▶



Kopieren Sie den ARN am einfachsten in die Zwischenablage (Bild 5)



Auch diese ID sollten Sie über die Zwischenablage übertragen, um Tippfehler zu vermeiden (Bild 6)

Hand ein Unterverzeichnis namens *src*, in dem das folgende Kommando abgearbeitet wird:

```
tamhan@TAMHAN14:~/Desktop/nmgskill/src$ npm install
--save alexa-sdk
nmgskill@1.0.0 /home/tamhan/Desktop/nmgskill
└┬ alexa-sdk@1.0.6
  ...

npm WARN EPACKAGEJSON nmgskill@1.0.0 No description
npm WARN EPACKAGEJSON nmgskill@1.0.0 No repository
field.
```

Die im Rahmen des Herunterladens auftretenden Warnmeldungen können allesamt ignoriert werden. Im nächsten Schritt muss eine Datei namens *index.js* angelegt werden, die als Einsprungpunkt in die restliche Programmlogik dient. Für unser Beispiel beginnt sie mit folgenden Deklarationen:

```
var APP_ID = undefined;
var AlexaSkill = require('./AlexaSkill');
```

Bei der Arbeit mit Alexa Skills hat sich die Nutzung der unter https://github.com/amzn/alexa-skills-kit-js/blob/master/samples/helloWorld/src/AlexaSkill.js bereitstehenden Datei *AlexaSkills.js* als Quasistandard etabliert. Das unter https://forums.developer.amazon.com/questions/3261/alexaskilljs-companion-to-all-skills.html und http://tobuildsomething.com/2015/08/14/Amazon-Alexa-JavaScript-SDK-The-Ultimate-Guide näher besprochene Modul stellt eine Art Helferbibliothek dar, die das Hantieren mit JSON und Co. wesentlich erleichtert.

Während die Inklusion des Moduls – es handelt sich dabei um eine nach den Regeln der Javascript-OOP von Osmani und Co. aufgebaute Klasse – ist hier auch die Anpassung der App-ID erforderlich. Es handelt sich dabei um einen Identifikationsstring, der im Backend von Alexa angezeigt wird (Bild 6). Im Fall unseres Beispiels lautet er *amzn1.ask.skill.acd77a6e-0177-4cc1-a2f6-038994af315f*.

Im nächsten Schritt parametrieren wir das Prototyp-System von Javascript, um eine neue Klasse namens *NMGSkill1* anzulegen:

```
var NMGSkill1 = function () {
  AlexaSkill.call(this, APP_ID);
};
NMGSkill1.prototype =
Object.create(AlexaSkill.prototype);
NMGSkill1.prototype.constructor = NMGSkill1;
```

Ganz unten am Ende der Datei kommt sodann das von Node.js bekannte Exports-Objekt, in dem wir die zu exponierenden Elemente deklarieren. Im Fall einer auf *AlexaSkills.js* aufgebauten Klasse genügt dazu folgender Code:

```
exports.handler = function (event, context) {
  var mySkill = new NMGSkill1();
  mySkill.execute(event, context);
};
```

Unser Skill kommt im Moment ohne Sitzungsverwaltung aus. Aus diesem Grund legen wir die beiden Methoden nur als Skelette an, die keine weitere Logik enthalten.

Informationen über Aufrufe

Wer daran interessiert ist, Informationen über die jeweiligen Aufrufe zu sammeln und sichtbar zu machen, kann dazu natürlich auf Instrumentierungsfunktionen wie *Console.log* zurückgreifen:

```
NMGSkill1.prototype.eventHandlers.onSessionStarted =
function (sessionStartedRequest, session) { };
NMGSkill1.prototype.eventHandlers.onSessionEnded =
function (sessionEndedRequest, session) { };
```

Als Nächstes müssen wir uns der Methode *onLaunch* zuwenden. Sie wird immer dann aufgerufen, wenn der Benutzer den Skill aktiviert, ohne weitere Informationen über den ge-

wünschten Intent anzuliefern. Wir fragen in diesem Fall durch Nutzung des *response*-Objekts beim Nutzer nach, um mehr über seine Wünsche zu erfahren:

```
NMGSkill1.prototype.eventHandlers.onLaunch = function
(launchRequest, session, response) {
  var speechOutput = "Herzlich Willkommen zur
  Applikation der NMG. Bitte artikulieren Sie Ihre
  Wünsche.";
  var repromptText = "Wenn Sie nichts sagen, kann ich
  Ihnen auch nicht helfen.";
  response.ask(speechOutput, repromptText);
};
```

Die Funktion *response.ask* nimmt dabei zwei Parameter entgegen. Der erste wird anfangs ausgegeben, der zweite dient als Bestärkung der an den Nutzer gestellten Frage.

Die vergleichsweise seltsame Namenskonvention – Amazon verbietet bei der Benennung von Intents unter anderem die Nutzung von Leerzeichen – erklärt sich bei einem Blick auf die Funktion *intentHandlers*:

```
NMGSkill1.prototype.intentHandlers = {
  "InfoOnDate": function (intent, session, response) {
    response.tellWithCard("Info on Date",
    "Info on Date", "Info on Date!");
  },
  "SayHello": function (intent, session, response) {
    response.tell("Hello World!");
  }
};
```

intentHandlers liefert ein Objekt zurück, das die für die einzelnen Sentenz vorgesehenen Händlerfunktionen bereitstellt beziehungsweise exponiert. Da wir uns im Moment mit der Implementierung nicht weiter aufhalten wollen, nutzen wir zwei inline deklarierte Funktionen. Damit ist die Arbeit an der Lambda-Funktion abgeschlossen. Prüfen Sie, ob die Datei *AlexaSkill.js* ebenfalls im *src*-Verzeichnis liegt. Verpacken Sie die beiden Dateien sodann in ein ZIP-Archiv.

Öffnen Sie die Funktionsliste im AWS-Backend und klicken Sie auf den *Code*-Tab. Dort findet sich – neben einer Textbox zur Bearbeitung – auch eine Combobox. Klicken Sie diese an und wählen Sie die Option *Upload .zip File*, um das im vorigen Schritt erzeugte Archiv hochzuladen. Klicken Sie danach unbedingt auf *Save*, um den Upload-Prozess abzuschließen. Unterbleibt dies, so wird das Archiv nicht auf den Server übertragen und der Funktionscode wird nicht aktualisiert. Beachten Sie auch, dass Sie nach dem Hochladen eines Archivs nicht mehr in die Rubrik *Code* wechseln dürfen. Tun Sie dies trotzdem, so reaktivieren Sie den im Rahmen der Erzeugung der Funktion angelegten Default-Code, der für unsere Zwecke nicht nutzbar ist.

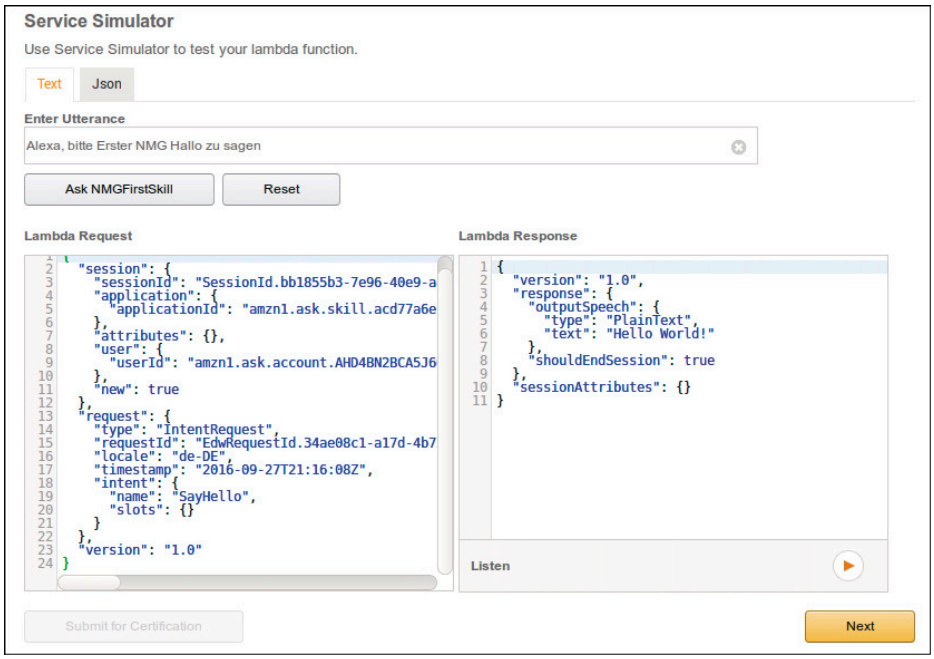
Wechseln Sie stattdessen wieder in die Alexa-Entwicklerkonsole und wählen Sie dort die Rubrik *Test*. Aktivieren Sie die Option *Enable Testing* und geben Sie einen beliebigen, an *First NMG* gerichteten Abfragestring in die Textbox ein. Nach dem Anklicken von *Ask NMGFirstSkill* wird sich der Server an die Arbeit machen und eine Antwort zurückliefern. Dieses Verhalten ist in Bild 7 gezeigt.

Auf Fehlerjagd

Ein alter Kalauer besagt, dass Programmierer 10 Prozent ihrer Zeit mit dem Entwickeln verbringen – die restlichen 90 Prozent gehen für die Suche nach Fehlern drauf. Eine in der Lambda-Ausführungsumgebung laufende Codestruktur ist insofern etwas diffizil, als sie sich nicht auf der lokalen Maschine befindet und dementsprechend nicht ohne Weiteres für Debugger ansprechbar ist.

Die einfachste Methode zur Lösung des Problems besteht darin, per Console.Log Meldungen in das Log der *Lambda*-Methode zu schreiben. Neben dem Klassiker unterstützt Lambda auch die Aufrufe *console.error(), console.warn()* und *console.info()*. Die Ausgabe der Logging-Methoden lässt sich sowohl während als auch nach der Programmausführung in CloudWatch abernten. Das unter https://console.aws.amazon.com/cloudwatch bereitstehende Portal ist bis zu einem gewissen Grad eine Eier legende Wollmilchsau. Das Finden der benötigten Informationen ist jedoch mitunter nicht einfach.

Die für uns interessante Rubrik trägt den Namen *Logs*. Achten Sie darauf, nicht versehentlich die im Untermenü ▶



Die Anfrage ist in unserer Funktion angekommen (Bild 7)

Logs, Metrics befindliche *Logs*-Option anzutreten. Nach dem Anklicken des Namens der zu analysierenden Funktion zeigt das Backend eine Liste der Logging-Gruppen. Klicken Sie diese an, um die einzelnen Datenströme zu selektieren.

Das eigentliche Hantieren mit dem Log-Viewer ist dann unproblematisch. Achten Sie lediglich darauf, dass Amazon von Zeit zu Zeit (zum Beispiel beim Hochladen eines Archivs) den neuen Datenstrom anliegt, der das Menü zum Nachladen neuerer Einträge außer Gefecht setzt.

Internationalisierung macht Freude

Funktioniert der Code im Großen und Ganzen, so möchte man im Allgemeinen das User Interface testen. In den USA ansässige Entwickler haben es hier leicht: Sie kaufen einen der um 50 US-Dollar erhältlichen Echo Dots und spielen nach Herzenslust herum. Wer keine Hardware kaufen kann oder möchte, kann stattdessen auf das unter https://echosim.io bereitstehende Alexa Skill Testing Tool zurückgreifen. Zum Einloggen verwenden Sie das gleiche Konto, das für die Anmeldung in AWS und im Alexa-Developer-Backend zum Einsatz kommt. Leider ist das Testwerkzeug im Moment nur mit einer englischsprachigen Echo-Implementierung verbunden (siehe dazu zum Beispiel https://www.reddit.com/r/amazonecho/comments/5540sb/fyi_virtual_alexas_echosim_rodger_wont_work_in) – dies bietet uns die Gelegenheit, uns mit der Realisierung mehrsprachiger Sprachdienste zu beschäftigen.

Öffnen Sie den Skill im Dashboard und klicken Sie auf den Tab *Add New Language*. Alexa fragt an dieser Stelle nach der Sprache. Wir wollen *English U.S.* auswählen, um EchoSim nutzen zu können. Als *Invocation Name* kommt diesmal *first NMG* zum Einsatz. Das Backend übernimmt die Konfigurationseinstellungen automatisch – da Englisch und Deutsch nur wenig miteinander zu tun haben, muss das Interaktionsschema allerdings neu angelegt werden. Kopieren Sie das Intent-Schema eins zu eins aus dem deutschen Skill und wenden Sie sich danach den Beispielausdrücken zu. In englischsprachigen Skills kommt die folgende Syntax zum Einsatz:

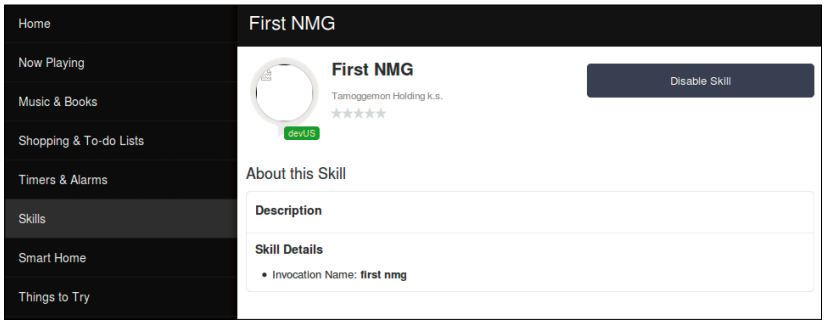
- Alexa, ask Daily Horoscopes for the horoscope for Gemini
- Alexa, ask Daily Horoscopes about Gemini
- Alexa, ask Daily Horoscopes what is the horoscope for Gemini
- Alexa, ask Daily Horoscopes what's the horoscope for Gemini
- Alexa, ask Daily Horoscopes to give me the horoscope for Gemini
- Alexa, ask Daily Horoscopes to tell me the horoscope for Gemini

Eine passende Version der Beispielausdrücke für den englischsprachigen Markt sähe folgendermaßen aus:

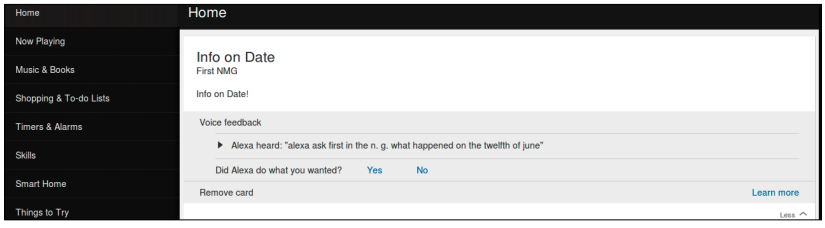
- InfoOnDate what happened on {Date}
- InfoOnDate about information on {Date}
- InfoOnDate to tell me more about {Date}
- SayHello be friendly
- SayHello say hello
- SayHello call out a warm welcome

Wechseln Sie danach in die Testansicht und senden Sie eine Anfrage nach dem Schema *Alexa, ask First NMG to be friendly*. Alexa wird sie – wie weiter oben im Fall des deutschen Skills – beantworten.

Vor der weitergehenden Nutzung im Alexa-Tester müssen wir unseren Skill im ersten Schritt aktivieren. Dazu ist die Alexa-App notwendig, die im Play Store (https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=en) leider nur für in der BRD oder in den USA befindliche Nutzer herunterladbar ist. Als Alternative bietet sich die unter http://alexa.amazon.com/spa/index.html#welcome bereitstehende Webversion an. Öffnen Sie sie in einem Browser Ihrer Wahl. Klicken Sie danach auf den *Skills*-Tab und aktivieren Sie den am oberen rechten Bildschirmrand eingeblendeten Link *Your Skills*. Die Alexa-App blendet daraufhin eine Liste aller in Ihrem Konto befindlichen Skills ein – prüfen Sie, ob *First NMG* wie in Bild 8 gezeigt aktiviert ist.



Dieser Skill ist einsatzbereit (Bild 8)



In der Alexa-App aufscheinende Karten können dem Nutzer zusätzliche Informationen bieten (Bild 9)

Öffnen Sie im nächsten Schritt EchoSim und loggen Sie sich abermals ein. Auf der Ubuntu-Workstation des Autors funktionierte die Webseite nur mit Google Chrome. Nach einer Prüfung der Mikrofoneinstellungen müssen Sie den *Listen*-Button gedrückt halten, während Sie Ihre Spracheingabe tätigen. Die Antwort des Systems erfolgt über die Systemlautsprecher.

An dieser Stelle ist der Unterschied zwischen der Methode *tell* und der Methode *tellWithCard* interessant. Aus Sicht der Sprach-

interaktion verhalten sich die beiden Funktionen gleich, da Alexa die angelieferten Informationen über die Kopfhörer ausgibt. Der Unterschied zwischen den beiden Varianten zeigt sich erst, wenn Sie nach dem Aufruf des jeweiligen Kommandos in die Alexa-Applikation wechseln. Die von *InfoAboutDate* erzeugte Karte präsentiert sich dort wie in Bild 9 gezeigt.

Parameter nutzen

Damit stellt sich die Frage, wie wir die eingehenden Parameter – der Skill *InfoOnDate* nimmt ja ein Datum entgegen – auf Codeseite weiterverarbeiten. Dazu ist im ersten Schritt eine Änderung der Methode *intentHandlers* erforderlich, die nun folgendermaßen aussieht:

```
NMGSkill1.prototype.intentHandlers = {
  "InfoOnDate": function (intent, session, response) {
    response.tell(intent.slots.Date.value + "was a
    nice day");
  },
  "SayHello": function (intent, session, response) {
    response.tell("Hello World");
  }
};
```

An dieser Stelle sind zwei Änderungen relevant: Erstens nutzen wir statt *tellWithCard* die einfachere Variante *tell*, da wir die weiter oben erwähnte Kartenansicht an dieser Stelle nicht mehr brauchen.

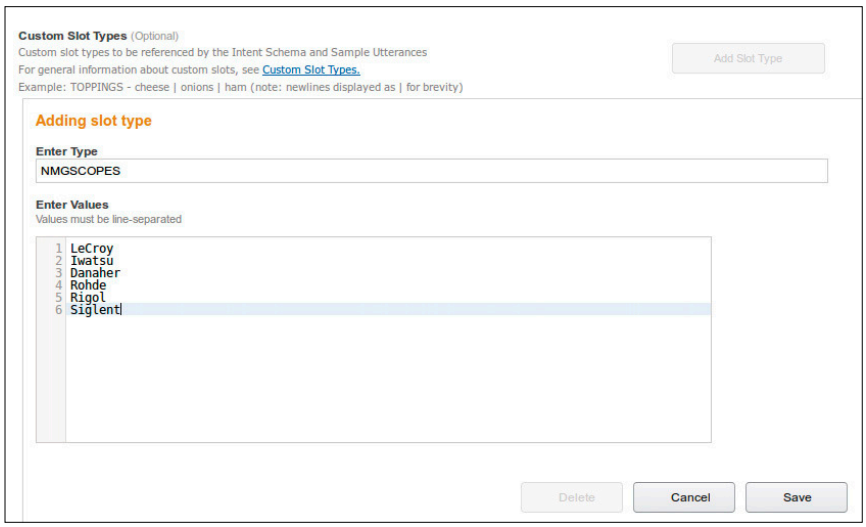
Das eigentliche Entgegennehmen der Eingabe erfolgt über die als Parameter angelieferte intent-Enumeration. Die einzelnen Slotwerte finden sich normalerweise in der Variablen *slots* – achten Sie darauf, dass die Slotnamen Case-sensitiv sind.

Verpacken Sie den Code an dieser Stelle in ein Archiv und laden Sie ihn in Richtung der Lambda-Funktion. Nach dem neuerlichen Anstoßen wird Alexa das angelieferte Datum in der Antwort erwähnen.

Eine besonders interessante Variante ist die Nutzung eines selbst errichteten Slots. Ein Beispiel dafür wäre eine Applikation, die den Benutzer nach seinem Lieblingsoszillographenhersteller fragt.

Kehren Sie dazu in die Rubrik *Interaction Model* zurück und klicken Sie im Feld *Custom Slot Type* auf den Button *Add Slot Type*. Das Alexa-Backend blendet daraufhin das in Bild 10 gezeigte Fenster ein, in dem Sie sowohl den Typnamen als auch die dazugehörige Werteliste angeben können.

Im Fall unseres Beispiels wollen wir uns auf einige weitverbreitete Hersteller beschränken. Konfigurieren Sie den Skill deshalb wie in der Abbildung gezeigt und klicken Sie nach dem Eingeben aller Werte auf *Save*. Alexa übernimmt die neu angelegte Deklaration sodann automatisch in die Wissensbasis. Nach dem erfolgreichen Durchlauf der Verifikation ist sie sofort einsetzbar. Erfreulicherweise blockiert die Überprüfung die restlichen Steuerelemente der Seite nicht – wir können also sofort weiterarbeiten.



Ein Custom Slot besteht aus einer Ansammlung von Werten (Bild 10)

Zur Nutzung des neuen Intents müssen wir eine Erweiterung des Intent-Arrays vornehmen. Fügen Sie oberhalb der im vorigen Abschnitt verwendenden Intent-Deklarationen folgenden Code ein:

```
{
  "intents": [
    {
      "intent": "InfoOnScope",
      "slots": [
        {
          "name": "Manufacturer",
          "type": "NMGSCOPES"
        }
      ]
    },
```

Ein auf einem selbst deklarierten Slot basierender Intent unterscheidet sich nur insofern von seinen gewöhnlichen Kollegen, als das Typ-Feld nun auf den weiter oben eingegebenen Namen lautet. Im Bereich der Beispielausdrücke findet sich ebenfalls nichts Überraschendes. Der neu angelegte Slot wird wie gewohnt in geschweifte Klammern gesetzt, um ihn für die Runtime zu markieren:

```
InfoOnScope about {Manufacturer}
InfoOnScope about how {Manufacturer} performs
InfoOnScope if {Manufacturer} is worth its price
```

An dieser Stelle sei angemerkt, dass es in älteren Versionen des Alexa-API ein als *Amazon.LITERAL* bezeichnetes Feld gab. Dieses lieferte einen String mit einer Best-Effort-Transkription der Benutzereingabe, die jedoch in der Praxis nur wenig befriedigend ausfiel. ▶

Amazon hat die Nutzung dieses Features mittlerweile untersagt. Wer es in einem seiner Skills trotzdem einsetzt, wird spätestens bei der Zertifikation aus dem Rennen geworfen.

Permutationen

Damit müssen wir uns abermals mit der Verarbeitung der Informationen beschäftigen. Da es sich bei unserem Custom Skill im Grunde genommen um einen gewöhnlichen Skill handelt, liegt die Nutzung des *intents*-Arrays nahe.

Leider weist Amazon in der Dokumentation explizit darauf hin, dass die Matching-Prozesse auch bei der Nutzung eines Custom Slots vergleichsweise tolerant sind. Erkennt die Spracherfassungslogik ein bestimmtes Wort mit hoher Wahrscheinlichkeit, so wird es zurückgeliefert – auch dann, wenn es nicht in der vom Entwickler angegebenen Liste aufscheint.

Zu allem Überfluss liefert die Engine im Erkennungsfall nicht unbedingt jenen Wert zurück, den man ihr in der Definition des Kastenslots eingeprägt hat. Tabelle 2 zeigt einige Permutationen, die aus der Dokumentation von Amazon übernommen wurden. Und nein: Es gibt keine klarere Spezifikation, die weitere Informationen über das Verfremdungs- beziehungsweise Encodierungsverfahren liefern würde.

Die Ermittlung der von Alexa angelieferten Resultate ist im Moment somit bis zu einem gewissen Grad Glückssache. Eine Möglichkeit, das Problem zu umgehen, ist, im Rahmen der Entwicklung des Programms einfach einmal alle Kommandos durchzutesten. Die angelieferten Resultate wandern sodann in ein Logfile, was zu folgender Variante von *intentHandlers* führt:

```
NMGSkill1.prototype.intentHandlers = {
  ...
  "InfoOnScope": function (intent, session, response) {
    response.tell("Got Oscilloscope info: " +
    intent.slots.Manufacturer.value );
    console.log(intent.slots.Manufacturer.value);
  },
```

Die Sprachausgabe die das erfolgreiche Erkennen eines Kommandos signalisiert, ist insofern hilfreich, als sie das blockartige Abarbeiten mehrerer Anfragen ermöglicht. Ist dies nicht der Fall, so können Sie während des Einsprechens der Testtexte nicht feststellen, ob Alexa die Informationen auch wirklich entgegengenommen hat.

Wer den Oszilloskop-Skill im vorliegenden Zustand ausführt, stellt fest, dass die Erkennung alles andere als zufriedenstellend ist. In vielen Fällen weist Alexa die eingehenden Anfragen nach Messgeräten dem *sayHello*-Intent zu. Dies liegt – neben Problemen mit dem deutschen Akzent des Autors – auch daran, dass die Beispielausdrücke der beiden Intents zu ähnlich sind. Eine bessere Version würde längere Beispielterme verwenden, etwa nach folgendem Schema:

```
InfoOnScope about oscilloscopes from {Manufacturer}
InfoOnScope about how oscilloscopes from {Manufacturer}
perform
InfoOnScope whether {Manufacturer} oscilloscopes are
worth their price
```

Ein Durchlauf aller verwendeten Hersteller ergab sodann das in der Tabelle 3 gezeigte Ergebnis – wer auf Nummer sicher gehen möchte, kann den eingehenden String mit diversen Algorithmen auf Ähnlichkeit prüfen.

Damit bleibt ein Problem: Was tun, wenn der Benutzer einen offensichtlich ungültigen Wert eingibt? Ein Lösungsweg wäre mehrfaches Nachfragen – dazu ist folgende Anpassung im Backend notwendig:

```
NMGSkill1.prototype.intentHandlers = {
  "InfoOnDate": function (intent, session, response) {
    response.tell(intent.slots.Date.value + "was a
    nice day");
  },
  "InfoOnScope": function (intent, session, response) {
    if(intent.slots.Manufacturer.value=="detroit" ||
    intent.slots.Manufacturer.value=="event"){
```

Tabelle 2: Permutationen

Im Slot angegebener Text	Gesprochene Form	Angelieferter String
two beers	two beers	2 beers
four inch	four inch	4"
all-in-one	all in one	all in 1
first amendment	first amendment	1st amendment
r. and b.	r. and b.	R&B
amazon.com	amazon dot com	amazon dot com

Tabelle 3: Ergebnisse der Testabfrage

Vorgesehener Name	Angelieferter String
LeCroy	detroit
Iwatsu	event
Danaher	gone
Rohde	old
Rigol	reigle
Siglent	sick

Links zum Thema

- School of Lambda
  http://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
- Fertige Beispielskills
  https://github.com/amzn/alexa-skills-kit-js

```
    response.tell("Thank you for your choice!");
  }
  else {
    response.ask("Sorry, I did not get that.",
    "Please try again!");
  }
  console.log(intent.slots.Manufacturer.value);
}
```

Wie im vorigen Fall verwenden wir auch diesmal das Ask-Objekt, um den Benutzer nach weiteren Informationen zu fragen. Alexa liefert die zweite Anfrage wie gewohnt aus, womit das Risiko einer Endlosschleife droht.

Anzumerken ist, dass auch diese Version nur sehr eingeschränkt funktioniert. In Tests des Autors wurde so gut wie nie ein wirklich brauchbares Ergebnis erzeugt. Es ist allerdings davon auszugehen, dass Amazon in nicht allzu ferner Zukunft an dieser Stelle nachbessert.

Laufe seitwärts

Komplexe Systeme – das Echo-Spracherkennungssystem gehört mit Sicherheit dazu – trainieren ihren Nutzern gewisse Verhaltensweisen an, die sie von Programm zu Programm übernehmen. Wer sein Produkt in diese vertraute Infrastruktur integriert, spart sich Kundensupportanfragen.

Amazon bildet die wichtigsten Teile des Alexa-GUI über spezielle Intents ab. Wer sein Programm in eines der Features einbinden möchte, implementiert den betreffenden Intent durch Anlegen eines Handlers für das jeweilige Ereignis. Tabelle 4 zeigt die wichtigsten Einsprungpunkte – die komplette Liste umfasst rund ein Dutzend Parameter und steht unter https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/implementing-the-built-in-intents#introduction bereit.

Als kleines Beispiel für die Integration zwischen App und Alexa wollen wir an dieser Stelle ein primitives Hilfesystem durchsprechen, das vom Benutzer mit den gewohnten Hilfe-Ausdrücken aktiviert werden kann. Dazu ist – wie immer – eine Anpassung der Intentstruktur erforderlich:

```
{
  "intent": "AMAZON.HelpIntent"
},
```

Amazon parametriert die im Betriebssystem angelegten Intents von Haus aus mit einigen Dutzend Beispielausdrücken, die die häufigsten Benutzeraussagen einfangen. Sollten Sie in Ihrem Programm aus irgendeinem Grund weitere Ausdrücke zu einem Systemintent hinzufügen wollen, so lässt sich dies nach folgendem Schema bewerkstelligen:

```
AMAZON.HelpIntent help me order a taxi
AMAZON.HelpIntent how do I order a taxi
```

Das eigentliche Entgegennehmen im Code erfolgt dann wie gewohnt – ergänzen Sie *intentHandlers* einfach um einen weiteren Eintrag.

Tabelle 4: Die wichtigsten Einsprungpunkte

String	Art des Intents
AMAZON.HelpIntent	Liefert Hilfeinformationen über den aktivierten Skill.
AMAZON.NoIntent	Erlaubt dem Benutzer das Verweigern einer Anfrage.
AMAZON.RepeatIntent	Erlaubt dem Benutzer das Wiederholen einer zuvor erledigten Aufgabe beziehungsweise Anfrage.
AMAZON.StartOverIntent	Bittet Alexa um einen Neustart des betreffenden Skills beziehungsweise des gerade laufenden Prozesses.
AMAZON.YesIntent	Erlaubt dem Benutzer das Bejahen einer Anfrage des Skills.

Wer seinen Skill mit der Gesamtheit der Alexa-Nutzer teilen möchte, muss nach dem erfolgreichen Durchtesten in die Alexa Developer Console zurückwechseln. Auf der linken Seite des Bildschirms findet sich der Tab *Publishing Information*, in dem Amazon diverse an den App Store erinnernde Informationen abfragt. Neben einer Reihe von Beschreibungstexten sind auch zwei Symbole in den Größen 108 x 108 und 512 x 512 Pixel erforderlich, die in der Skill-Liste aufscheinen.

Als letzte Hürde präsentiert sich sodann der Tab *Privacy& Compliance*, in dem Amazon nach einigen rechtlichen Informationen fragt. Nach dem wahrheitsgemäßen Ausfüllen aller Felder – Amazons Qualitätssicherungsteam ist sehr genau – fehlt nur noch ein Klick auf *Submit for Certification*, um den Skill zum Test freizugeben. Wenn er die Qualitätsansprüche von Amazon erfüllt, steht er einige Zeit später in der Rubrik *Skills* zur Verfügung. Achten Sie darauf, Ihre Nutzer darauf hinzuweisen, dass Sie den betreffenden Skill vor der erstmaligen Nutzung von Hand aktivieren müssen.

Fazit

Die Arbeit mit Alexa-Skills setzt bei Desktop- und Handcomputer-erfahrenen Entwicklern Umdenken voraus. Wer sich an die Vorgehensweisen gewöhnt hat, kann Skills mit minimalem Aufwand zusammenstellen – aus technischer Sicht ist die Entwicklung nicht sonderlich schwierig.

Unklar ist im Moment noch, wie die Erzeugnisse monetisierbar sind. Amazon bietet derzeit keine Möglichkeit an, die Nutzerschaft zur Kasse zu bitten. ■

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

Sashkin / Fotolia

DAS MAGENTO COMMERCE ORDER MANAGEMENT

Auf allen Channels dasselbe

Was im Privaten nervt, lässt Omnichannel-Händler zufrieden durchatmen.

In Omnichannel-Modellen ist das reibungslose Management von Bestell-, Bestands- und Fulfillment-Informationen über eine Vielzahl von Kanälen hinweg entscheidend für den Unternehmenserfolg: also für zufriedene Kunden, gute Verkaufszahlen und niedrige Prozesskosten.

Damit Omnichannel-Händler entspannt in den Tag starten können, müssen daher alle Channels intelligent miteinander verknüpft sein, sowohl technisch als auch prozessseitig. Je komplexer die Omnichannel-Organisation, umso aufwendiger der Aufbau. Für den Einsatz mit Magento 2 gibt es jetzt eine neue Lösung, die echten Omni-Commerce verspricht: das Magento Commerce Order Management (Bild 1).

Omnichannel ist eine Kostenfrage

Der Investitionsbedarf für den Aufbau und die Verzahnung aller relevanten Kanäle ist nicht unerheblich, wenn auch immer einzelfallabhängig. Wie beim Thema Bezahlverfahren ist der richtige Mix der Omnichannel-Services wichtig: Click-to-Shop, Buy Online Return In Store (BORIS) oder Buy Online Pickup In Store (BOPIS) = Click & Collect? Beim Versand Same Day Delivery oder doch eher ein Personal Delivery Window?

Zielgruppengerechte Auswahl

Die Liste der Möglichkeiten, der beteiligten Stakeholder und der Systeme ist gewaltig und vergleichbar mit einem Sinfonieorchester. Meist reicht aber schon ein deutlich kleineres Ensemble für den Erfolg. Wichtig ist, im ersten Schritt eine zielgruppengerechte Auswahl zu treffen und diese dann später bei Bedarf dynamisch weiterzuentwickeln.

Denn letztlich geht es zwar darum, dass Kunden die Waren ordern, bezahlen, erhalten und zurückgeben können, so wie es ihnen gefällt. Allerdings ist es legitim, wenn Händler diese Omnichannel-Services auf die profitableren Kundengrup-

pen maßschneidern. Deren Bedürfnisse und Verhalten gilt es daher zu kennen. Allgemein gilt jedoch: Je komplexer die Omnichannel-Organisation, desto aufwendiger ist der Aufbau der entsprechenden Infrastruktur.

Der passende Mix allein genügt aber nicht, um die Kosten im Griff zu behalten. Um ein solches Orchester effizient zu dirigieren, müssen alle Informationen, die für das Management von Bestellungen nötig sind, aus allen Kanälen und für alle mit der Verwaltung der einzelnen Kanäle befassten Systeme jederzeit bereitstehen. Dies gelingt nur über eine zentrale Plattform: Das ist der Zweck des Magento Commerce Order Managements (MCOM).

Das Magento Commerce Order Management

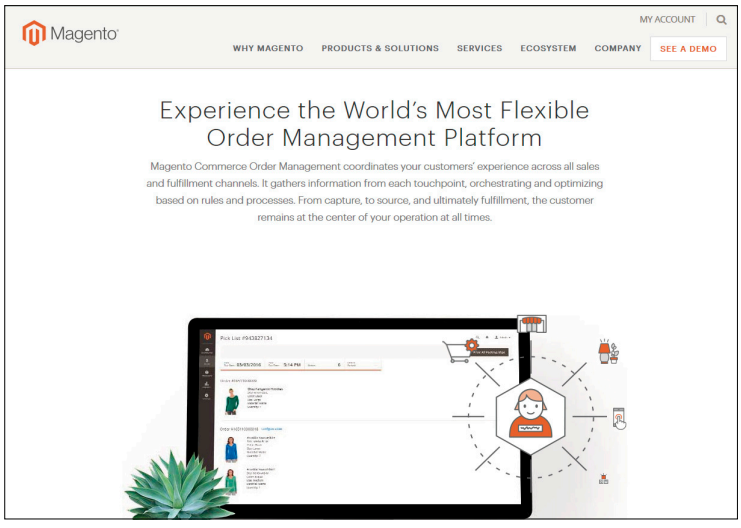
Das Magento Commerce Order Management ist eine cloudbasierte Standalone-Order-Management-Lösung. Magentos Idee ist es, eine zentrale Instanz zu schaffen, um alle Quellen, in denen Informationen zu Lager-, Bestell- und Fulfillment-Prozessen verwaltet beziehungsweise generiert werden, zu vernetzen und so eine 360-Grad-Sicht auf den aktuellen Status von Waren zu erhalten.

Auf der Nachfrageseite kann jede Vertriebsplattform und jeder Service integriert werden, ganz gleich ob Webshop, Mobile Shop, App, Marktplatz, Filiale oder Kundenhotline. Auf Angebots- oder Vorratsseite geht es um die Anbindung von Lieferanten, Lagern, Logistikpartnern oder eben auch Filialnetzen. Damit stellt das MCOM insbesondere eine Lösung für Omnichannel-Strategien in einem modernen Filialeinzelhandel- und Marktplatzmodell dar.

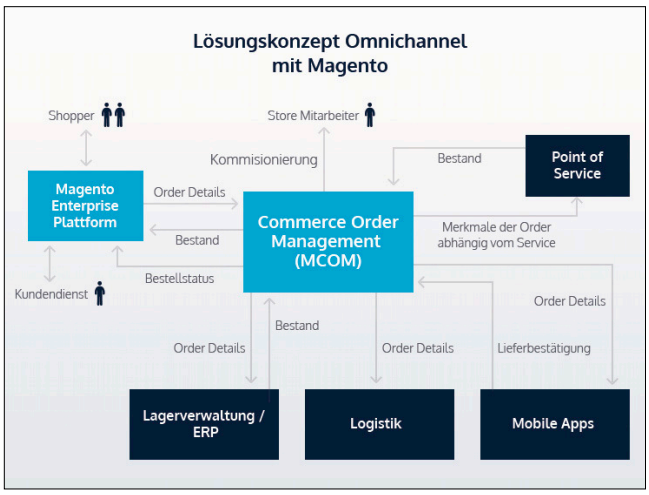
Leistungsspektrum

Vereinfacht lässt sich das Leistungsspektrum des Magento Commerce Order Managements in vier Felder einteilen:

- **Globale Bestandsverwaltung:** Der Bestand eines Produkts an jedem erdenklichen Lagerungsort eines Omnichannel-Netzwerks wird ganzheitlich erfasst, vom Ladenlokal über Lager bei Händlern, Lieferanten und Distributionszentren bis hin zu den disponierten Waren und dem Wareneingang.



Magento 2 bietet Optionen für ein echtes Omni-Commerce (Bild 1)



Omnichannel mit Magento realisiert (Bild 2)

- **Order Management:** Verwaltet werden alle Bestellarten, die Bestellabwicklung, Reservierungen, Retouren und Stornierungen sowie Versandarten.
- **Omnichannel Fulfillment:** Magento unterstützt die Kommissionierung bei Ship-to-Store, Ship-from-Store, Click & Collect sowie Pick & Pack-Verfahren und bindet Carrier, also Partner für den Frachtverkehr wie Speditionen, mit an.
- **Kundenservices:** Ein kanalübergreifendes Tracking aller Informationen zum Auftragsstatus sorgt für optimale Auskunftsqualität.

Dieses Leistungsspektrum dient dem Ziel, Omnichannel ganz aus Kundensicht zu gestalten und so ein positives Einkaufserlebnis zu schaffen, damit der Kunde wiederkommt (Bild 2). Das kann etwa bedeuten, Bestellungen nicht nur so schnell wie möglich zu bearbeiten, sondern sie wahlweise auf dem kürzesten Weg, zu den günstigsten Konditionen oder mit dem bestmöglichen Versicherungsschutz auszuliefern.

Die Workflows zwischen den einzelnen Akteuren einer Omnichannel-Organisation, um solche Kundenwünsche von Fall zu Fall zu erfüllen, sind sehr komplex, hängen von den unterschiedlichsten Faktoren ab und unterscheiden sich in jeder Organisation und in jedem Unternehmen erheblich.

Konfigurierbare Geschäftsregeln

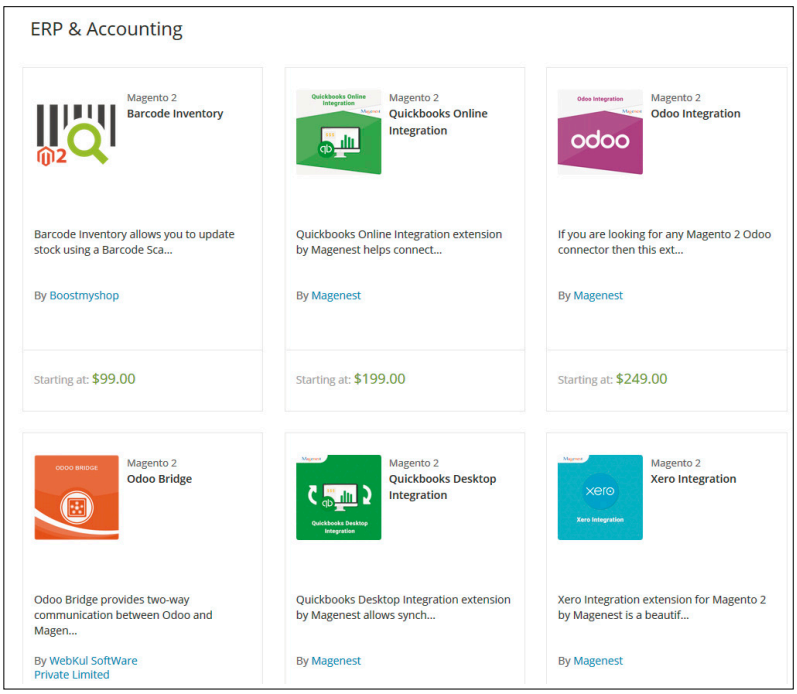
Deswegen setzt Magento unter anderem auf ein Set frei konfigurierbarer Geschäftsregeln, sogenannte Order Broker Rules. In den Kategorien Lager, Produkte, Versand und Bestellung können die unterschiedlichsten Parameter wie Öffnungszeiten, Arbeitskosten, Mindestbestand, Verpackung, Gebinde, grenzüberschreitender Versand, Lieferart, Dienstleister, Service Level et cetera frei angepasst werden. Damit stehen Nutzern mehr als 150 Varianten zur Verfügung, um ganz individuelle Fulfillment-Workflows zu gestalten.

Diese Variabilität erlaubt nicht nur die einfache Anpassung des Magento Commerce Order Ma- ▶

nagements an die spezielle Organisation eines Handelsunternehmens. Gerade komplexe Organisationsstrukturen mit mehreren Händlern, etwa Einkaufszentren, Outlets oder Franchise-Partnerschaften, profitieren von der Magento-Lösung.

Hier agieren im Backend die unterschiedlichsten Systeme. Order- und Fullfilment-Prozesse werden unterschiedlich abgebildet und es gibt mehrere Online-Shops mit einer Vielzahl von Checkout-Lösungen und Payment-Service-Providern.

Dennoch sollen den Kunden einheitliche Services geboten werden. Vergleichbar mit einer Middleware oder einem Gateway werden vom Magento Commerce Order Management die für die Bestellabwicklung nötigen Informationen extrahiert – unabhängig von der Quelle –, an zentraler Stelle zusammengeführt und für die anderen Systemen bereitgestellt. Eine direkte Kommunikation zwischen den diversen Systemen entfällt zugunsten einer übergeordneten Struktur. Damit wird die Komplexität beherrschbar.



Magento bietet eine breite Palette an Erweiterungen (Bild 3)

Exemplarischer Aufbau einer Commerce-Suite

Eine Magento Commerce Suite – gemeint ist eine vollständige IT-Infrastruktur – besteht exemplarisch aus der E-Commerce-Plattform, dem Commerce Order Management, einem Konnektor dazwischen, einer E-Mail-Client- und einer Call-Center-Lösung sowie den Drittsystemen des Händlers zur Verwaltung von Produkt-, Preis-, Kunden- und Unternehmensdaten sowie der Logistikprozesse. Diese werden über Magento-eigene Services angebunden (Bild 3).

Die Ausrichtung auf komplexe, nicht durchgängig standardisierte Handelsstrukturen mit vielfältigen Akteuren setzt zwingend eine hohe Flexibilität und Konnektivität des Magento Commerce Order Managements voraus. Eine der größten Besonderheiten des Magento Commerce Order Managements ist daher die Unabhängigkeit von der eigenen Commerce-Plattform. Die Lösung funktioniert genauso mit Hybris, Websphere oder Demandware wie mit der Magento Enterprise Edition.

Eine regionale Apothekengenossenschaft könnte dann ihren Kunden auch auf der eigenen, gewohnten Plattform die Produkte aus einem Franchise-Netzwerk anbieten – ohne sie selbst vorrätig zu halten. Oder der Kunde könnte in der Franchise-Apotheke an seinem Wohnort ein Rezept einlösen und sich die aktuell nicht vorhandenen Medikamente zur Abholung in einer anderen Filiale bereitstellen lassen, beispielsweise am Ferienort. All das ist vielleicht nicht neu. Es wirtschaftlich realisieren zu können allerdings schon.

Weitere Einsatzszenarien

Das Magento Commerce Order Management als reine Omnichannel-Lösung zu begreifen ist aber zu kurz gedacht. Auf einer Metaebene betrachtet bildet Magento den Warenstrom in einem komplexen Netzwerk ab und versteht dabei jeden Ort, an dem sich eine Einheit befinden kann, als ein Lager. Zur effizienten Steuerung erfasst und verwaltet das System dazu alle möglichen und nötigen Metainformationen (zum Beispiel Ort, Auftraggeber, Order-ID, Datum, Ziel, Auftragshistorie, Prioritäten, Art der Zustellung et cetera).

Letztlich geht es um den Transfer von Bestandsinformationen und darum, was mit dem Bestand geschehen soll. Damit unterstützt das Magento Commerce Order Management aber auch ganz andere Geschäftsmodelle und ermöglicht neue, digitale Services.

Das Anwendungsspektrum reicht von der Unterstützung des Außendienstes, etwa bei der Bestückung von Werkzeugtischen in der Industrie oder der Bevorratung von Laboren, über ein effizientes, automatisiertes Ersatzteilmanagement in der Baubranche (Ersatzteile werden in einem Netzwerk aus Baumaschinen, Baustellen, Bauhöfen, Zentrallagern und Lieferanten transferiert) bis hin zu komplexen Abo-Modellen, in denen der Kunde nur noch Produktgruppen und Konditionen vorgibt. Die Qual der Wahl hat dann das Order Management … und das kommt damit gut zurecht. ■

Ralf Lieser
ist Technical Consultant und Head of Quality Management bei der Netz98 GmbH
www.netz98.de

Developer Newsletter

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

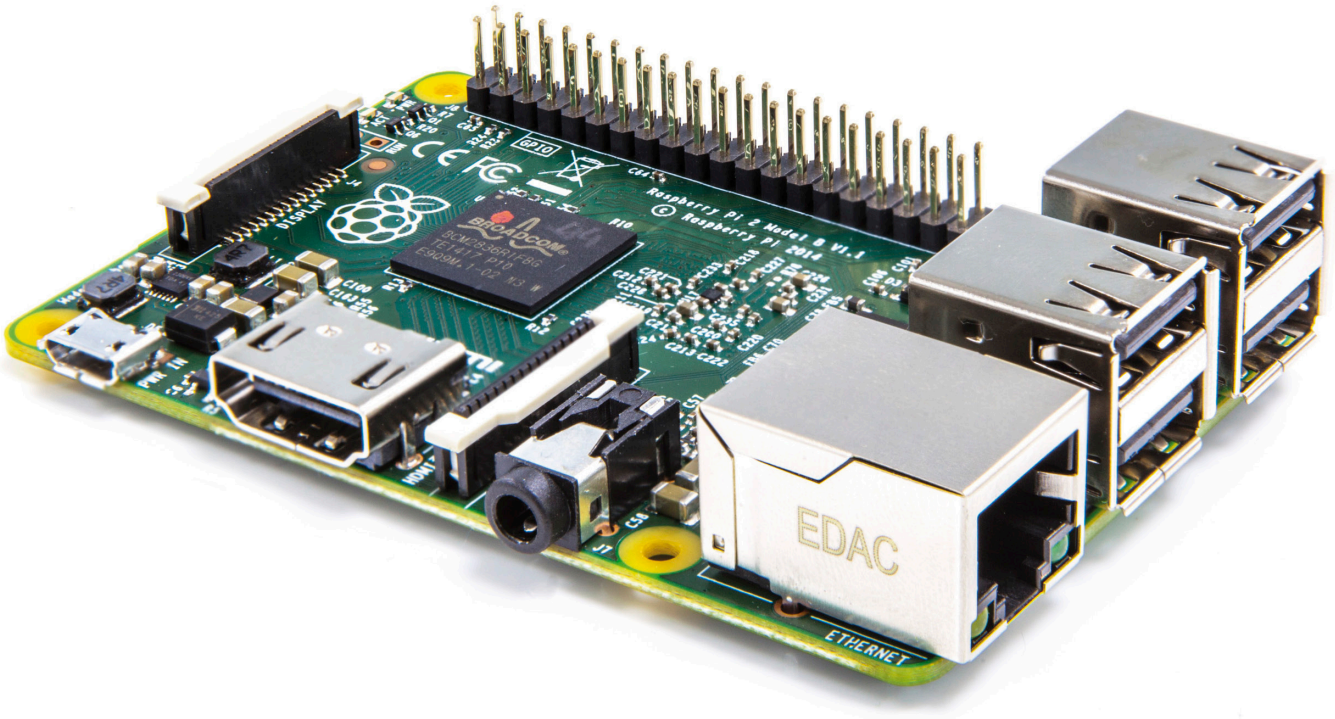
Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile



RASPBERRY PI MIT PHP STEUERN

Webtechnik steuert Hardware

Experimente mit dem Raspberry sind auch für PHP-Entwickler lohnend.

Wenn Sie einen Raspberry Pi besitzen und bereits Erfahrung mit der Programmiersprache PHP haben, möchten Sie Ihre Kenntnisse sicher gerne auch dort nutzen. Das eröffnet eine neue Welt: Sie können per PHP Lichter zum Leuchten bringen, Geräte einschalten oder messen und das Ganze dann per Webtechnik von überall im Haus nutzen. Entsprechende Konfiguration Ihres Routers vorausgesetzt, klappt diese Steuerung sogar aus der Ferne.

Den Raspberry mit PHP auszustatten klappt dank der enthaltenen Werkzeuge genauso einfach wie auf jedem anderen Linux-System. Lediglich die Nutzung von PHP 7 ist einen Tick aufwendiger, da dieses noch nicht in den Standardpaketen der aktuellen Distribution enthalten ist.

Für die Nutzung des GPIO-Systems Ihres Raspberry Pi gibt es noch keine fertig installierbaren PHP-Module aus den Repositories, wie sie etwa für MySQL oder die ZIP-Komprimierung zur Verfügung stehen. Ein gangbarer Weg führt über das Tool *gpio*, das in der C-Bibliothek WiringPi enthalten ist. Damit können Sie von der Kommandozeile aus bequem Ausgänge an- und einschalten:

```
gpio -g mode 18 out
gpio -g write 18 1
```

Mit diesen beiden Kommandos definieren Sie im Beispiel den Pin GPIO 18 als Ausgabekanal und schalten ihn dann ein.

Diese Aufrufe können Sie nun genauso aus PHP heraus vornehmen:

```
system ("gpio -g mode 18 out");
system ("gpio -g write 18 1");
```

Das funktioniert, ist aber fummelig, weil Sie sich die Befehle immer selbst zusammenbauen müssen. Wollen Sie einlesen statt ausgeben, haben Sie dazu auch noch die Rückmeldung des *gpio*-Kommandos auszuwerten.

Viel schöner ist da die Nutzung eines fertigen API. Am besten sollte dieses aber seinerseits nicht den Weg über das Tool *gpio* gehen, weil das relativ aufwendig und darum langsam ist. Darum nutzen wir die PHP-Erweiterung Carica Wires von Thomas Weinert, die einen Wrapper für die Bibliothek WiringPi darstellt. Er verwendet also direkt die C-Library, statt jeweils einen Shell-Aufruf von *gpio* zu starten.

PHP 7 auf dem Raspberry Pi aufspielen

Wir gehen davon aus, dass Sie als Betriebssystem auf Ihrem Raspberry Pi das verbreitete Raspian verwenden. Falls Sie für die PHP-Experimente ein frisches System auf die Speicherkarte installieren, können Sie statt der normalen Version die Lite-Variante nutzen, die auf die grafische Oberfläche verzichtet und darum mit weniger Platz auskommt. Carica Wires verlangt unbedingt nach PHP 7, und angesichts der schwach-

brüstigen Raspberry Pi-CPU ist es auch sinnvoll, mit der neuesten Version von PHP zu arbeiten, die deutlich weniger Ressourcen verbraucht.

Allerdings können Sie das mit dem aktuellen Raspian Jessie nicht einfach so installieren. Erst die nächste Experimentier-Version namens Stretch verfügt über die entsprechenden Pakete. Es ist also ein wenig Zusatzarbeit für die Installation notwendig. Dazu müssen Sie zuerst die Repositories von Stretch einbinden: Öffnen Sie die dafür zuständige Datei mit einem Editor:

```
sudo nano /etc/apt/sources.list
```

Fügen Sie dann am Ende der Datei diese Zeile ein:

```
deb http://mirrordirector.raspbian.org/raspbian/
stretch main contrib non-free rpi
```

Beenden Sie den Nano-Editor mit [Strg X] und bestätigen Sie das Schreiben der Änderungen.

Von nun an würde Ihr Raspberry Pi sich allerdings alle weiteren neuen Pakete von dieser Quelle holen, was Sie sicher nicht möchten. Um das zu korrigieren, öffnen Sie die Datei für die APT-Voreinstellungen im Editor

```
sudo nano /etc/apt/preferences
```

Dort geben Sie die folgenden Kommandos ein:

```
Package: *
Pin: release n=jessie
Pin-Priority: 600
```

Das teilt dem System mit, dass weiterhin Jessie die höchste Priorität haben soll. Das Stretch-Repository wird nur verwendet, wenn Sie dem Kommando *apt-get* explizit dieses Repository als Quelle angeben.

Nun lassen Sie in gewohnter Manier die Liste der Pakete neu einlesen:

```
sudo apt-get update
```

Danach installieren Sie die benötigten Pakete für PHP 7, die Entwicklungstools und en passant auch gleich noch den Webserver Apache:

```
sudo apt-get -t=stretch install php7.0 php7.0-dev
```

Der Zusatzparameter *-t=stretch* bewirkt, dass sich *apt-get* in diesem Fall nicht der normalen Repos bedient, sondern das zuvor hinzugefügte experimentelle Repository verwendet.

So installieren Sie gpio und die WiringPi-Library

Falls Sie es auf Ihrer Workstation noch nicht installiert haben, holen Sie sich zuerst das Tool Git:

```
sudo apt-get install git-core
```

Dann laden Sie sich WiringPi herunter. Danach müssen Sie das Tool noch in gewohnter Art und Weise kompilieren:

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
cd ..
```

Nun sollten Sie nach der Eingabe von *gpio* eine Hilfemeldung erhalten, die Ihnen alle Parameter erklärt.

LED an Raspberry anschließen

Haben Sie bereits eine LED an Ihren Raspberry Pi angeschlossen, können Sie mit diesen Zeilen schon ausprobieren, ob alles klappt (die Pin-Nummer ist eventuell anzupassen):

```
gpio mode 18 out
gpio write 18 1
gpio write 18 0
```

Nun fehlt nur noch die Bibliothek Carica Wiring, also das Verbindungsglied zwischen PHP und WiringPi. Laden Sie dazu die Bibliothek von GitHub herunter:

```
git clone https://github.com/ThomasWeinert/
carica-ext-wires
```

Verwenden Sie folgende Kommandos, um die Übersetzung vorzubereiten, zu kompilieren und die fertige Erweiterungsdatei ins Verzeichnis mit den PHP-Libraries zu installieren:

```
phpize
./configure
make
sudo make install
```

Jetzt müssen Sie nur noch die *php.ini* so erweitern, dass die erzeugte Erweiterung geladen wird:

```
sudo nano /etc/php/7.0/apache2/php.ini
```

Tragen Sie hier am Ende die folgende Zeile nach:

```
extension=wires.so
```

Falls Sie die Erweiterung auch von der Kommandozeile aus verwenden möchten, wiederholen Sie die Definition auch für die Datei */etc/php/7.0/cli/php.ini*.

Nun gilt es noch die Problematik der Zugangsrechte zu lösen. Dazu nehmen Sie zuerst einmal den Apache-User in die Gruppe *gpio* auf:

```
sudo adduser www-data gpio
```

Öffnen Sie mit dem Editor die Datei */etc/apache2/envvars* und fügen Sie dort die Zeile *export WIRINGPI_GPIOMEM=1* hinzu. Dies sorgt dafür, dass die WiringPi-Bibliothek beim ▶

Zugriff von Apache aus die Variante verwendet, die keine Root-Rechte erfordert. Ein Neustart von Apache schließt den Vorgang ab.

Ein Licht aufgehen lassen

Schließen Sie eine LED an den GPIO Ihres Raspberry Pi an. Dazu verbinden Sie zum Beispiel den GPIO Port 18, also den physikalischen Pin Nummer 12, mit der Anode der LED (gekennzeichnet durch ein längeres Beinchen und die kleinere Fläche im Kunststoffkörper).

Das andere Bein, also die Kathode, verbinden Sie mit einem Ground-Pin des Anschlusses, etwa dem physikalischen Pin Nummer 25. Damit der Strom für die LED nicht zu stark wird fügen Sie dabei noch einen Widerstand ein. 470 Ohm sind ein ganz guter Begrenzer für die meisten LED-Typen (Bild 1).

Die Schritte zum Anknipsen der LED nach dem Initialisieren der Bibliothek sind dann identisch zur Methode per Tool gpio: Sie definieren zuerst den gewünschten Pin als Ausgang und legen ihn dann auf High:

```
<?php
use Carica\Gpio\WiringPi;
WiringPi\setup();
WiringPi\pinMode(18, WiringPi\OUTPUT);
```

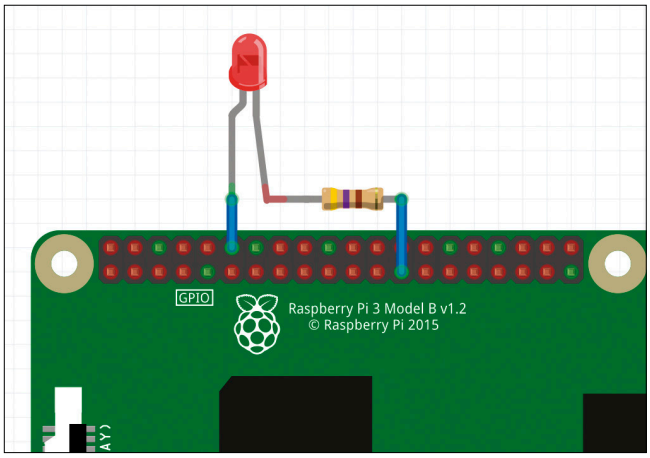
Das Programm können Sie dann schnell so umbauen, dass die LED 5-mal blinkt:

```
<?php
$pin = 18;
use Carica\Gpio\WiringPi;
WiringPi\setup();
WiringPi\pinMode($pin, WiringPi\OUTPUT);

for($i=0;$i<5;$i++){
  WiringPi\digitalWrite($pin, WiringPi\HIGH);
  sleep(1);
  WiringPi\digitalWrite($pin, WiringPi\LOW);
  sleep(1);
}
```

Der Pin 18 kann nun nicht nur als digitaler Ein- und Ausgang, sondern auch für ein Ausgangssignal mit Pulsweitenmodulation (PWM) verwendet werden. Dabei wird am Ausgang ein Rechtecksignal mit fester Frequenz erzeugt, bei dem Sie festlegen können, wie das Verhältnis zwischen High- und Low-Anteilen sein soll. Sie können damit alle Zwischenstufen zwischen den beiden Extremen Dauernull und Dauersignal frei wählen.

Pulsweitenmodulation (PWM)

PWM wird aktiv, indem Sie den Ausgang mit der Funktion *pwmWrite()* setzen. Zur Steuerung des Verhältnisses von Low zu High in der Signalform geben Sie in der Funktion *pwmWrite()* als zweiten Parameter einen Wert von 0 bis 1023 vor. Bei einem Wert von 511 besteht das Signal also aus 50 Prozent Low und 50 Prozent High. Die Pulsbreitensteuerung können Sie zum Beispiel dazu verwenden, um die LED zu dimmen. Tatsächlich wird sie ganz oft ein- und wieder ausgeschaltet. Durch die Trägheit des menschlichen Auges erkennt man das aber nicht, sondern sieht ein entsprechend dem Verhältnis gedimmtes Leuchten.



Eine LED ist schnell an Ihren Raspberry Pi angeschlossen und ermöglicht die ersten Experimente mit PHP (Bild 1)

Das folgende Skript erzeugt ein stetiges An- und Abschwellen der LED:

```
<?php
$delay=2000;
$pin = 18;

use Carica\Gpio\WiringPi;
WiringPi\setup();
WiringPi\pinMode($pin, WiringPi\PWM_OUTPUT);

for ($j=0;$j<5;$j++)
{
  // ansteigende Helligkeit
  for ($i=10;$i<512;$i++) {
    WiringPi\pwmWrite($pin, $i);
    usleep ($delay);
  }
  // abnehmende Helligkeit
  for ($i=512;$i>=10;$i--) {
    WiringPi\pwmWrite($pin, $i);
    usleep($delay);
  }
}
// ganz aus
WiringPi\pwmWrite($pin,0);
```

Mit den Funktionen für den PWM-Modus können Sie also das Fehlen eines echten D/A-Wandlers im Raspberry Pi ausgleichen – zumindest für bestimmte Peripherie wie LEDs oder Motoren, bei denen die Taktung des Signals keine störenden Nebeneffekte hat. Mit den 1024 Stufen lassen sich diese angeschlossenen Verbraucher recht fein regeln.

Auch der umgekehrte Weg ist möglich, also die Verwendung eines der Anschlüsse als Eingang. Mit folgendem Skript geben Sie den aktuellen Zustand *HIGH* oder *LOW* aus, der am GPIO 17 (physikalischer Pin 11) anliegt, also dem Nachbaranschluss des zuvor benutzten GPIO 18:

```
<?php
// input.php
$pin = 17;
use Carica\Gpio\WiringPi;
WiringPi\setup();

WiringPi\pinMode($pin, WiringPi\INPUT);

$cond = WiringPi\digitalRead($pin) ? "HIGH" : "LOW";
echo "Zustand an Pin $pin = $cond";
```

Zum Ausprobieren ist es wahrscheinlich besser, das Skript von der Kommandozeile aus zu starten und die letzten beiden Zeilen in eine Endlosschleife zu packen. Dann erhalten Sie bis zum Abbruch über [Strg C] den aktuellen Zustand, ohne den Browser neu laden zu müssen. Dazu müssen Sie dann aber das Skript per *sudo* starten.

So testen Sie den Aufbau mit Bordmitteln

Wollen Sie einfach nur schnell testen, ob eine am Raspberry Pi angeschlossene LED zum Leuchten gebracht werden kann, müssen Sie nicht die ganze Installation nachvollziehen, die hier erklärt wird. Schon das Grundsystem Raspian selbst ist dazu in der Lage. Dazu schreiben Sie per *echo*-Befehl in die Pseudo-Verzeichnisse unter */sys/class/gpio*, die als Schnittstelle zum Eingabe-/Ausgabe-Port des Raspberry Pi dienen.

Wir nehmen an, Sie haben eine LED an den GPIO 18 angeschlossen. Zuerst müssen Sie per Schreiben auf den *export*-Pfad den Dateizugang für den gewählten Port anlegen:

```
sudo echo 18 > /sys/class/gpio/export
```

Nun richten Sie diesen Port für die Ausgabe ein:

```
sudo echo out > /sys/class/gpio/gpio18/direction
```

Das Schreiben von *1* auf den *value*-Pfad setzt den Ausgang unter Strom:

```
sudo echo 1 > /sys/class/gpio/gpio18/value
```

Und mit folgendem Kommando wird die LED wieder dunkel:

```
sudo echo 0 > /sys/class/gpio/gpio18/value
```

Nach einem Neustart des Raspberry Pi ist der Pfad */sys/class/gpio/gpio18* übrigens nicht mehr da. Sie müssen also alle Befehle wiederholen.

Rund um die Benennung der Anschlüsse des GPIO-Steckers gibt es leider eine verwirrende Vielfalt von Bezeichnungen. Der Grund dafür sind verschiedene Sichtweisen: Am besten nachvollziehbar ist die physikalische Benennung: Die 40 Anschlüsse werden einfach durchgezählt: Pin 1 ist der ganz links unten, Pin 2 der darüber und so weiter. Das gebräuchlichste Namensschema, dem wir auch im Beitrag folgen, orientiert sich an der originalen Benennung der Chipausgänge des Herstellers Broadcom (BCM), und nennt den physikalischen Pin 12 beispielsweise GPIO 18.

Benennung der Anschlüsse des GPIO-Steckers

Der Autor der Bibliothek WiringPi hat auch noch sein eigenes Süppchen gekocht und sich von der einsteigerfreundlichen Logik der Benennung der Arduino-Ports inspirieren lassen. Dabei werden die nutzbaren Interface-Anschlüsse der Reihe nach durchgezählt. Der erste verfügbare Port bekommt so den Namen *0* (physikalischer Pin 11, GPIO 17).

Eine gute Übersicht über die Details der Belegung und den aktuellen Zustand der Ports liefert das *gpio*-Tool, wenn Sie es mit *readall* als Parameter aufrufen. Dann erhalten Sie eine

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | OUT  | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

Das Tool gpio liefert Ihnen den aktuellen Zustand aller Ports des GPIO und zeigt auch gleich alle drei Benennungsvarianten für die Pins an (Bild 2)

Übersicht aller Anschlüsse zusammen mit den verschiedenen Benennungen und dem aktuellen Zustand.

In Bild 2 sehen Sie etwa, dass BCM 18, in der üblichen Bezeichnung ist das eigentlich GPIO 18, derzeit als Ausgang definiert ist und den Wert *0* hat, also aus ist. Mit *gpio* können Sie alle drei Benennungssysteme verwenden. Standardmäßig interpretiert es die an ihn übergebene Nummer im WiringPi-System. Mit dem Parameter *-g* wird dagegen das übliche GPIO-System verwendet. Und Sie können auch mit den physikalischen Nummern arbeiten. Dazu geben Sie den Parameter *-1* mit. Folgende drei Varianten bewirken also dasselbe:

```
gpio -g write 18 1
gpio -1 write 12 1
gpio write 1 1
```

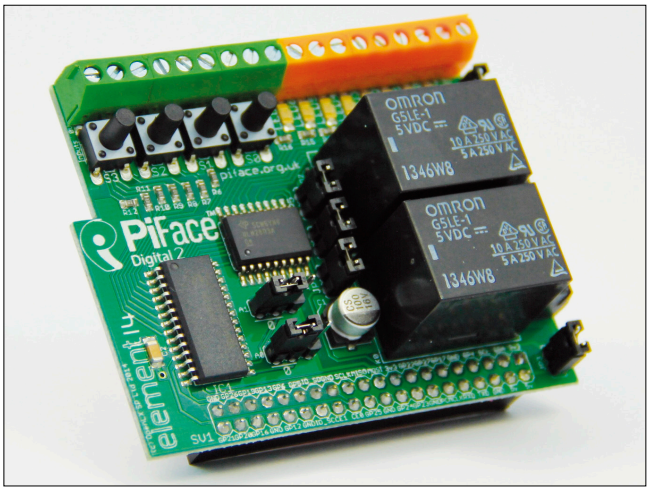
Wenn Sie mit dem Raspberry Pi nicht nur ein wenig experimentieren, sondern ernsthaft steuern und schalten wol- ▶

len, dann werden die beschränkten Möglichkeiten der Hardware schnell zur Spaßbremse.

Investieren Sie noch einmal rund 30 Euro, dann bekommen Sie mit dem Piface Digital 2 eine Huckepackplatine, die viele neue Wege zur Kommunikation mit der Außenwelt bietet.

Das Zusatzboard verfügt über zwei Relais, mit denen Sie auch 230-Volt-Verbraucher mit einer maximalen Last von 5 Ampere schalten können, dazu vier eingebaute Taster, acht digitale Eingänge und acht Open-Collector-Ausgänge mit je einer Status-LED. Alle Aus- und Eingänge sind mit Lüsterklemmen-Leisten verbunden, womit sich externe Geräte recht einfach anschließen lassen.

Nur in einem Punkt muss die Platine passen: Sie hat keinen eingebauten A/D-Wandler zum Erfassen von analogen Signalen (Bild 3).



Mit der Erweiterungsplatine Piface Digital 2 öffnen Sie Ihren Raspberry Pi noch einfacher für die Außenwelt (Bild 3)

Das Problem mit den Root-Rechten

Die Verknüpfung der Hardware des Raspberry Pi mit dem Betriebssystem sieht vor, dass nur der *root*-Benutzer Zugriff darauf hat.

Das Tool *gpio* funktioniert aber auch, wenn man nur als Standarduser *pi* eingelogt ist. Dass das klappt, liegt daran, wie dessen Installer die Datei */usr/local/bin/gpio* angelegt hat, die bei der Eingabe des Kommandos ausgeführt wird. Sehen Sie sich deren Attribute genau an:

```
-rwsr-xr-x 1 root root
```

Das gesetzte SUID-Bit – erkennbar am *s* an der vierten Attributstelle – sorgt dafür, dass jede Ausführung der Datei mit den Rechten des Owners erfolgt, hier also *root*. Ganz gleich wer also die Datei ausführt, sie erhält dabei Root-Rechte.

Wenn Sie aus einem PHP-Programm auf dem Webserver heraus versuchen, eine Funktion der Bibliothek WiringPi aufzurufen, ohne für die notwendigen Rechte gesorgt zu haben, erhalten Sie im Errorlog die folgende Meldung:

```
wiringPiSetup: Must be root. (Did you forget sudo?)
```

Im Kontext eines Webservers hilft der Hinweis auf *sudo* nur wenig weiter – die Bibliothek bringt den Fehler einfach, ohne auf die Art des Aufrufs Rücksicht zu nehmen.

Die Holzhammer-Methode gegen dieses Problem wäre es, den Apache nicht mit den Rechten des Benutzers *www-data*, sondern als *root* auszuführen. Aber dann ist Ihr Raspberry Pi nur einen Klick davon entfernt, gekapert zu werden.

Dass der Zwang zu Root-Rechten seinen Besitzern Steine in den Weg legt, haben auch die Entwickler des Systems eingesehen und darum alternativ ein Interface zur Hardware im Dateisystem unter */dev/gpiomem* eingerichtet, das normalen Usern den Zugang erlaubt. Dann ist es nur noch notwendig, dass der User *www-data* zu zur Linux-Gruppe *gpio* hinzugefügt wird, weil die Zugriffe auf */dev/gpiomem* an diese Gruppenzugehörigkeit gebunden sind.

Damit die Bibliothek WiringPi anstelle des normalen Wegs den über */dev/gpiomem* geht, muss man ihr das durch das Setzen der Environment-Variablen *WIRINGPI_GPIOMEM* auf den Wert *1* mitteilen.

Ob ein älteres System die Methode per */dev/gpiomem* unterstützt, kann Ihnen das *gpio*-Tool sagen. ■

Links zum Thema

- Zentrale Anlaufstelle für die Raspian-Distribution und viele Infos rund um den Mini-Computer
  http://raspberrypi.org
- Repository für die PHP-Erweiterung Carica Wires
  http://github.com/ThomasWeinert/carica-ext-wires
- Homepage der WiringPi-Bibliothek, die das Kommandozeilen-Programm gpio mitbringt
  http://wiringpi.com
- Heimat der Erweiterungsplatine mit Relais, LEDs und mehr Anschlüssen
  www.piface.org.uk
- Raspberry Pi als Webserver
  http://raspberrywebserver.com
- Raspberry Pi-Sensoren auslesen
  www.laub-home.de/wiki/Raspberry_Pi_Sensoren_auslesen

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

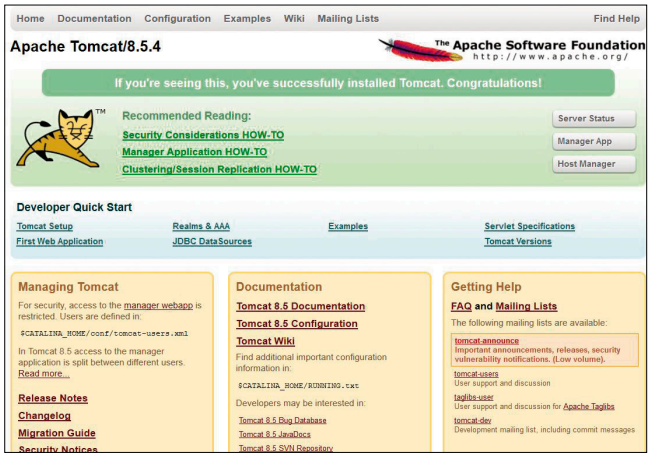
Ein Magazin der Verlagsgruppe Ebner Ulm • www.ebnerverlag.de

APACHE TOMCAT IM PRODUKTIVEN EINSATZ

Langlebige Katze

Bei Java-Webanwendungen zählt Apache Tomcat zu den beliebtesten Applikationsservern.

Der Apache Tomcat Webserver gehört seit 17 Jahren zu den am meisten eingesetzten Systemen im Bereich Java-Anwendungsserver. Ursprünglich wurde Tomcat als Servlet-Container-Referenzimplementierung JServ der Apache Foundation von Sun übergeben, unter deren Lizenz und Dach er bis heute steht. Er unterstützt die IETF- und Java-Webstandards, wie HTTP, Java Servlet 3.1, JavaServer Pages 2.3, Java Unified Expression Language 3.0 und WebSocket 1.1.



Startseite: So präsentiert sich nach dem Aufruf die Tomcat-Einstiegsseite (Bild 1)

Um den aktuellen Apache Tomcat 8.5.x zu verwenden, benötigen wir mindestens eine Java-8-Laufzeitumgebung. Am besten verwendet man Java 8, da dies diese mehr Sicherheit, Performance und vor allem Support bietet. Java 9 und damit die HTTP/2-Unterstützung wird erst Mitte des nächsten Jahres verfügbar sein. Sie kann jedoch bereits mit den Entwicklerversionen von Tomcat 9 ausprobiert werden. Da die bis Ende des Jahres erscheinende Tomcat-6.0-Version nach zehn Jahren nicht mehr unterstützt wird, sollten bisherige Nutzer oder Neueinsteiger die Version 8.5 verwenden.

Installation

Nach dem Herunterladen der für das jeweilige Betriebssystem (Unix/Windows) passenden Version setzen Sie die *JRE_HOME* und müssen die JVM in *PATH* haben. Dann können Sie Apache Tomcat mit *%CATALINA_HOME%\bin\startup.bat* starten. Standardmäßig ist Tomcat dann über *http://localhost:8080* erreichbar. Dann erscheint die in Bild 1 gezeigte Startseite, die die wichtigsten Informationen inklusive Links an einer Stelle präsentiert.

Die wichtigste Datei zur Server-Konfiguration, *server.xml,* befindet sich im *conf*-Verzeichnis. Die Protokolle landen im *log*-Verzeichnis und die Skripts zum Starten und Stoppen befinden sich im *bin*-Verzeichnis. Die Anwendungen selbst liegen im *webapps*-Verzeichnis. Standardmäßig wird Tomcat mit der Dokumentation, Beispielanwendungen, der virtuellen Hostmanager-Anwendung und der Manager-Anwendung ausgeliefert. Die wichtigsten Verzeichnisse und typische Nutzer von Tomcat sind in Bild 2 dargestellt.

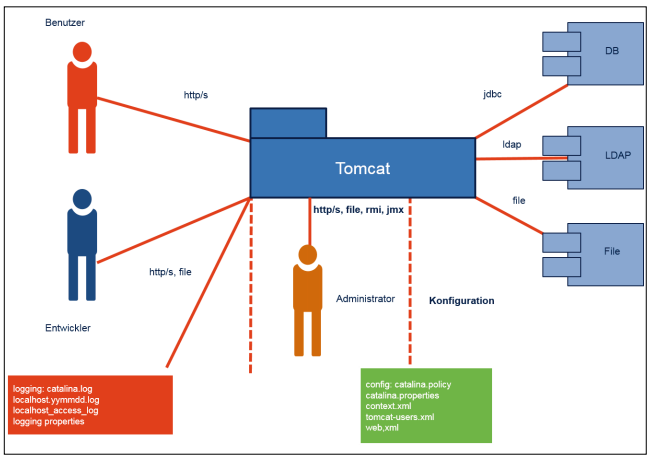
Für den produktiven Einsatz stellen sich spezielle Anforderungen. Deswegen muss die Standardinstallation daran angepasst werden (Bild 3).

Als Erstes entfernt man die nicht benötigten Anwendungen im *webapps*-Verzeichnis. Außerdem wird man nur die wirklich benötigten Ports freischalten. Bevor man die *server.xml*-Datei editiert, sollte man die wichtigsten Tomcat-Komponenten und ihre Einstellungen kennen (Bild 4).

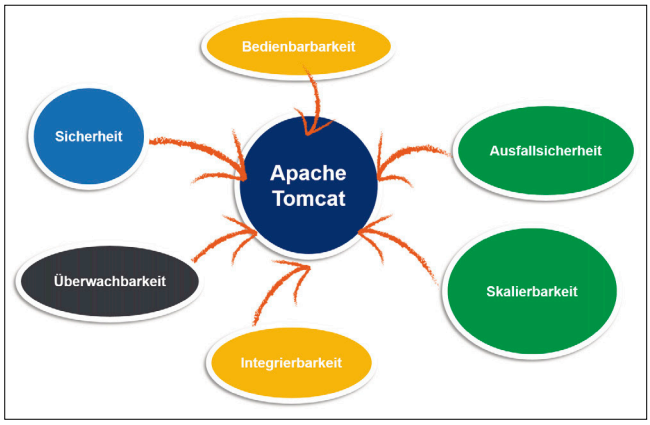
Dateibasierte Benutzerverwaltung

So ist dort über den *UserDatabaseRealm* eine einfache dateibasierte Benutzerverwaltung konfiguriert. In der referenzierten Datei *tomcat-users.xml* müssen Sie zur Verwendung der Manager-Administrationsanwendung, die Sie mit *http://localhost:8080/manager* aufrufen, einen Benutzer mit Passwort und den Rollen *manager-status und manager-gui <user username="tomcat" password="geheim" roles="manager-status,manager-gui" />* hinterlegen.

So können Sie sich zum Beispiel mit *curl --user tomcat:geheim --url http://localhost:8080/manager/status/all?XML=true* den gesamten Server-Status als Text ausgeben lassen, um diesen dann mit einem Skript regelmäßig auszuwerten. Dieser Weg



Verzeichnisse und typische Nutzer von Tomcat (Bild 2)



Anforderungen an einen produktiven Tomcat-Server (Bild 3)

wird sehr oft auch von dem Nagios-Modul *check_tomcat.pl* gegangen.

Skalierbar und ausfallsicher

Die wichtigsten Parameter für die Performance eines Applikationsservers sind genügend Hauptspeicher und die passende Größe der Pools, um schnell Anfragen oder Verbindungen bedienen zu können, ohne immer wieder neue erstellen zu müssen. Beim HTTP-Connector ist wichtig, dass das Http-11Nio2Protocol verwendet wird. Zusätzlich können die Parameter *maxKeepAliveRequests="1" connectionTimeout="2000"* und *maxThreads="400"* mit ausreichenden Größen gesetzt werden. Tomcat 8 verwendet die Apache Commons DBCP 2.x für das Pooling von Datenbankverbindungen, der auch JDBC 4.0 unterstützt.

So kann in der Datei *context.xml* eine *DataSource* mit folgenden Parametern angelegt werden:

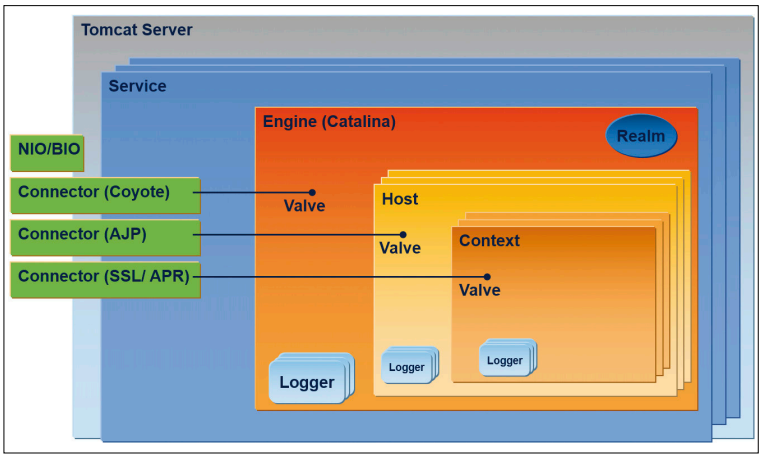
```
<Context>
  <Resource name="jdbc/db"
    auth="Container"
    type="javax.sql.DataSource"
    maxTotal="150"
    maxIdle="20"
    maxWaitMillis="-1"
    testOnBorrow="true"
    removeAbandonedOnBorrow="true"
    removeAbandonedOnMaintenance="true"
    username="USER"
    password="GEHEIM"
    driverClassName="oracle.jdbc.OracleDriver"
    url="jdbc:oracle:thin:@HOST:1521:PORT"
    validationQuery="SELECT 1 FROM DUAL"
  />
```

Seit der Java Garbage Collector G1 von Java 8 standardmäßig verwendet wird, ist der Tuningaufwand für die virtuelle Maschine geringer. Trotzdem ist es sinnvoll, einige Parameter zu setzen, wie *-XX: +UseLargePages, -XX: +UseStringDeduplication* und *-XX: +ExitOnOutOfMemoryError*. Ebenso sollten die Parameter *-Xms4G -Xmx4G* für die Speichergrößen des JVM-Heaps auf dieselben Oberwerte gesetzt werden. Es hat sich bewährt, die Anzahl der parallelen GC-Threads auf die Anzahl der verfügbaren CPU-Kerne minus 1 zu setzen. Also bei einem 8-Kern-Prozessor auf *-XX: ParallelGCThreads=7*.

Um die Protokollierung zu beschleunigen und zu verhindern, dass diese im Fehlerfall die Anwendung ausbremst, sollte man zumindest einen asynchronen Logger in der Datei *conf/logging.properties* verwenden und dabei die nicht verwendeten Logger entfernen:

```
handlers = 1catalina.org.apache.juli.AsyncFileHandler
.handlers = 1catalina.org.apache.juli.AsyncFileHandler
```

Am besten legt man das Protokollverzeichnis auch auf eine separate Platte, sodass hier noch weniger Seiteneffekte für die Anwendung auftreten können. Statt des JULI-Loggers kann man auch Log4j verwenden, indem man die Datei *log4j2.xml* zusammen mit den *log4j-core-2.6.2.jar* und *log4j-api-2.6.2.jar* ins Verzeichnis *%CATALINA_BASE%/lib* kopiert. Seit Tomcat 8.5 wird nur noch Log4j2 und nicht mehr sein Vorgänger unterstützt. Log4j hat den Vorteil, dass es sehr viele Einstellmöglichkeiten gibt, etwa das JsonLayout zum Formatieren der Logs im leicht auswertbaren JSON-Format.



Konfiguration: Die wichtigsten Tomcat-Komponenten (Bild 4)

Um Tomcat produktiv zu betreiben, sollten wichtige Komponenten redundant ausgelegt werden. Ein bewährtes Mittel ist hier ein Cluster mit mindestens zwei Knoten und einem vorgeschalteten Lastverteiler. Neben einer besseren Skalierbarkeit erreicht man damit eine höhere Ausfallsicherheit. Als schöner Nebeneffekt verbessert sich auch die Wartbarkeit und Anpassbarkeit.

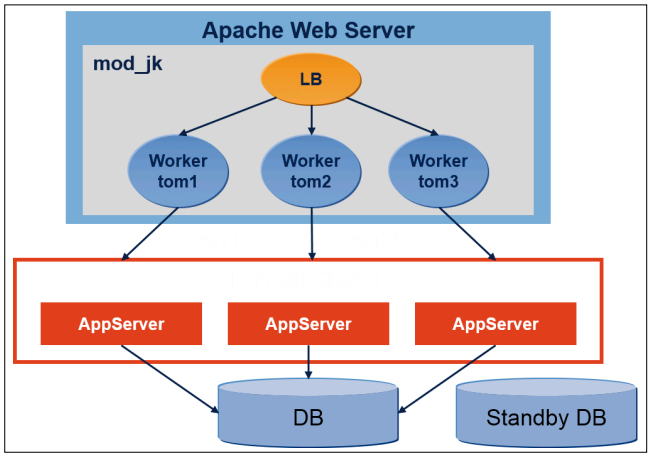
So ist es zum Beispiel möglich, einen Tomcat-Knoten zu aktualisieren, ohne die Gesamtverfügbarkeit des Systems zu gefährden. Hilfreich bei Tuning-Maßnahmen ist es auch, nur für einen Knoten die entsprechenden Parameter anzupassen und zu beobachten, wie dieser sich im Unterschied zu dem nicht angepassten Knoten verhält. ▶

Eine typische Cluster-Konfiguration mit einem HTTPD-Webserver als Lastverteiler und zwei Tomcat-Knoten zeigt Bild 5.

Als Protokoll kommt entweder HTTP oder das Tomcat-eigene AJP (Apache JServ Protocol) zur Auswahl. Meist wird man das AJP-Protokoll verwenden, da dieses mehr Konfigurations- und Überwachungsmöglichkeiten bietet und außerdem als binäres Protokoll schneller als das textbasierte HTTP-Protokoll ist. Nachdem man das Modul *mod_jk* installiert hat, konfiguriert man es in der *httpd.conf*-Datei:

```
LoadModule    jk_module  modules/mod_jk.so
JkWorkersFile C:\tomcat\config\workers.properties
JkOptions +ForwardKeySize +ForwardURICompat
-ForwardDirectories
```

Die Worker für die einzelnen Knoten *node01* und *node02* werden in der Datei *workers.properties* konfiguriert.



Tomcat-Cluster mit zwei Knoten (Bild 5)

```
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers= node01,node02
worker.loadbalancer.sticky_session=True
worker.loadbalancer.requiredSecret="secret key word"
worker.loadbalancer.retries=1
worker.loadbalancer.method=B
worker.node01.type=ajp13
worker.node01.port=8009<ajp port>
worker.node01.host=<tomcat ip addr>
worker.node01.lbfactor=1
worker.node02.type=ajp13
worker.node02.port=8009<ajp port>
worker.node02.host=<tomcat ip addr>
worker.node02.lbfactor=1
```

Der Port 8009 bezieht sich auf dem AJP-Connector in der *server.xml* des Tomcat: *<Connector port="8009" protocol="AJP/1.3" requiredSecret="secret key word" redirectPort="8443" />*. Die Verbindung zwischen dem Load Balancer und dem Tomcat-Knoten ist über ein vorher ausgetauschtes Geheimnis, das bei jeder Verbindung vorab mitgeschickt wird, zusätzlich abgesichert. HTTP/2 verspricht zusammen mit TCP-FAST-Open, Verbindungen effizienter zu nutzen. Wer möchte, kann den experimentellen HTTP/2-Support mit OpenSSL und ALPN auch in Tomcat 8.5 ausprobieren. Dazu muss HTTP/2 in der HTTP-Connector-Konfiguration mit *<UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />* aktiviert werden.

Webanwendungen beschleunigen

Um die Auslieferung von Webressourcen zu beschleunigen, gibt es viele Möglichkeiten. Verwendet die Webanwendung noch JSPs, so werden durch ein Vorkompilieren der Seiten nicht nur Fehler früher erkannt und behoben, sondern auch der erste Aufruf der Seite beschleunigt.

Mit den folgenden geänderten Parametern in der Datei *conf\web.xml* können HTML-Seiten aus JSP schneller und kleiner ausgeliefert werden. Jedoch sollte besonders beim Parameter *trimSpaces* vorher getestet werden, ob die erzeugten Class-Dateien im Browser tatsächlich korrekt angezeigt werden:

```
<servlet>
  <servlet-name>jsp</servlet-name>
  <servlet-class>org.apache.jasper.servlet.JspServlet
  </servlet-class>
  <init-param>
    <param-name>development</param-name>
    <param-value>false</param-value>
  </init-param>
  <init-param>
    <param-name>trimSpaces</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>genStringAsCharArray</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>compilerTargetVM</param-name>
    <param-value>1.8</param-value>
  </init-param>
  <init-param>
    <param-name>mappedfile</param-name>
    <param-value>false</param-value>
  </init-param>
```

Eine weitere Möglichkeit, die Auslieferung großer Dateien zu beschleunigen, ist die Komprimierung. Im Connector in der Datei *conf\server.xml* können folgende Parameter ergänzt werden, wenn sich nicht ein separater Proxy- oder Content-Cache-Server bereits darum kümmert:

```
<Connector port="8080" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false" redirectPort="8443"
```

```
acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true"
compression="on"
compressionMinSize="2048"
noCompressionUserAgents="gozilla, traviata"
compressableMimeType=
  "text/csv,text/html,text/xmlapplication/
  json,application/pdf"/>
```

Dabei ist es sinnvoll, die Komprimierung nur auf große Dateien bestimmter Dateitypen einzuschränken, da sich nur dort eine Komprimierung lohnt. Eine Komprimierung spart zwar Übertragungskapazität, geht jedoch mit einer höheren CPU-Belastung einher.

Da Tomcat oft nicht alleine, sondern beispielsweise mit einem vorgeschalteten Load Balancer oder mit Content Delivery Networks (CDN) eingesetzt wird, kann es in vielen Fällen sinnvoll sein, diese Spezialisten für die schnelle Auslieferung von statischen Inhalten zu verwenden. Damit Webcrawler nicht zu viele Sitzungen erzeugen, kann die Valve *CrawlerSessionManagerValve* verwendet werden und in der *server.xml* hinzugefügt werden:

```
<Valve className="org.apache.catalina.valves.Crawler
SessionManagerValve"
crawlerUserAgents=".*googlebot.\*|.*yahoo.*"
sessionInactiveInterval="600"/>
```

Sicherheit beginnt schon bei der Installation. So sollten nur die nötigsten Dateien installiert und nur die wirklich benötigten Einstellungen konfiguriert werden.

Die verwendete Installationsdatei sollte immer überprüfen, ob diese nicht später verändert wurde. Das kann man einfach mit der md5-Hashsumme verifizieren, nämlich über *md5sum -c apache-tomcat-8.5.5.zip.md5*. Außerdem sollte man alle kritischen Komponenten (Tomcat, Java, JDBC-Treiber) in aktuellen Versionen einsetzen und auch regelmäßig aktualisieren. Um auf dem Laufenden zu bleiben, sollte man die Bekanntgabe-Mailingliste, Sicherheitsbekanntgaben, die Releasenotes oder das Changelog des Tomcat-Projekts regelmäßig lesen.

Einfache Pflege

Um eine einfachere Pflege zu erreichen, sollten die Installationsdateien in zwei separate Verzeichnisse *CATALINA_HOME* (*bin* für Startup- und Shutdown-Skripts und *lib* für die Bibliotheken) und *CATALINA_BASE* (für den Rest: *conf, logs, webapps, bin/setenv.bat)* abgelegt werden. So kann Tomcat oder der Datenbanktreiber separat aktualisiert werden, ohne die Konfiguration zu beeinflussen.

Grundsätzlich können auch alte Konfigurationen verwendet werden. Diese sollten jedoch bei Hauptversionswechseln besser überprüft werden, da sich hier manchmal Parameternamen und Standardwerte ändern können. Um nicht zu viele Informationen preiszugeben und dadurch Angriffe zu erleichtern, sollte die Tomcat-Version nicht im Header oder bei Fehlerseiten angezeigt werden. Dazu sollten Sie einerseits bei den verwendeten Connectoren das Attribut *server* auf einen festen Text, zum Beispiel *Apache,* setzen: *<Connector port= "8080" ... server="Apache" />*. Deaktivieren Sie zudem im Fehlerbericht die Weitergabe der Produktversion mit *<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport="false" showServerInfo= "false"/>*.

Nicht erst seit Heartbleed oder Poodle wird auf die Verwendung einer korrekten Verschlüsselung Wert gelegt. HTTPS-Verbindungen werden durch die Let's-Encrypt-Initiative und auch durch ein besseres Ranking der Seiten bei Google gefördert. Bei Tomcat gibt es zwei Varianten: Entweder verwendet man OpenSSL mit der Apache Tomcat Native Library, die auf der Apache Portable Runtime (APR) basiert, oder man ▶

Links zum Thema

- Download Tomcat 8.5
  http://tomcat.apache.org/download-80.cgi
- Änderungen
  http://tomcat.apache.org/tomcat-8.5-doc/changelog.html
- Versionsübersicht
  http://wiki.apache.org/tomcat/TomcatVersions
- End of life for Apache Tomcat 6.0
  http://tomcat.apache.org/tomcat-60-eol.html
- Monitoring and Managing Tomcat
  http://tomcat.apache.org/tomcat-9.0-doc/monitoring.html
- Tomcat-8.5-Sicherheitshinweise
  http://tomcat.apache.org/security-8.html
- TLS-Chiffrensammlungen
  http://wiki.apache.org/tomcat/Security/Ciphers
- SSL Server Test
  https://www.ssllabs.com/ssltest
- CIS Apache Tomcat 8 Benchmark v1.0.1
  https://benchmarks.cisecurity.org/en-us/?route=downloads.show.single.tomcat8.101
- Migrating from 8.0 to 8.5
  http://tomcat.apache.org/migration-85.html
- Unterstützte Standards
  http://wiki.apache.org/tomcat/Specifications
- Apache Tomcat 8 Configuration Reference
  https://tomcat.apache.org/tomcat-8.5-doc/config
- JavaMelody
  https://javamelody.googlecode.com
- SSLTest-Client
  http://wiki.apache.org/tomcat/tools/SSLTest.java

verwendet die Standard Java Secure Socket Extension (JSSE). Deren Konfigurationsparameter wurden in Version 8.5 mit dem SSLHostConfig-Element aneinander angeglichen. Da Java standardmäßig mit schwachen Schlüsseln ausgeliefert wird, muss man noch die Unlimited-Strength-Jurisdiction-Policy-Dateien mit den starken Schlüsseln von Oracle herunterladen und die Dateien *local_policy.jar* und *US_export_policy.jar* in das Verzeichnis *jre/lib/security* kopieren.

SSL-Konfiguration

Bei den zu verwendenden Chiffrensammlungen sollte man für Java 8 *HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!DHE* und für OpenSSL *HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!kRSA* angeben. Damit schafft man es zum Beispiel für den SSL Server Test, das Rating von B auf A zu verbessern.

Eine SSL-Konfiguration für HTTPS, was bei HTTP/2 für Browser der Standard ist, sieht dann wie folgt aus:

```
<Connector port="8443" protocol=
"org.apache.coyote.http11.Http11AprProtocol"
maxThreads="150" SSLEnabled="true" >
  <UpgradeProtocol className=
  "org.apache.coyote.http2.Http2Protocol" />
  <SSLHostConfig honorCipherOrder="true"
  ciphers="EECDH+ECDSA+AESGCMEECDH+aRSA+ECDSA+SHA256EE
  CDH+aRSA!aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP
  !DSS" protocols="TLSv1.2">
    <Certificate certificateKeyPassword="test"
    certificateKeyFile="bin\key.pem"
    certificateFile="bin\cert.pem"
    type="RSA" />
  </SSLHostConfig>
</Connector>
```

Für HTTP/1.1 und JSSE sieht eine Konfiguration mit dem neuen SSLHostConfig-Element so aus:

```
<Connector port="8443" protocol=
"org.apache.coyote.http11.Http11Nio2Protocol"
SSLEnabled="true" maxThreads="150" >
  <SSLHostConfig honorCipherOrder="true" ciphers=
  "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
  TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384"
  protocols="TLSv1.2">
    <Certificate  certificateKeystoreFile=
    "bin\keystore" certificateKeystorePassword=
    "changeit" certificateKeyAlias="tomcat" />
  </SSLHostConfig>
</Connector>
```

Um zu überprüfen, ob die Konfigurationsdatei korrekt ist, rufen Sie *configtest.bat* auf.

Welche Chiffren aktuell sind, können Sie mit *http://localhost:8080/manager/text/sslConnectorCiphers* oder mit dem kleinen Testprogramm aus dem Tomcat-Wiki *java SSLTest localhost:8443* erfahren.

Webanwendungen absichern

Neben dem Webserver müssen auch die Webanwendungen abgesichert werden. Eine einfache Möglichkeit, CSRF-Angriffe (Cross-Site Request Forgery) zu verhindern, ist die des zwischengeschalteten, Token-basierten CsrfPreventionFilters. Dieser ist bereits für die Verwaltungsanwendung Manager aktiviert und kann auch für die eigenen Webanwendungen in der Datei *WEB-INF\web.xml* aktiviert werden

```
<filter>
  <filter-name>CSRF</filter-name>
  <filter-class>org.apache.catalina.filters.
  CsrfPreventionFilter</filter-class>
  <init-param>
    <param-name>entryPoints</param-name>
    <param-value>/html,/html/,/html/list,/index.jsp
    </param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>CSRF</filter-name>
  <servlet-name>HTMLManager</servlet-name>
  <servlet-name>jsp</servlet-name>
</filter-mapping>
```

Weitere Möglichkeiten, um Tomcat bei Bedarf noch sicherer zu machen, finden sich im CIS Apache Tomcat 8 Benchmark v1.0.1.

Ausblick

Trotz Trends wie Microservices und Serverless-Anwendungen gehört Tomcat für Java-Webanwendungen noch lange nicht zum alten Eisen. Schließlich verrrichtet er zuverlässig in vielen kritischen Anwendungen seit über 17 Jahren seinen Dienst.

Durch seine langen Supportzyklen und die gute Abwärtskompatibilität ist ein Wechsel auf die neueste Version ohne große Probleme möglich. Vor allem die Nutzer der Version 6 sollten auf eine neuere Version wie 8.5 wechseln, da sie anderenfalls keine Fehlerkorrekturen mehr erhalten.

Parallel zum Erscheinen von Java 9 wird dann Tomcat 9 in einer produktiven Version verfügbar sein, um vor allem die Möglichkeiten der neuen Java-Version Mitte 2017 nutzen zu können. Katzen wird ja oft ein langes Leben nachgesagt. Bei Apache Tomcat trifft das sicher zu. ■

Frank Pientka
ist Principal Software Architekt bei der Materna GmbH in Dortmund. Er beschäftigt sich seit mehreren Jahrzehnten mit Java-Enterprise-Webanwendungen und ist Autor eines Buches zu Apache Geronimo.

developer media

developer media Trainings

Updates für Ihr Know-How

„SharePoint? Kann weit mehr als bloß Dokumente verwalten!“

Martin Groblschegg
SharePoint-Entwickler,
Berater und Trainer

Moderne Webentwicklung mit ASP.NET Core

Trainer: David Tielke

3 Tage, 21.-23.11.2016, Köln
Ab 2.199,- EUR zzgl. MwSt.

SharePoint Search im praktischen Einsatz

Trainer: Martin Groblschegg

2 Tage, 24.-25.11.2016, Köln
Ab 1.799,- EUR zzgl. MwSt.

Add-In Development für SharePoint 2016 und Office 365

Trainer: Martin Groblschegg

3 Tage, 13.-15.12.2016, 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.

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

GRAFIK FÜR ENTWICKLER

Ebenen und Masken

Neben den Auswahlwerkzeugen helfen Ebenen und Masken bei der professionellen Bildbearbeitung.

Adobe Photoshop CC 2015 liefert umfangreiche Werkzeuge und Funktionen, um Bildobjekte auszuwählen, diese freizustellen und zu bearbeiten. Der vorige Teil dieser Serie hat die unterschiedlichen Methoden zum Freistellen beleuchtet. Hier soll nun ein Einblick darüber gegeben werden, in welchen Zusammenhang Auswahlen zu Ebenen und Masken stehen und welche Anwendungsbereiche denkbar sind.

Ebenen entstehen auf vielerlei Art. Beispielsweise dann, wenn ein Bildelement kopiert und wieder eingesetzt oder ein Text mit dem Textwerkzeug geschrieben wird. Weiter gibt es noch spezielle Korrekturebenen. Versehen mit einer Maske ist es hier möglich, Bildbereiche einzeln zu bearbeiten.

Die Hintergrundebene

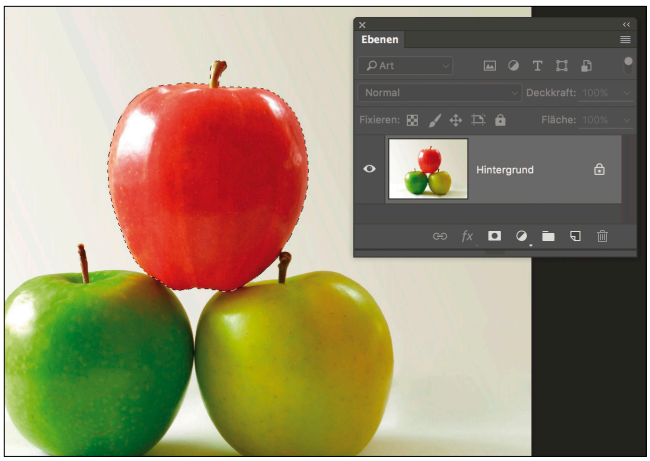
Bevor eine neue Ebene entsteht, wird häufig zunächst eine Auswahl getroffen.

Ist ein Motiv gewählt und soll es nun von seinem Hintergrund befreit werden, gibt es mehrere Möglichkeiten. Zum einen kann das Objekt lediglich auf eine einfarbige Fläche gesetzt werden – in diesem Fall können alle Elemente des Bildes, also Motiv wie Hintergrund – auf der ursprünglichen Ebene liegen bleiben. Hat ein Bild nur eine Ebene, handelt es sich in der Regel um die Ebene Hintergrund, wie sie Adobe Photoshop CC 2015 nennt. Per *Bearbeiten, Fläche füllen* werden die Flächen, die nicht in der Auswahl enthalten sind, mit der entsprechenden Farbe eingefärbt (Bild 1, Bild 2).

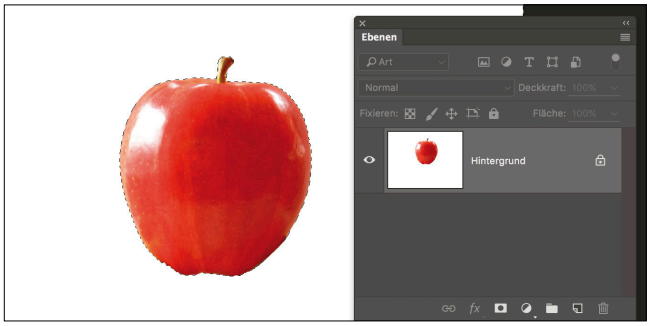
Besteht das Bild aus der Ebene Hintergrund und wird die gewählte Fläche gelöscht, erscheint der Dialog *Fläche füllen*. Hier kann neben der Vorder- und Hintergrundfarbe eine neue Farbe oder ein Muster gewählt werden. Die Option *Inhaltsbasiert* bewirkt, dass umliegende Bereiche gesucht und in die Auswahl kopiert werden. Auch in diesem Fall bleibt allein die Hintergrundebene bestehen. Eine Hintergrundebene ist also fixiert, was auch das kleine Schlosssymbol am rechten Ende der Ebene auf dem Ebenenbedienfeld zeigt.

Es zeigt sich also: Aus einer fixierten Ebene können keine Bildelemente gelöscht werden, um etwa nur das Motiv zu behalten. Transparenz ist somit bei dieser Vorgehensweise nicht möglich.

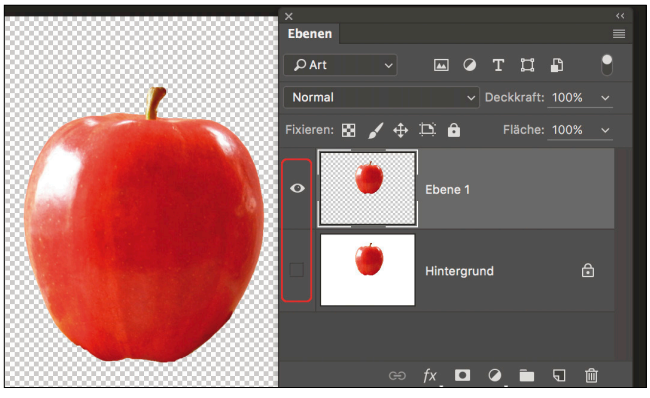
Wird das gewählte Objekt hingegen kopiert und in dasselbe oder auch ein anderes Bild eingefügt, legt Photoshop für das Duplikat eine neue Ebene an. Im Beispiel (Bild 3) wurde also der zuvor gewählte Apfel durch den Kopiervorgang auf eine weitere Ebene gelegt, alle Bereiche um den Apfel sind transparent. Photoshop zeigt transparente Bereiche als schachbrettartiges Muster, das jedoch erst zu sehen ist, wenn



Das Bild hat lediglich eine Ebene, die Hintergrundebene (Bild 1)



Hier wurden alle Bilddetails um den roten Apfel weiß gefärbt – das Bild besteht trotzdem noch aus nur einer Ebene (Bild 2)

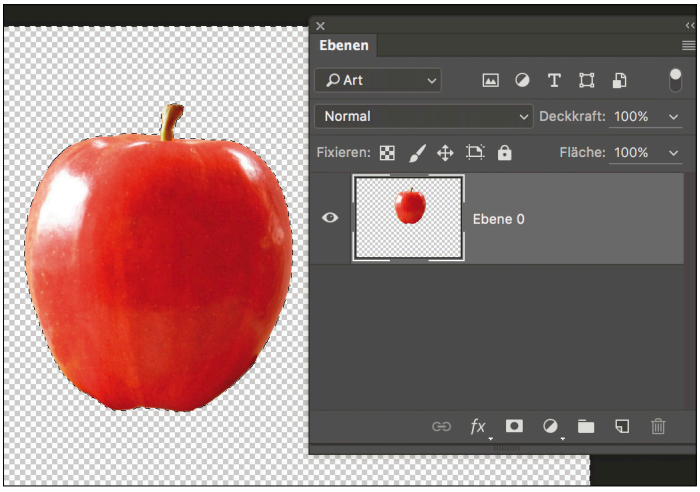


Hier ist die Sichtbarkeit der Hintergrundebene ausgeschaltet; zu sehen ist der Apfel der darüberliegenden Ebene mit transparentem Hintergrund (Bild 3)

die Hintergrundebene – die ja die komplette Bildfläche mit Bildpunkten füllt – ausgeblendet ist. Eine Ebene kann per Klick auf das Auge-Symbol links ein- und ausgeblendet werden.

Soll jedoch kein Duplikat angelegt, sondern nur der Hintergrund gelöscht werden, muss zuvor die Fixierung der Hintergrundebene aufgehoben werden. Dies gelingt durch einen Klick auf das Schlosssymbol im Ebenenbedienfeld. Photoshop wandelt dann die Hintergrundebene in die Ebene *0* um.

Nun kann eine Auswahl an Bildstellen über die Entfernen-Taste oder über den Befehl *Bearbeiten, Löschen* aus der Ebene entfernt werden – es erscheint das Schachbrettmuster, das Zeichen für Transparenz, obwohl das Bild aus nur einer Ebene besteht (Bild 4).



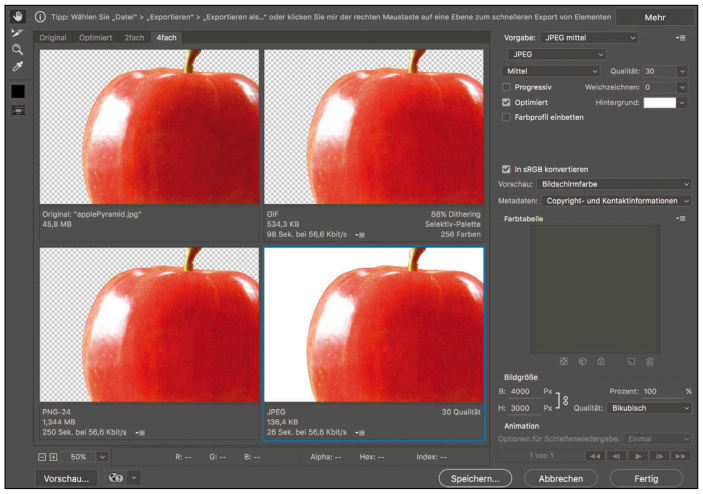
Das Bild besteht nun aus einer Ebene, die nicht fixiert ist und somit transparente Bereiche enthalten kann (Bild 4)

Dialog für Web speichern

Ob nun der Hintergrund in einer Farbe angelegt oder gelöscht wird, kann beim Einsatz des Bildes eine wichtige Rolle spielen. So kann etwa ein Objekt mit transparentem Hintergrund auf einer Website eine durchgängige Fläche belegen, die eine bestimmte Farbe oder auch ein Muster zeigt.

Dazu muss das Bild im entsprechenden Format vorliegen, das Transparenzen kennt. Infrage kommen etwa GIF oder PNG, da diese Formate, anders als JPEG, transparente Bereiche ermöglichen. Ein Bild im GIF-Format zeigt lediglich 256 Farben und eignet sich so nur für Grafiken, die mit wenigen Farben und Verläufen auskommen, etwa für ein Logo. Auch PNG-8 arbeitet nur mit der reduzierten Farbpalette. Für den Apfel im Beispiel sind also beide Formate ungeeignet.

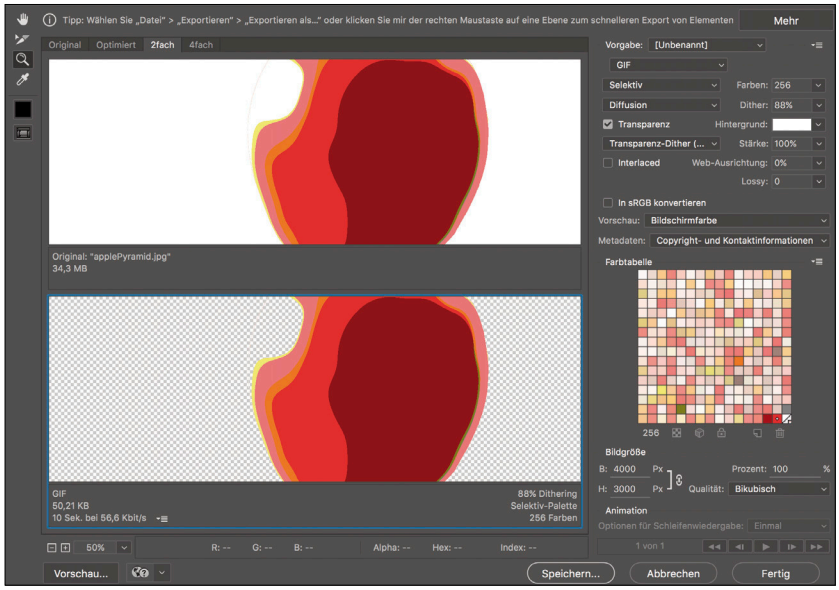
JPEG zeigt zwar alle Farben (256 pro Kanal), kann jedoch keine transparenten Bereiche darstellen. Hier punktet PNG-24, da dieses Format bei vollem Farbumfang auch mit Transparenzen umgehen kann.



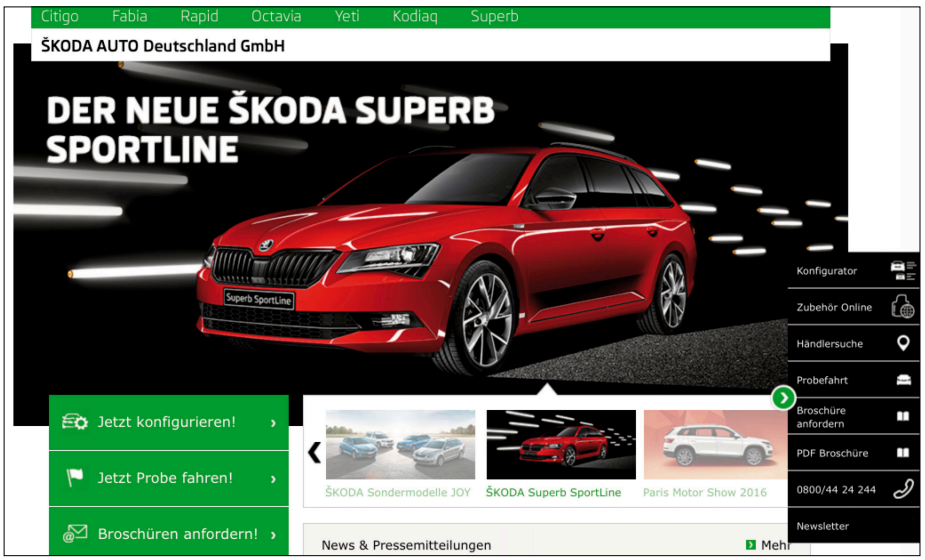
Der Dialog Für Web speichern liefert einen guten Überblick (Bild 5)

Für das Speichern von Bildern für das Web stellt Photoshop über *Datei, Exportieren, Für Web speichern* einen separaten Dialog bereit. Hier zeigen zwei bis vier Vorschaubereiche den in der Größe einstellbaren Bildausschnitt in den für digitale Anwendungen geeigneten, frei wählbaren Formaten. Unter den Vorschaubildern ist außerdem die jeweilige Dateigröße abzulesen, die das Bild bei den gewählten Einstellungen einnimmt (Bild 5).

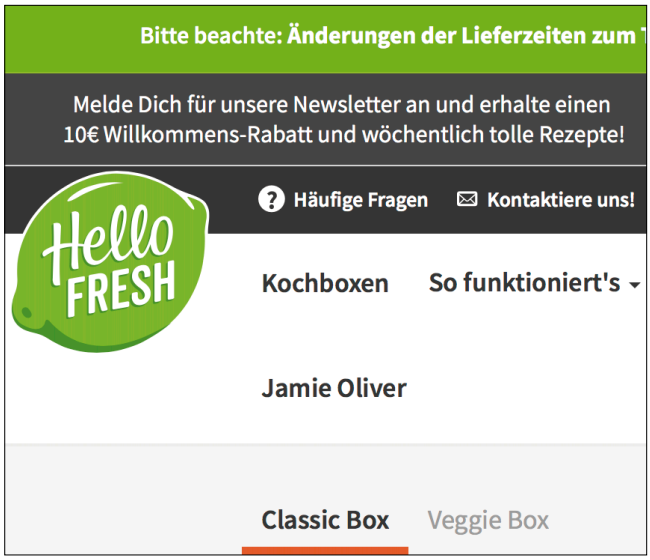
Zudem liefert der Dialog weitere Optionen am rechten Fensterrand. Soll etwa ein Bild als GIF oder PNG-8 mit transparentem Hintergrund gespeichert werden, muss zuvor keine Auswahl getroffen werden. Hier kann in der Farbtabelle rechts einer oder mehreren Farben Transparenz zugeordnet werden. Voraussetzung ist natürlich ein einfarbiger Hintergrund. Da nur Grafiken ▶



Grafische Elemente auf einfarbigem Hintergrund können ohne Auswahl über den Dialog *Für Web speichern* freigestellt werden (Bild 6)



Ist das Auto erst einmal freigestellt, kann es schnell auf einen anderen Hintergrund montiert werden (Bild 7)



Das Logo besteht aus homogenen Farbflächen und kann als PNG oder GIF auch auf einem mehrfarbigen Hintergrund platziert werden (Bild 8)

Das Textlogo wurde in Pfade umgewandelt und in einem Grafikprogramm als SVG gespeichert (Bild 9)

mit wenigen Farben und ohne Verläufe für diese Technik infrage kommen, wurde für das Beispiel (Bild 6) der Apfel grafisch aufbereitet, und er besteht dort nur aus wenigen, homogenen Farbflächen.

Anwendungen

Bilder mit transparentem Hintergrund sind in allen Medien zu finden. So kommen sie bei Printproduktionen als Freisteller zum Einsatz, etwa um Fließtexte um das Bildmotiv laufen zu lassen.

Im Web oder bei anderen mobilen Anwendungen sind Motive mit transparentem Hintergrund beispielsweise vor einem anderen Bild zu finden. So kann etwa ein Motiv im Vordergrund, im Beispiel der rote Skoda, blitzschnell in einer anderen Umgebung gezeigt werden (Bild 7).

In der Regel werden solche Montagen jedoch bereits im Bildbearbeitungsprogramm zusammengestellt, da hier ein Format mit vollem Farbumfang gewählt werden kann und somit eine bessere Bildqualität gewährleistet ist. Auch lassen sich so dem Hintergrund und dem Objekt angepasste Schatten realistisch in die Szene einbauen.

Andere Möglichkeiten bieten Grafiken aus einfarbigen Farbflächen. So etwa das Logo auf der Seite Hellofresh.de (Bild 8). Mit nur drei Farben, zwei Grüntönen und Weiß, klaren Kanten und ohne Verläufe kann es als PNG oder GIF mit Transparenz gespeichert werden, weist eine kleine Dateigröße auf und kann so auch außerhalb des Bildbearbeitungsprogramms auf einem beliebigen Hintergrund platziert werden.

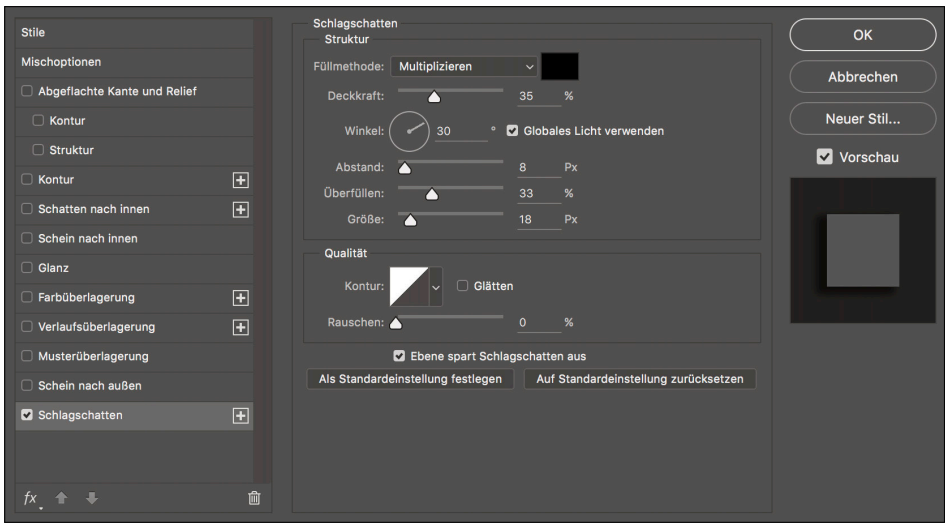
Text auf einer extra Ebene

Wie bereits zu Beginn erwähnt, liegt auch Text auf einer extra Ebene. Hierzu reicht die Arbeit mit dem Textwerkzeug – es sei denn, es findet ein Textmaskierungswerkzeug Verwendung.

Generell haben Texte in einem Bildbearbeitungsprogramm nichts verloren. Hierzu gibt es spezielle Layout- oder Grafiksoftware, etwa InDesign oder Illustrator. Einen Text in Photoshop anzulegen ist dann sinnvoll, wenn er etwa künstlerisch verfremdet werden soll.

Bei digitalen Anwendungen sollte Text aus mehreren Gründen direkt im Editor verfasst werden. Anders ist es, wenn eine bestimmte Schrift übernommen werden muss, etwa bei einem Schriftlogo, bei dem es gilt, das Corporate Design einzuhalten. Beispiel Mein-schoener-Garten.de: Hier wurde das Logo mit transparentem Hintergrund als SVG-Grafik abgespeichert. Die Textinformationen sind nicht mehr erhalten, da die Buchstaben in Pfade umgewandelt wurden – eine korrekte Darstellung des Textes ist so gewährleistet (Bild 9).

Eine solche Arbeit ist in Photoshop nur bedingt möglich. Zwar kann Text in eine Form umgewandelt werden, die dann ebenfalls durch Pfade definiert ist. SVG steht jedoch nicht als



Ebenenstile dienen dazu, ein Element vom Hintergrund abzuheben (Bild 10)

Format zum Speichern zur Verfügung; hier ist Illustrator das Programm der Wahl.

Bearbeitungsmöglichkeiten einer Ebene

Die Vorteile bei der Arbeit mit Ebenen liegen nicht nur in der Transparenz. Elemente, die auf einer extra Ebene liegen, lassen sich in unterschiedlicher Art und Weise bearbeiten. Werden sie an einen anderen Platz geschoben, sind wieder die auf den Ebenen darunter liegenden Bildbereiche zu sehen. Zudem können sie skaliert, gedreht oder in anderer Weise transformiert werden.

Um ein Objekt, das auf einer eigenen Ebene liegt, vom Hintergrund besser abzuheben, kann es mit einem Ebenenstil belegt werden. Hier sind neben dem immer noch sehr häufig verwendeten Schlagschatten auch andere Stile wie Kontur, Glanz oder Farbüberlagerung zu finden (Bild 10). Einen solchen Effekt zeigt beispielsweise die App der Tageszeitung BILD: In einem Beitrag über den Rennfahrer Sebastian Vettel wurde dieser freigestellt, mit einem weißen Schein nach außen versehen und über ein anderes Hintergrundbild gelegt (Bild 11).

Einstellungsebenen

Für die Justierung von Farbe, Helligkeit und Kontrast bietet Photoshop eigene Einstellungs- oder Korrekturebenen. Diese sind einerseits im Ebenenmenü zu finden, zum anderen bietet das Bildbearbeitungsprogramm dafür das separate Bedienfeld *Korrekturen*. Zu finden sind hier alle Bearbeitungsoptionen, die auch im Menü *Bild* unter *Korrekturen* zu erreichen sind – mit einem großen Vorteil: Einstellungs- oder Korrekturebenen legen ihre Einstellungen über die Ebenen, die mit Bildpunkten belegt sind, und ändern diese nicht.

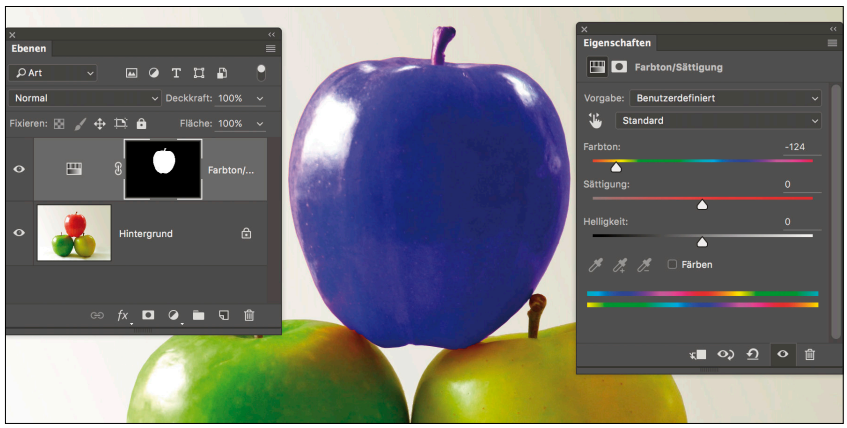
Somit bleiben die Originaldaten erhalten. Außerdem lassen sich so über die Einstellungs- oder Korrekturebene auch nachträglich noch Änderungen vornehmen.

Die Maske der Einstellungsebene

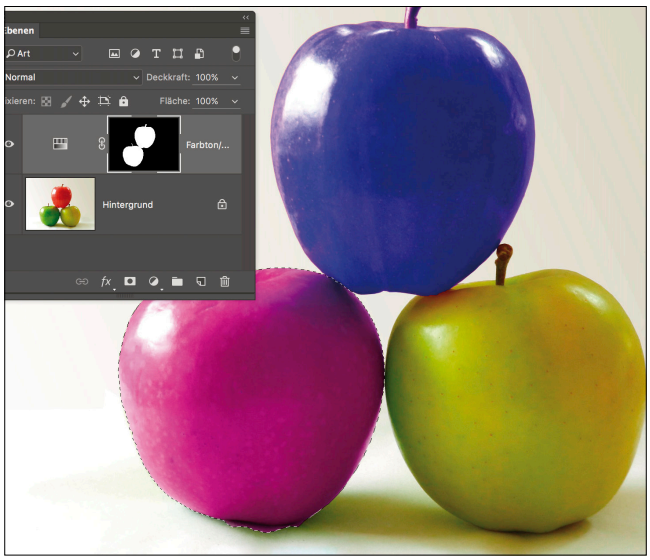
Sobald eine Einstellungsebene angewählt wurde, legt Photoshop diese über die aktivierte Ebene. Zwei Miniaturen sind nun zu sehen: die Ebenenminiatur, die das Symbol der gewählten Einstellung zeigt, und rechts daneben eine Maskenminiatur. Ist eine Auswahl aktiv, wird lediglich der gewählte Bereich mit den Einstellungen belegt; der Rest ist maskiert und somit von der Änderung nicht betroffen.

Im Beispiel (Bild 12) wurde zunächst der rote Apfel gewählt. Über die Korrekturebene *Farbton/Sättigung* wech- ▶

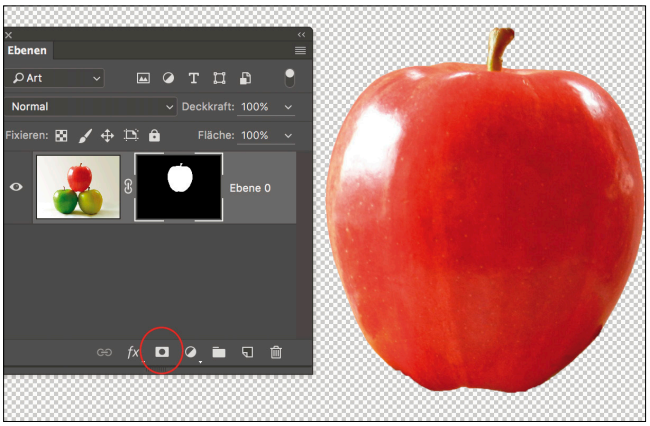
Vettel mit Glow: Die BILD-App verwendet für Bildmontagen Ebenenstile (Bild 11)



Einstellungsebene mit Ebenenmaske: Lediglich der gewählte Bildteil übernimmt die Änderungen (Bild 12)



Werden weitere Bereiche in der Ebenenmaske weiß gefärbt, wirken sich die Einstellungen auch auf diese Bildpunkte aus (Bild 13)



Per Klick auf die Schaltfläche *Ebenenmaske* schaltet Photoshop die nicht gewählten Bildbereiche unsichtbar (Bild 14)

selt der Apfel nun seine Farbe – die beiden unteren Äpfel sowie der Hintergrund sind von der Änderung nicht betroffen. Dieser Umstand wird auch in der Maskenminiatur sichtbar: Lediglich die Fläche des ehemals roten Apfels ist hier weiß, der Rest ist schwarz. Weiß zeigt die nicht maskierten Flächen und somit die Korrekturen, Schwarz kennzeichnet dagegen maskierte Bereiche.

Einstellungen übernehmen

Soll nun auch der grüne Apfel unten links die Einstellungen übernehmen, kann diese Fläche demaskiert werden. Hierzu kann einfach mit weißer Farbe in die Ebenenmaske gemalt werden; besser und exakter geht es jedoch über eine Auswahl: Zunächst wird, am einfachsten mit dem Schnellauswahlwerkzeug, der zweite Apfel gewählt. Dann wird bei aktivierter Ebenenmaskenminiatur (ein weißer Rand erscheint darum) die gewählte Fläche in der Maske weiß gefüllt.

Kleine Info am Rande: Dass der Apfel nicht blau, sondern magenta wird, liegt an der Art der Korrektur: Ändert man den Farbton, übernehmen die Bildpunkte die jeweilige Komplementärfarbe (Bild 13).

Auch ohne eine Korrekturebene sind Ebenenmasken leicht zu erstellen – und dabei gut zu gebrauchen. Am schnellsten geht das über das Ebenenbedienfeld: Per Klick auf die Schaltfläche *Maske hinzufügen* wird – ohne Auswahl – über die gesamte Fläche eine Maske gelegt.

Diese ist jedoch weiß, es ist also die gesamte Bildfläche sichtbar. Um Bereiche der Ebene tatsächlich zu maskieren, müssen diese einfach mit schwarzer Farbe in der Maske gefüllt werden. Sind vor der Maskierung bereits Bildflächen gewählt, ist nur die Auswahl sichtbar, den Rest maskiert Photoshop automatisch. Das funktioniert sogar auf einer Hintergrundebene, da Photoshop diese beim Anlegen der Maske automatisch in eine Ebene *0* umwandelt (Bild 14).

Vorteil einer Maske

Der Vorteil einer solchen Ebenenmaske liegt darin, dass sie wesentlich flexibler ist als andere Freisteller, bei denen die unerwünschten Bildbereiche einfach gelöscht wurden. So werden einfach per Pinselstrich in der Maske weitere Stellen im Bild sichtbar oder unsichtbar.

Sollen dann die verborgenen Bildstellen tatsächlich gelöscht werden, wird einfach die Maske auf den kleinen Mülleimer, links unten im Ebenenbedienfeld, gezogen. Die Maske verschwindet nun, mit ihr allerdings auch alle unsichtbaren Pixel.

Fazit

Wer nur kleinere Arbeiten an seinen Bildern durchführen möchte, etwa nur eine Größenänderung oder eine schnelle Korrektur, die die komplette Fläche betrifft, kommt auch ohne Ebenen und Masken aus.

Sobald allerdings komplexere Änderungen geplant sind, geht es nicht mehr ohne Ebenen. Das betrifft Korrekturen an einzelnen Bildstellen genauso wie aufwendige Montagen, die sich aus mehreren Bildelementen zusammensetzen. Zudem erlauben es Masken und Einstellungsebenen, die Bilder nichtdestruktiv zu korrigieren und so das Ausgangsmaterial zu erhalten. Doch sind damit die Möglichkeiten in Photoshop noch längst nicht ausgeschöpft: Neben Ebenen und Masken gibt es zahlreiche weitere Optionen, die eine Auswahl ganz nach dem erforderlichen Anwendungszweck ermöglichen. Eine weitere, dritte Folge zu diesem Thema in der nächsten Heftausgabe beleuchtet beispielsweise den Einsatz von Kanälen und Pfaden. ■

Katharina Sckommodau
arbeitet als Autorin, Grafikerin und Dozentin, unter anderem für die Akademie der Bayerischen Presse und für Macromedia. Sie veröffentlicht regelmäßig Beiträge in Fachzeitschriften und verwirklichte mehrere Buchprojekte.

developer media

developer media Trainings

Updates für Ihr Know-How

„Auch Entwickler sollten hin und wieder das Kleingedruckte lesen."

Antje Kilián
Spezialistin für Vertrags-, Urheber-, Lizenz- und Computerstrafrecht

UX und UI-Design für Entwickler

Trainerin: Peggy Reuter-Heinrich

2 Tage, 01.-02.12.2016, Köln
Ab 1.799,- EUR zzgl. MwSt.

Softwarequalität – Eine Einführung

Trainer: David Tielke

1 Tag, 08.12.2016, Köln
Ab 599,- EUR zzgl. MwSt.

IT-Recht für Entwickler

Trainerin: Antje Kilián

2 Tage, 12.-13.12.2016, Köln
Ab 1.799,- EUR zzgl. MwSt.

Server-side SharePoint Development

Trainer: Martin Groblschegg

3 Tage, 29.11.-01.12.2016, 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

GAMIFICATION ALS MOTIVATOR IN SPIELFREMDEN BEREICHEN

Lasst uns spielen

Gamification will die Motivatoren eines Spiels auf spielfremde Bereiche übertragen.

Das Ziel: Nutzer sollen zu unliebsamen oder nervigen Aufgaben motiviert werden. Auch im Bereich mobiler Anwendungen hat dieser Ansatz Einzug gehalten.

Bevor wir über die Übertragung von Elementen eines Spiels auf spielfremde Bereiche diskutieren und uns Konzepte und bestehende Anwendungen ansehen, wollen wir versuchen zu klären, was man eigentlich unter einem Spiel versteht.

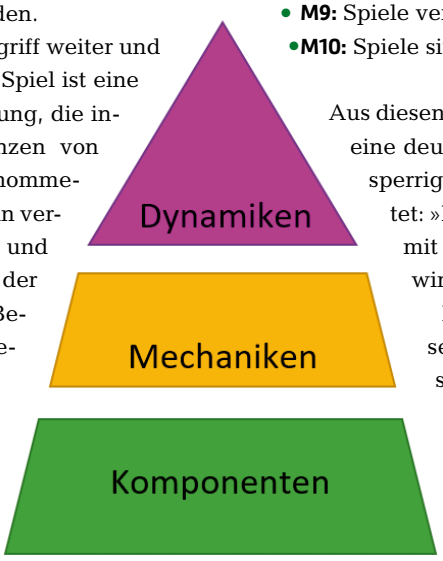
Anders könnte man auch fragen, welche Eigenschaften kennzeichnen ein Spiel? Was führt dazu, dass wir von interessanten Spielen nicht lassen können, wir sie immer wieder spielen und dabei auch bei jeder neuen Durchführung erneut Spaß haben? Dabei macht es zunächst keinen Unterschied, ob wir Computerspiele, Gesellschaftsspiele oder jede andere Form von Spielen betrachten.

Was ist ein Spiel?

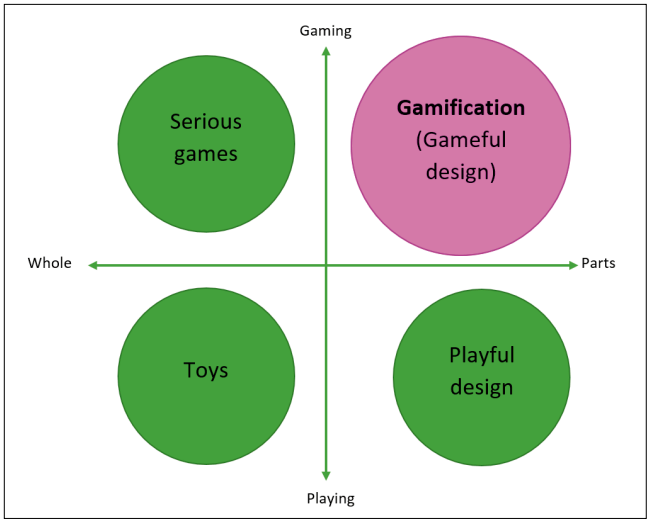
Sehr oft versucht man, einerseits Spiel und anderseits Ernst als Kontraste gegenüberzustellen. Diese Abgrenzung ist jedoch nur in Einzelfällen zur Unterscheidung geeignet. Betrachten Sie beispielsweise ein Kind: Es nimmt sein Spiel in der Regel durchaus sehr ernst. Verlieren im Spiel kann zu tiefer Traurigkeit und echter Enttäuschung führen.

Ein anderer Ansatz stellt die Kontraste Spiel und Arbeit gegenüber. Auch diese Unterscheidung greift mit einem Blick auf die Lebenswirklichkeit meist ebenso zu kurz. Viele Sportspiele werden in extrem professioneller Art und Weise ausgeführt, beispielsweise Profi-Fußball. Das notwendige Training, die Ausbildung und letztendlich die Verdienstmöglichkeiten zeigen, dass manche Spiele durchaus den Charakter von Arbeit haben und eben auch mit der zuvor diskutierten Ernsthaftigkeit durchgeführt werden.

Daraus folgt, dass man den Spielbegriff weiter und umfassender definieren muss: »(Das) Spiel ist eine freiwillige Handlung oder Beschäftigung, die innerhalb gewisser festgesetzter Grenzen von Zeit und Raum nach freiwillig angenommenen, aber unbedingt bindenden Regeln verrichtet wird, ihr Ziel in sich selbst hat und begleitet wird von einem Gefühl der Spannung und Freude und einem Bewusstsein des Andersseins als das gewöhnliche Leben.«



Spielelemente: Gamification verwendet viele Elemente des Game-Designs (Bild 2)



Die Veranschaulichung von Gamification (Bild 1)

Diese Definition zeigt, dass man einem Spiel bestimmte Schlüsselmerkmale zuordnen kann. Die wichtigsten Merkmale (M) wollen wir hier benennen (Quelle: Schell, J.: Die Kunst des Game Design):

- **M1:** Spiele werden willentlich gespielt (freiwillig).
- **M2:** Spiele haben eine Zielsetzung.
- **M3:** Spiele beinhalten einen Konflikt.
- **M4:** Spiele haben Regeln.
- **M5:** Spiele können gewonnen oder verloren werden.
- **M6:** Spiele sind interaktiv.
- **M7:** Spiele stellen Spieler vor eine Herausforderung.
- **M8:** Spiele können eine eigene Bedeutsamkeit generieren.
- **M9:** Spiele verwickeln die Spieler in das Geschehen.
- **M10:** Spiele sind geschlossene, formale Systeme.

Aus diesen Merkmalen heraus hat der Autor des Buches eine deutlich kompaktere, einprägsame und weniger sperrige Definition für ein Spiel gegeben. Diese lautet: »Ein Spiel ist eine Problemlösungsaktivität, die mit einer spielerischen Einstellung angegangen wird.«

Ein kleines Zwischenfazit können wir an dieser Stelle bereits ziehen: Spiele eignen sich sehr gut dazu, Menschen zu einem bestimmten Verhalten zu motivieren. Spaß und Erfahrungen, die man durch das Spiel macht, sind wichtige Faktoren, die uns dazu bringen, zu spielen. Die Bewältigung von Hindernissen, das Verfolgen eines Ziels, die Suche

nach etwas Neuem und Unbekanntem und schließlich eine Interaktion mit anderen Spielern gehören ebenso zum Spaß am Spielen.

Möchte man die Erkenntnisse und Wirkungsweisen eines Spiels auf andere Bereiche übertragen, so spricht man von Gamification. Der Artikel stellt die Hintergründe dieses interessanten Ansatzes vor und zeigt die praktische Anwendung in Software, insbesondere in Form von Apps für Smartphones und Tabletts.

Gamification

Der Begriff Gamification bezieht sich ursprünglich auf das englische Wort Game (Spiel) und besagt gemäß der oben genannten Definition, dass ein Problem mit Hilfe eines spielerischen Ansatzes gelöst werden soll. Abzugrenzen sind die Begriffe Game und Play (Bild 1).

Play kann dagegen als freies Bewegen innerhalb fester Grenzen definiert werden. Ernsthafte Spiele (Serious Games) sind vollwertige Spiele, bei denen Wissen vermittelt wird. Auch ist Gamification von Simulationen oder Planspielen zu unterscheiden.

Dieses sind bekannte Situationen, die auf wesentliche Elemente und Grundregeln reduziert werden. Das Ziel ist, eine neue und innovative Lösung zu erproben beziehungsweise zu ermitteln.

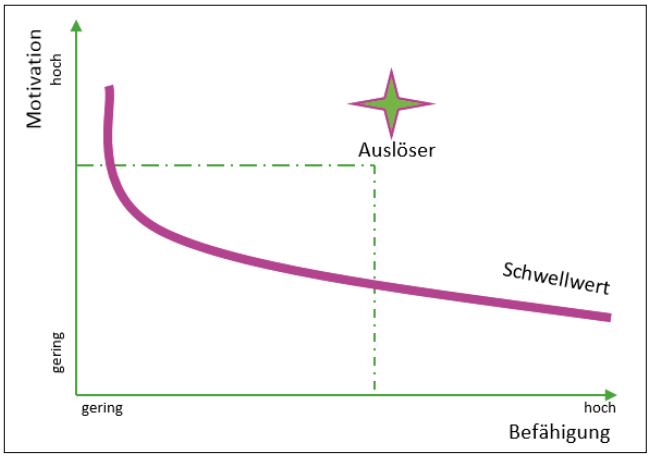
Spieltypische Elemente

Spiele enthalten bestimmte Elemente, damit sie funktionieren und die genannten Merkmale erfüllen. Gamification verwendet viele dieser Elemente des Game-Designs.

Grundsätzlich können alle Spielelemente in drei Hauptkategorien aufgeteilt werden: Dynamiken, Mechaniken und Komponenten. Diese sind so organisiert, dass jede Mechanik mit mehreren Dynamiken und jede Komponente mit einer

Tabelle 1: Typische Spielkomponenten

Komponente	Beschreibung
Ranglisten	Erlauben Vergleiche zwischen den erreichten Leistungen der Spieler. Beispielsweise kann anhand der erreichten Punkte oder des Spielfortschritts ein Wettkampf zwischen den Spielern simuliert werden.
Belohnungen	Diese sind eine Art der Anerkennung für eine erbrachte Leistung: zum Beispiel in Form von Badges, Auszeichnungen, Titeln oder einem erhöhten Punktestand.
Status	Eine für andere Mitspieler sichtbare Anzeige des aktuellen Spielfortschritts. Auf diese Weise werden der erreichte Punktestand, das aktuelle Level, errungene Auszeichnungen, der Grad der Erfüllung der Mission und der Leistungsfortschritt dargestellt.
Feedback	Der Spieler erhält direktes Feedback auf seine Handlungen. Bei positiven Rückmeldungen wirkt dieses als Motivator.
Quests	Ein Rätsel oder fortlaufende Aufgaben, die innerhalb einer bestimmten Zeit gelöst werden müssen.



Das Verhaltensmodell von Fogg (Bild 3)

oder mehreren Mechaniken beziehungsweise Dynamiken verknüpft ist (Bild 2). Was versteht man unter diesen Elementen?

- **Dynamiken:** Es handelt sich um Elemente wie zum Beispiel die Geschichte des Spiels, die Beziehungen der Spieler untereinander oder die Emotionen bei den Spielern. Sie sind Elemente, die man beachten muss, aber nicht direkt in ein Spiel integrieren kann. Verschiedene Spielertypen brauchen unterschiedliche Dynamiken, damit ihre Motivation und ihr Engagement langfristig sichergestellt sind. Zum Beispiel verlangen Spieler vom Typ Killer nach schnelleren Dynamiken als Spieler vom Typ Socialiser.
- **Mechaniken:** Diese sind Herausforderungen, Wettbewerb, Belohnung und Bewertung. Damit versucht man, das Engagement bei den Spielern zu wecken, um die beschriebenen Dynamiken zu erreichen und umzusetzen.
- **Komponenten:** Es sind die spezifischsten Formen, die Mechaniken und Dynamiken annehmen können. Typische Komponenten sind in Tabelle 1 wiedergegeben.

Was bringt Menschen zu einem bestimmten Verhalten? Wie kann es durch Spielelemente beeinflusst werden? Die Erklärung gelingt am besten anhand des Verhaltensmodells von Fogg. Drei grundlegendende Faktoren müssen zusammenkommen:

- **Motivation:** Die Person muss einen Grund haben, etwas zu tun. Die Quellen der Motivation sind: Empfindungen, Erwartungen und soziale Beziehungen.
- **Befähigung:** Die Person muss sich in der Lage fühlen, etwas zu tun. Je einfacher die Aufgabe ist, desto schneller breitet sich das Gefühl aus, in der Lage zu sein, aktiv zu handeln.
- **Auslöser:** Es muss ein Impuls vorhanden sein, welcher der Person das Gefühl von Motivation und Befähigung vermittelt. Bei genügend Motivation und Befähigung löst dieser sofort ein bestimmtes Verhalten aus.

Bild 3 erklärt das Modell grafisch: Die x-Achse verdeutlicht die Befähigung, etwas zu tun, die y-Achse verdeutlicht die Motivation. Die Kurve beschreibt den Schwellenwert, ab dem es durch einen Auslöser zu einer Verhaltensänderung ▶

kommt. Fehlt entweder der Faktor Motivation oder der Faktor Befähigung, beziehungsweise liegt die Kombination der beiden Faktoren unterhalb des Schwellenwerts, so kommt es zu keiner Verhaltensanpassung.

Übertragen wir dieses auf einen konkreten Fall: Nehmen wir an, wir möchten eine Person mit Hilfe eines Bewegungskurses zu einer gesünderen Lebensweise bringen. Die Person muss einerseits motiviert sein, überhaupt über ihre Lebensweise nachzudenken, und in gewisser Weise auch zu dem Schluss kommen, dass eine Verbesserung notwendig ist (Motivation).

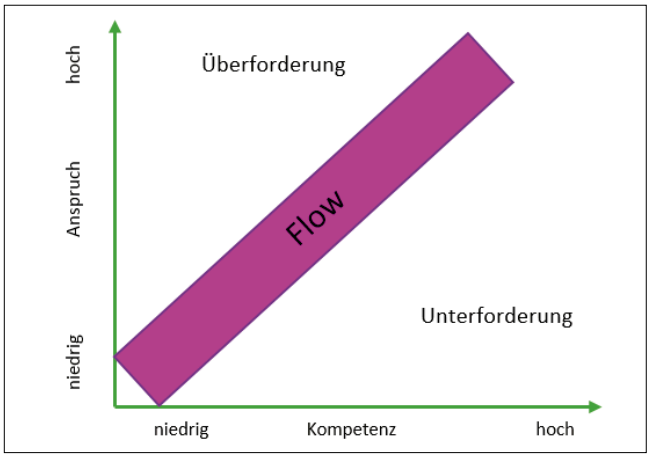
Ebenso muss die Person dazu in der Lage sein, auch am Bewegungskurs teilzunehmen. Es dürfen also keine persönlichen oder externen Hinderungsgründe dagegen sprechen (Befähigung). Trotz grundsätzlicher Motivation und Befähigung bedarf es in der Regel noch eines konkreten auslösenden Moments, um mit dem Sport nun auch letztendlich zu beginnen. Beispielsweise man wird konkret von einer Person dazu aufgefordert.

Bei Gamifikation wird versucht, alle drei Faktoren auf Arbeitsaufgaben oder sonstige bisher nicht zufriedenstellende Aufgaben zu übertragen. Das Ziel: Die Aufgaben sollen zu spannenden und unterhaltsamen Erlebnissen werden.

In den Flow-Zustand versetzen

Die geforderte Leistung muss den Fähigkeiten entsprechen. Ein Spiel darf weder unter- noch überfordern. Im Fall der Unterforderung wird es sehr schnell langweilig, im Fall der Überforderung sieht man keine Chance, das gewünschte Ergebnis zu erreichen.

Entlang des genannten Flow-Zustands (Bild 4) entspricht das Leistungsniveau den Fähigkeiten. In diesem Zustand hat die Person keinen übermäßigen Zeitdruck, ist nicht abgelenkt und hangelt sich von Aufgabe zu Aufgabe. Daher wird in der Regel gut strukturiert und motiviert gearbeitet. Zu schwierige Aufgaben ruinieren diesen Zustand. Das Ziel in diesem Zusammenhang: In einer Gamification-Anwendungen soll der Flow-Zustand erreicht werden. Wie bereits erwähnt, versucht man durch Gamification, bestimmte Aktivitäten unterhaltsamer und spaßiger zu gestalten. Insbesondere ist das der Fall, wenn die Aktivitäten mit Arbeit verbunden sind. Die Motivation zur Arbeit an trockenen, wenig spannenden und aufregenden Aufgaben soll gesteigert werden.

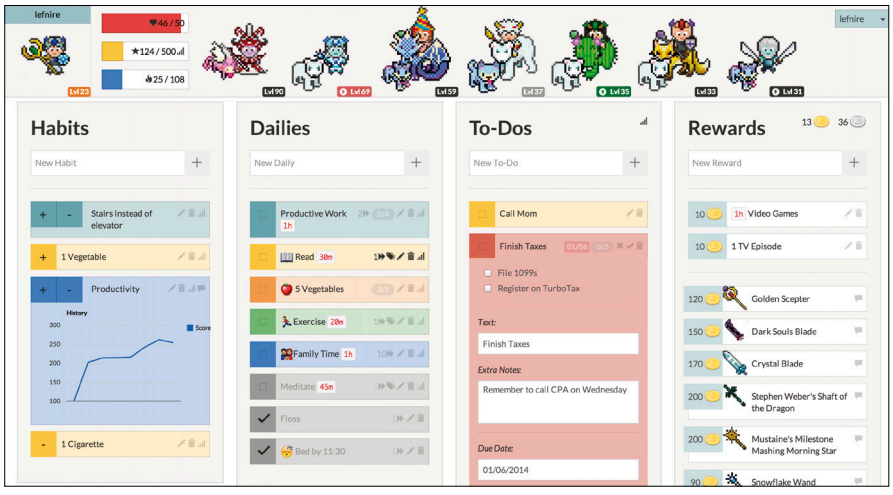


Flow-Konzept: In einer Gamification-Anwendung soll der Flow-Zustand erreicht werden (Bild 4)

Im positiven Fall führt es zu einer Verhaltensänderung, insbesondere dann, wenn ein positives Ergebnis aus den spielerischen Aktivitäten in den nichtspielerischen Kontext transferiert werden kann.

Grundsätzlich bezieht sich Gamification nicht nur auf digitale Anwendungen. Dennoch sind die meisten Einsatzgebiete mit Smartphones, Tablets oder Software allgemein verbunden. Die möglichen Einsatzgebiete sind im Kontext von sozialem Leben, Gesellschaft, Sport und gesunder Ernährung, Nachhaltigkeit et cetera zu finden. Einige konkrete Beispiele:

- **Free Rice:** Es handelt sich um eine gemeinnützige Webseite, deren Nutzer gleichzeitig Englisch-Vokabeln lernen und Reiskörner für Hungernde spenden können. Sie finanziert sich über Werbung und ihre Spenden werden vom World Food Programme der Vereinten Nationen an Bedürftige verteilt. Richtige Antworten werden mit virtuellen Reiskörner belohnt, welche später gegen echten Spenden eingetauscht werden können. Das Motto lautet: Play online, learn online and feed the hungry.
- **Mint.com:** Die Software bietet Hilfe bei der Verwaltung von Finanzen für Privatpersonen und verwandelt dieses in ein Spiel.
- **Weight Watchers:** Animiert mit Hilfe eines Punktesystems, auf eine gesunde Ernährung umzusteigen.



Die App Habitica verwendet Gamification (Bild 5)

Ebenso sind Gamification-Ansätze in der Arbeitswelt zu finden. Dabei sind sowohl unternehmensinterne als auch unternehmensexterne Anwendungen anzutreffen. Ein Beispiel für eine unternehmensexterne Anwendung: Die Bewerber des Marriot Hotels können

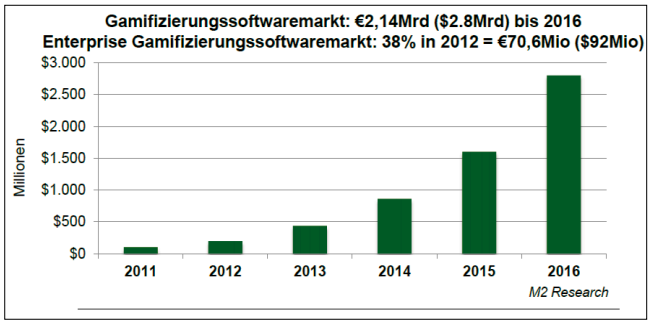
in einer Facebook-App spielerisch Arbeiten in der Hotelküche lösen.

Externe und interne Nutzung von Gamification unterscheiden sich oft in der Freiwilligkeit der Nutzung. Bei der unternehmensexternen Gamification entscheiden die Nutzer selbst, ob sie das Spiel beginnen oder nicht. Unternehmensinterne Spiele sind oft nichtfreiwilliger Natur.

Grundsätzlich sollte man Gamification-Ansätze – wenn möglich – optional zur Lösung bestimmter Arbeitsaufgaben anbieten. Es wird auch immer Mitarbeiter geben, die monotone oder an sich wenig motivierende Aufgaben ohne externe zusätzliche Anreize gut ausführen. Ein Zwang zu Teilnahme an Gamification wäre dann kontraproduktiv und würden den motivierenden Effekt zerstören.

Gamification in Apps

Blicken wir konkret auf einige Apps, die mit dem Hilfsmittel Gamification arbeiten. Ein gutes Beispiel ist die Fitness-App Zombies, Run! Der Anwender wird während des Joggens in eine Spielwelt geschickt, wo er Versorgungsgüter in einem Zombiegebiet verteilen soll. Die Erfassung des Standorts erfolgt per GPS-Signal. Kommt der Läufer den virtuellen Zombies zu nah, muss er sein Lauftempo steigern. Der Anwender soll so zu einer höheren sportlichen Leistung mit Hilfe spielerische Elemente motiviert werden. Das Spiel funktioniert ebenso beim Walken.



Die Entwicklung des Gamifizierungsmarkts (Bild 6)

Spielertypen

Grundsätzlich kann man zwischen folgenden Typen von Spielern unterscheiden:

- **Achiever:** Klar definierte Ziele sind eine gute Voraussetzung, um möglichst viel zu erreichen und zu sammeln, zum Beispiel Punkte, Level, Auszeichnungen et cetera.
- **Explorer:** Sie wollen möglichst viele unbekannte Gebiete und Funktionen untersuchen beziehungsweise entdecken.
- **Killer:** Dieser Spielertyp mag die Herausforderungen gegen andere Spieler. Leaderboards und Rankings lösen diese Mechanik aus.
- **Socialiser:** Dieser Spielertyp sucht Kontakt zu den anderen Mitspielern. Sie zeigen wenig Interesse für den Wettkampf, dafür sind sie am Austausch von Erlebnissen und Informationen mit anderen Spielern sehr interessiert.

Links zum Thema

- Spielwissenschaft
  https://de.wikipedia.org/wiki/Spielwissenschaft#Entstehungstheorien
- Ünsür, Z.: Gamification und BPMN
  ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart.../BCLR-0187.pdf
- Spiel-Elemente in der IT-Beratung
  https://www.maibornwolff.de/sites/default/files/news/files/whitepaper_spiel-elemente_in_der_it-beratung_maibornwolff.pdf
- Zeitmanagement-Anwendung Habitica
  https://habitica.com

Ebenso beliebt ist die Nike+-Running-App. Hier werden die Bewegungsdaten des Läufers statistisch aufbereitet. Diese kann man anhand einer Karte nachvollziehen und mit anderen Usern austauschen. Motivierend wirken also zwei Aspekte: zum einen, die Entwicklung der eigenen Laufleistung über einen längeren Zeitraum zu beobachten, und noch viel intensiver der Vergleich mit anderen Personen, welche die gleichen Ziele verfolgen.

Habitica verwandelt die eigene To-do-Liste in ein Rollenspiel. Mit Pixel-Schwertern und Zaubersprüchen kämpft man zusammen mit Freunden um die Perfektionierung des Alltags (Bild 5).

Fazit

Gamification ist längst kein Spiel mehr. Das Phänomen ist heutzutage zum Trend im Bereich Software- und App-Entwicklung geworden. Über die erreichte Motivation wird ganz klar Geld verdient. Gleichgültig welche Aussagekraft man Bild 6 zurechnet, mit der Entwicklung der entsprechenden Software werden erhebliche Umsätze generiert und gleichzeitig wird diese Software nicht zum Spaß produziert. In Untersuchungen konnte bereits nachgewiesen werden, dass das bespielte Business davon deutlich profitiert. Die Mitarbeiter und Anwender dürfte es in den meisten Fällen freuen, erledigen wir doch so manche ungeliebte Aufgabe vielleicht mit etwas mehr Freude. ■

Olena Bochkor
studierte Betriebswirtschaftslehre mit dem Schwerpunkt Wirtschaftsinformatik.
http://larinet.com

Mathias Rosenthal/Fotolia

PLATFORM AS A SERVICE (PAAS)

Applikationen in der Cloud entwickeln

Mit PaaS lassen sich Anwendungen schneller konzipieren und einfacher ausliefern.

Beim Betreibermodell Platform as a Service (PaaS) stellt ein Cloud-Provider eine Plattform mit Betriebssystem, Middleware und Entwicklungswerkzeugen zur Verfügung, auf der ein Kunde eigene Applikationen entwickeln und/oder betreiben kann – so die allgemein akzeptierte Definition des National Institute of Standards and Technology (NIST).

In diesem Artikel analysieren wir das Potenzial und die Grenzen dieser klassischen, auch als Application Platform as a Service (aPaaS) bezeichneten Angebote (Bild 1).

Die Vorteile von PaaS

Application Platforms as a Service entlasten Entwickler und Administratoren vor allem dadurch, dass sie ihnen viele infrastrukturnahe Routineaufgaben abnehmen. »Der Anwender muss sich nicht um die Server kümmern, das Betriebssystem warten oder bei steigender Last für mehr Rechenleistung sorgen, er kann sich ganz auf das Programmieren seiner Applikation konzentrieren«, sagt René Büst, Senior Analyst und Cloud Practice Lead bei Crisp Research. »Die Entwicklerproduktivität lässt sich deutlich erhöhen, wenn Umgebungen bei Bedarf automatisiert provisioniert werden können. Häufig kann dies sogar durch die Entwickler selbst veranlasst werden«, ergänzt Holger Sirtl, Cloud Solution Architect bei Microsoft Deutschland. Entwicklungs- und Betriebsprozesse ließen sich so sehr viel besser verzahnen.

Anwendungen schneller auf den Markt bringen

»Mit PaaS für die Anwendungsentwicklung und -implementierung können Unternehmen Anwendungen schneller auf den Markt bringen«, nennt Michael Ramsperger, Regional

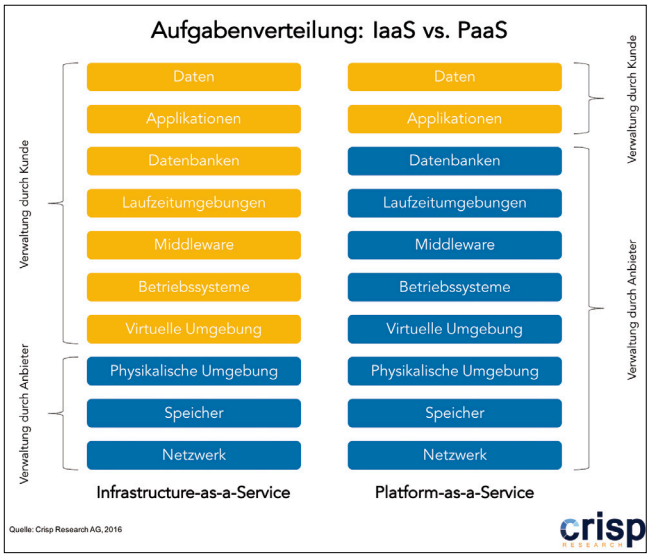
Director Central Europe bei Pivotal, einen weiteren Vorteil. »Entwickler haben die Möglichkeit, kleinere Einheiten zu kreieren, die einfach zu installieren sind.«

Neben der reinen Entwicklung sieht Constantin Gonzalez, Principal Solutions Architect bei der Amazon Web Services Germany GmbH, auch Vorteile im Betrieb, etwa beim Monitoring oder der automatischen Skalierung von Apps. »Generell bilden PaaS beziehungsweise PaaS-APIs oft den Grundstein für mehr Transparenz und Standardisierung des Entwicklungs- und Betriebsmodells, was in der Folge auch zur Optimierung von Prozessen und Kosteneinsparungen führen kann«, sagt er.

Auch wenn der PaaS-Markt im Vergleich zu Infrastructure as a Service (IaaS) und vor allem zu Software as a Service (SaaS) klein ist, erkennen immer mehr Firmen die Vorteile von PaaS.

Laut Gartner wird die weltweite Nachfrage nach PaaS allein aus der Public Cloud 2016 im Jahresvergleich um 21 Prozent auf 4,6 Milliarden Dollar wachsen.

Und die Experton Group nennt im »Cloud Vendor Benchmark 2015« ein Marktvolumen für PaaS im deutschen B2B-Cloud-Service-Markt von über 250 Millionen Euro, wobei 80 bis 90 Prozent auf aPaaS entfallen.



IaaS versus PaaS: Platform as a Service (PaaS) nimmt dem Kunden viele Aufgaben ab, die er bei Infrastructure as a Service (IaaS) selbst erledigen müsste (Bild 1)

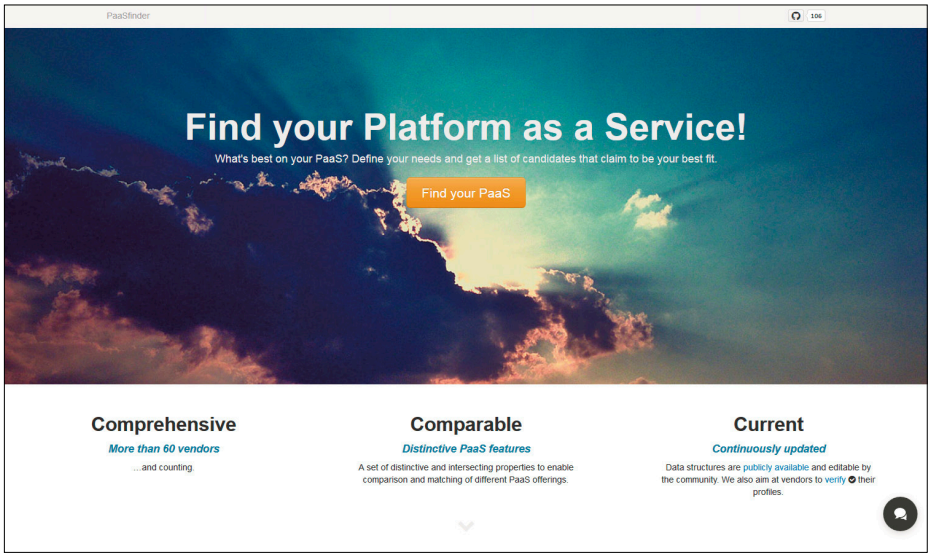
Voraussetzungen für die Nutzung

Wer PaaS optimal nutzen möchte, muss seine Applikationen fit machen für die Cloud, was nicht immer gelingt. »Gerade bei bestehender, proprietärer Standardsoftware oder sogenannten Packaged Solutions sind die notwendigen Anpassungen (…) zum Teil gar nicht möglich beziehungsweise nur mit hohem Aufwand zu realisieren«, sagt Amazon-Mitarbeiter Gonzalez. René Büst von Crisp Research geht sogar noch einen Schritt weiter: »Platform as a Service ist nur für cloudnative Applikationen geeignet.«

»Wir sehen aktuell einen starken Trend hin zu cloudnativen Lösungen, der sich in den kommenden Jahren noch verstärken wird. Das bedeutet, dass sich die Art und Weise, wie IT-Architekturen entwickelt werden, grundlegend verändern wird«, bestätigt Pivotal-Manager Ramsperger. Dazu müsse sich auch die Kultur in den Unternehmen verändern, so Ramsperger weiter: »Auch wenn schnelle Implementierungen möglich sind, gibt es interne Prozesse, die die Weiterentwicklung bestehender Anwendungen verlangsamen können.«

Bei der Entwicklung und Integration von Anwendungen in der Cloud spielen nach Ansicht von Ramsperger Microservices eine Schlüsselrolle, zusammen mit kontinuierlicher Bereitstellung und DevOps: »Diese Konzepte sind so stark miteinander verflochten, dass sie sich nicht mehr trennen lassen.«

Das Angebot an Applikationsplattformen in der Cloud ist groß. Die von Stefan Kolb, wissenschaftlicher Mitarbeiter am Lehrstuhl für Praktische Informatik an der Universität Bamberg, zusammengestellte Übersicht auf https://paasfinder.org listet über 60 Provider auf (Bild 2).



Das Angebot an Applikationsplattformen in der Cloud ist groß (Bild 2)

Vor der Entscheidung für eine Plattform sollte sich ein Unternehmen sehr genau ansehen, ob das Angebot nicht nur die aktuellen Anforderungen erfüllt, sondern eine Entwicklung zulässt. Ein Umzug von einem Provider zum anderen ist zwar mittlerweile deutlich einfacher geworden, aber immer noch mit Aufwand verbunden. ▶

Interview

»Eine komplette Portabilität gibt es nicht«

Stefan Kolb, wissenschaftlicher Mitarbeiter am Lehrstuhl für Praktische Informatik an der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg, beschäftigt sich mit der Portabilität von Apps zwischen verschiedenen Providern.

Im Interview mit **web & mobile developer** erklärt er, warum PaaS nicht gleich PaaS ist, und wo es bei der Migration von einem Anbietern zum anderen noch am meisten hakt.

Stefan Kolb
Wissenschaftlicher Mitarbeiter an der Universität Bamberg
www.uni-bamberg.de

web & mobile developer: *Herr Kolb, 2014 haben Sie in einem Artikel zur Portabilität von Applikationen den Markt für Platform as a Service (PaaS) als weitgehend fragmentiert bezeichnet. Was hat sich seitdem verändert?*
Stefan Kolb: Der Markt ist immer noch sehr heterogen, auch wenn es einen gewissen Trend zur Konzentration gibt. Viele Start-ups wurden aufgekauft oder sind vom Markt verschwunden. So kristallisiert sich immer mehr heraus, welche PaaS-Anbieter stark sind und es auch bleiben werden. Von einer Konsolidierung sind wir bei PaaS aber noch weit entfernt.

web & mobile developer: *Worin bestehen die besonderen Herausforderungen bei der Portierung von Applikationen in PaaS, beispielsweise im Vergleich zu IaaS?*
Kolb: Um diese Frage zu beantworten, muss man zunächst klären, welche Art von Portierung man durchführen möchte. Geht es darum, eine Applikation überhaupt erst einmal von einem lokalen System in die Cloud zu bringen, oder sprechen wir über die Migration von einem Anbieter zum anderen?

web & mobile developer: *Fangen wir doch erst einmal mit der Migration in die Cloud an …*
Kolb: … da stellt sich natürlich die Hauptfrage, ob die zu migrierende Applikation überhaupt für den Einsatz in der Cloud

»Die PaaS-Provider wollen ihre Technologien durchsetzen und nicht komplett vergleichbar sein, um einen Wettbewerb rein über den Preis zu verhindern.«

geeignet ist. Um sie zu beantworten, kann man sich zum Beispiel an den zwölf Faktoren orientieren, die der PaaS-Provider Heroku für eine als Software as a Service geeignete Anwendung definiert hat. Sie finden sich auf der Webseite »The Twelve-Factor App« (12factor.net). Dazu gehören Zustandslosigkeit und »Shared Nothing«, explizit deklarierte und isolierte Abhängigkeiten und eine maximale Robustheit, bei der Prozesse sehr schnell hoch- und problemlos wieder heruntergefahren werden können, um nur einige zu nennen …

web & mobile developer: *… und wie sieht es bei der Migration zwischen PaaS-Anbietern aus?*
Kolb: Diese müssen zunächst einmal vergleichbar sein. Die spezialisierten SaaS-zentrierten Angebote fallen also schon einmal heraus. Hier handelt man sich in der Regel einen erheblichen Vendor-Lock-in ein.

Die IaaS-zentrischen und die generischen PaaS-Angebote kann man dagegen ganz gut miteinander vergleichen, weil sie ähnliche Technologien einsetzen. Betrachten wir die generischen, dann hat sich in Sachen Portierung in den vergangenen Jahren viel getan. PaaS-Plattformen wie Google App Engine, die ich als erste Generation der generischen PaaS bezeichnen würde, nutzen viele proprietäre APIs, spezielle Datenbanken und eine eigene, eingeschränkte Laufzeitumgebung. Das erschwert eine Migration erheblich. Bei der zweiten Generation wie Heroku oder Cloud Foundry ist das deutlich besser.

web & mobile developer: *Welche Faktoren erleichtern die Migration bei diesen Anbietern?*
Kolb: Sie setzen intern auf Container und Open-Source-Technologien, nutzen also keine proprietären APIs und verwenden offene Datenbanken wie MongoDB. Viele nutzen auch die von Heroku entwickelten Buildpacks, die automatisch den Typ der App erkennen und die notwendigen Laufzeitumgebungen sowie Abhängigkeiten in den Containern installieren. Wichtig ist natürlich auch, dass die zu portierende Anwendung entsprechend gestaltet ist, also auf offene Technologien setzt und sich an den erwähnten zwölf Faktoren orientiert.

web & mobile developer: *Kann ich also Applikationen nahtlos zwischen PaaS-Plattformen hin- und herschieben, wenn ich mich an die 12-Faktor-Regel halte?*
Kolb: Nein, dazu fehlen allgemein verbindliche Standards. Auch wenn viele auf dieselben Technologien setzen, so gibt es doch Unterschiede, die bei einer Migration eine Anpassung erforderlich machen.

web & mobile developer: *Warum einigen sich die Anbieter denn nicht auf gemeinsame Standards?*
Kolb: Das liegt wohl nicht im Interesse der Anbieter. Die PaaS-

Provider wollen ihre Technologien durchsetzen und nicht komplett vergleichbar sein, um einen Wettbewerb rein über den Preis zu verhindern, wie wir ihn bei IaaS ja schon haben.

web & mobile developer: *Ist die Migration immer relativ einfach oder gibt es auch Fälle, die sich schwer portieren lassen?*
Kolb: Immer dann, wenn viele Schritte automatisch ablaufen sollen, wird es schwierig, etwa bei der DevOps-Automation.

Die APIs und die Command Line Interfaces für Automatisierungsaufgaben sind von Provider zu Provider noch sehr unterschiedlich.

web & mobile developer: *Welche Kriterien sollte ich dann bei der Wahl eines PaaS-Providers anlegen?*
Kolb: Das hängt davon ab, welche Anwendungen ein Unternehmen in die Cloud bringen möchte und wie sich diese weiterentwickeln sollen. Verwendet der Kunde beispielsweise nur eine Programmiersprache, etwa PHP, sind spezialisierte Anbieter interessant. Polyglotte Applikationen müssen dagegen auf eine Plattform, die alle verwendeten Sprachen beherrscht.

Auch die Größe des Unternehmens und des adressierten Nutzerkreises spielt eine Rolle. Als kleine Firma kann man mit den knapp kalkulierten Angeboten neuer Provider häufig sehr viel Geld sparen, weil die Leistung für den Zweck ausreicht. Große Unternehmen werden dagegen schon allein aus Skalierungsgründen etablierte Plattformen bevorzugen.

»Ein Unternehmen muss sehr viele Fragen beantworten, bevor es sich auf die Suche nach einem Provider machen kann.«

Dann ist vorab die langfristige Cloud-Strategie zu klären. Soll nur das Prototyping in der Public Cloud stattfinden und die Applikation später, etwa aus Datenschutzgründen, in eine Private Cloud überführt werden? In diesem Fall sollte der Provider beide Modelle unterstützen.

Der letzte wichtige Punkt betrifft die Infrastruktur: Wie viel Flexibilität und Infrastrukturzugriff benötige ich? Soll das PaaS-System um weitere Runtime Languages erweiterbar sein? Möchte ich selbst eingreifen, etwa mit eigenen Buildpacks? Brauche ich Support für Docker-Container? Muss ich direkten Zugriff auf die Infrastruktur haben, um beispielsweise spezielle Libraries zu installieren? Sie sehen also, dass ein Unternehmen erst einmal sehr viele Fragen selbst beantworten muss, bevor es sich auf die Suche nach dem geeigneten Provider machen kann. Mit der nötigen Vorarbeit kann man dann zum Beispiel auf meinem PaaS-Vergleichsportal https://paasfinder.org sehr schnell die Provider finden, die den eigenen Anforderungen entsprechen.

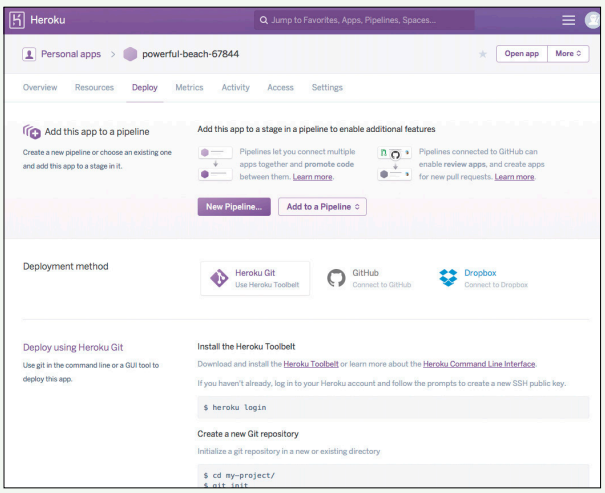
Zu den wichtigste Kriterien bei der Auswahl eines PaaS-Anbieters zählen:

- **Datenschutz und Datensicherheit:** Sollen nicht nur Entwicklung und Test, sondern auch der Produktivbetrieb in der Cloud stattfinden und sind personenbezogene oder sonstige sensible Daten betroffen, dann sind die Bestimmungen des Bundesdatenschutzgesetzes und zukünftig der Datenschutz-Grundverordnung zu beachten.
- **Elastizität:** Vor allem bei schwer vorhersagbaren Lastschwankungen sollte die Plattform automatisiert und in Echtzeit skalieren. »Die Bereitstellung von Rechenleistung für Anwendungen muss bei einigen Lösungen über Pools aktiv geplant und manuell verwaltet werden«, sagt ▶

PaaS-geeignete Applikationen

PaaS-Pionier Heroku empfiehlt zwölf Praktiken, um PaaS-fähige Applikationen zu bauen:

1. **Codebase:** eine einzige im Versionsmanagementsystem verwaltete Codebase, viele Deployments
2. **Abhängigkeiten:** explizit deklarieren und isolieren
3. **Konfiguration:** in Umgebungsvariablen ablegen
4. **Unterstützende Dienste:** als angehängte Ressourcen behandeln
5. **Build, release, run:** Build- und Run-Phase strikt trennen
6. **Prozesse:** die App als einen oder mehrere Prozesse ausführen
7. **Bindung an Ports:** Dienste durch das Binden von Ports exportieren
8. **Nebenläufigkeit:** mit dem Prozess-Modell skalieren
9. **Einweggebrauch:** robust mit schnellem Start und problemlosem Stopp
10. **Dev-Prod-Vergleichbarkeit:** Entwicklung, Staging und Produktion so ähnlich wie möglich halten
11. **Logs:** Logs als Strom von Ereignissen behandeln
12. **Admin-Prozesse:** Admin-/Management-Aufgaben als einmalige Vorgänge behandeln



Heroku zählt zu den Pionieren in Sachen PaaS

AWS-Mann Gonzalez, »das heißt, die Gesamtkapazität der PaaS ist eigentlich fix, sodass man letztlich doch wieder Gefahr läuft, Unter- oder Überkapazitäten zu haben.«

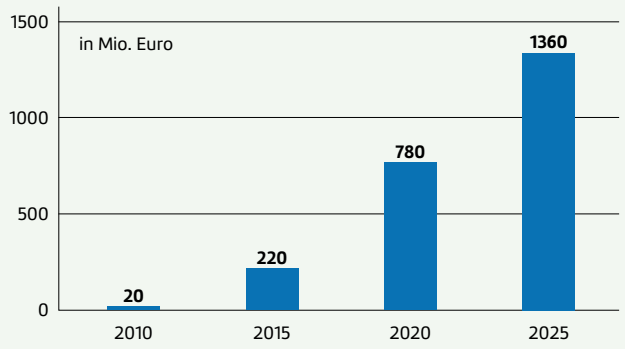
- **Funktionale Eignung:** Die Plattform muss alle verwendeten Programmiersprachen und Entwicklungswerkzeuge unterstützen sowie sich in die Deployment- und Betriebsprozesse des Kunden integrieren lassen.
- **Flexibilität:** Je mehr Ausführungsmodelle wie automatisiert bereitgestellte virtuelle Maschinen, Container, Microservices oder Web-Hosting-Umgebungen eine Plattform offeriere, desto eher könne der Kunde das für ihn am besten geeignete Modell finden, sagt Holger Sirtl von Microsoft. »Hier sind Plattformen von Vorteil, die dem Unternehmen Wahlfreiheit bieten.«
- **Schnelligkeit:** Die Plattform sollte den Kunden nicht nur durch kurze Innovationszyklen bei der schnellen Weiterentwicklung seiner Anwendungen unterstützen, sondern auch bei deren Vermarktung. »Wie lange dauert es, bis Sie mit PaaS Ihren Kunden einen Mehrwert bieten können?«, fragt Michael Ramsperger von Pivotal.
- **Preis:** Ein Vergleich des Preis-Leistungs-Verhältnisses ist bei PaaS nicht so leicht möglich wie bei IaaS, da die Performance-Klassen der Anbieter nur bedingt miteinander korrespondieren. »Es ist kaum abzuschätzen, wie viel leistungsfähiger die Premium-Instanz von Anbieter A im Vergleich zur Standard-Instanz von Anbieter B ist«, sagt Stefan Kolb. Auf der anderen Seite werden meist nur wenige Leistungsstufen angeboten, was den Vergleich im Unterschied zu IaaS mit seinen vielen Komponenten wiederum leichter macht. Die vorgefertigten Pakete geben Planungssicherheit, sagt René Büst von Crisp Research: »Ich weiß im Voraus, was ich am Ende des Monats bezahlen muss, diese Sicherheit habe ich in dem Maß bei IaaS nicht.«

Constantin Gonzalez von Amazon Web Services warnt allerdings davor, formalen Auswahlkriterien zu viel Gewicht beizumessen. Er glaubt vielmehr, dass es wichtig ist, dass die Kunden sich bei der Auswahl eines PaaS nicht in endlosen Checklisten und Vergleichstabellen verlieren und dabei ihre Endverbraucher und ursprünglichen Anforderungen und Ziele vergessen.

Links zum Thema

- Amazon / Amazon Web Services (AWS)
  https://aws.amazon.com/de
- Fortrabbit / Fortrabbit
  www.fortrabbit.com
- Fujitsu Technology Solutions / Fujitsu Cloud Services K5
  www.fujitsu.com/uk/solutions/cloud/k5/paas
- Hewlett Packard Enterprise / HPE Helion Stackato
  www8.hp.com/us/en/cloud/stackato.html
- IBM / Bluemix
  http://bluemix.net
- Pivotal / Pivotal Cloud Foundry
  http://pivotal.io/platform
- T-Systems International / AppAgile basierend auf OpenShift
  www.t-systems.com/de/de/loesungen/cloud/loesungen/appagile/platform-as-a-service-62872
- Nucleus
  https://github.com/stefan-kolb/nucleus
- Universität Bamberg
  www.uni-bamberg.de
- National Institute of Standards and Technology (NIST)
  www.nist.gov
- Gartner
  www.gartner.com
- Experton Group
  www.experton-group.de
- PaaS-Vergleichsportal
  https://paasfinder.org

Public-Cloud-Umsatz mit PaaS



Prognose: Der PaaS-Umsatz in der Public Cloud soll hierzulande bis 2025 kräftig zulegen

web & mobile developer 12/2016 PAC/Statista

Von PaaS zu Serverless Computing

Es ist gut möglich, dass die Entwicklung in anderen Bahnen verläuft und die Trennung in IaaS, PaaS und SaaS in wenigen Jahren kaum noch von Bedeutung ist. »IaaS und PaaS wachsen immer mehr zusammen«, erklärt Crisp-Analyst Büst. »Wir bei AWS denken nicht in Kategorien wie IaaS, PaaS oder SaaS, sondern betrachten Cloud Computing als eine Plattform, bei der Bausteine flexibel miteinander kombiniert werden können, um eine ganzheitliche Lösung aufzubauen«, bestätigt Constantin Gonzalez diesen Trend.

Am weitesten geht diese Integration beim sogenannten Serverless Computing, auch als Serverless Programming, Serverless Infrastructure oder Function as a Service (FaaS) bezeichnet. Während man bei PaaS über die APIs immer noch per Software für Verfügbarkeit und Skalierbarkeit sorgen muss, erledigt das in einer Serverless Infrastructure das System. Eine Art Blackbox übernimmt das Kapazitätsmanagement. »Man kann sich als Entwickler auf seine Kernaufgabe

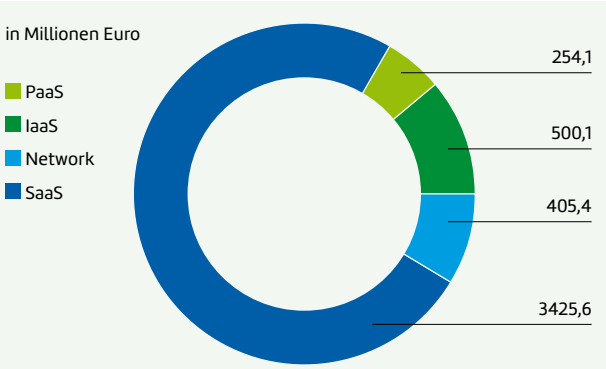
konzentrieren und muss sich nicht mehr mit den Eigenschaften der Infrastruktur oder einer Plattform beschäftigen«, so René Büst. »Wir sehen, dass viele Kunden beginnen, Aspekte von serverlosen Architekturen und NoOps-Modellen konsequent weiterzudenken«, ergänzt Gonzalez.

Zusätzlicher Komfort

Den zusätzlichen Komfort solcher Modelle erkauft man sich allerdings durch eine deutlich größere Abhängigkeit vom Anbieter. »Man muss die Applikation zwar nicht mehr an die Plattform anpassen, aber dafür sogenannte Functions schreiben, mit denen die Blackbox gesteuert wird«, erklärt der Crisp-Analyst. Diese funktionierten nur im System des jeweiligen Providers. Wer Flexibilität und Kontrolle behalten möchte, werde deshalb auch in Zukunft auf IaaS setzen, so Büst weiter.

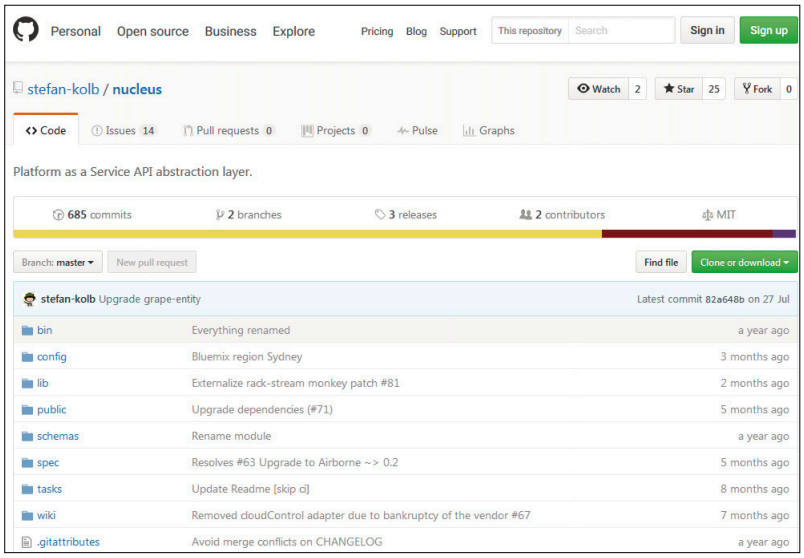
Für Microsoft-Architect Sirtl ist Flexibilität jeodch nicht alles: »Das oft zu beobachtende Streben nach Plattformunabhängigkeit ist in der Regel vom Wunsch getrieben, im Zweifelsfall die Plattform beziehungsweise den Plattformanbieter austauschen zu können. Auf der anderen Seite ergeben sich durch eine starke Bindung an eine Plattform gegebenenfalls Möglichkeiten, die Vorteile der Plattform besser nutzen zu können.«

PaaS im deutschsprachigen Markt



Ausgaben für B2B-Cloud-Services: PaaS spielt im deutschsprachigen Markt noch eine untergeordnete Rolle

web & mobile developer 12/2016 Experton Group 2015



Projekte wie Nucleus erleichtern einen Wechsel von einer Plattform auf eine andere (Bild 3)

Fazit

Zusammenfassend lässt sich feststellen: Platform as a Service erleichtert Programmierern und Administratoren die Arbeit, reduziert die Komplexität und beschleunigt die Anwendungsentwicklung. Vieles, um das man sich im eigenen Rechenzentrum oder beim IaaS-Anbieter kümmern muss, macht die Plattform von allein.

Das Mehr an Komfort erkauft man sich jedoch mit weniger Kontrolle, Transparenz und Unabhängigkeit. So hat man beispielsweise bei den meisten PaaS-Angeboten keinen direkten Zugriff auf das Betriebssystem.

Der Entwicklungsprozess selbst unterscheidet sich nicht grundsätzlich von der Anwendungsentwicklung für Application-Server. Anwendungen werden lokal spezifiziert, entworfen, entwickelt, getestet, paketiert und schließlich in die Plattform übertragen. Viele Anbieter erlauben es, mehrere Versionen der gleichen Anwendung parallel laufen zu lassen, um so zum Beispiel Live-, Stage- und Testumgebungen anzubieten.

In puncto Vendor-Lock-in kann allerdings Entwarnung gegeben werden. Ein Wechsel von einer Plattform auf die andere ist heute ohne großen Aufwand möglich, auch wir wenn noch weit von einer nahtlosen Migration entfernt sind. Projekte wie Nucleus (https://github.com/stefan-kolb/nucleus) könnten hier in naher Zukunft Abhilfe schaffen (Bild 3).

Auch die Open-Source-Plattformen Cloud Foundry und OpenShift sind gute Ansatzpunkte für eine möglichst große Unabhängigkeit. Setzen die Private Cloud im eigenen Rechenzentrum und der Provider der Wahl auf eine dieser Plattformen, lassen sich ohne größere Schwierigkeiten auch Hybrid-Cloud-Modelle auf PaaS-Basis realisieren. ■

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

MODERNE SOFTWARE-ENTWICKLUNG

Entwicklung, die begeistert

Die moderne Software-Entwicklung ist geprägt von zahlreichen neuen Begiffen, Technologien und Methoden.

Wieso gelingt es Unternehmen, trotz der permanenten Veränderungen des technologischen Umfelds im IT-Geschäft dauerhaft erfolgreich am Markt zu sein, tolle Produkte am laufenden Band zu produzieren und immer mehr zufriedene Mitarbeiter anzuziehen? Wie bringe ich meine Kunden und Mitarbeiter dazu, dass sie ebenso begeistert bei der Sache sind?

In den letzten Jahren haben in der modernen Software-Entwicklung viele neue Begriffe, Technologien und Methodiken Einzug gehalten. Je nach gegebenem Kontext haben sich einige dieser Konzepte bewährt und wurden Arbeitsabläufe und Prozesse übernommen. Andere Ansätze eignen sich im konkreten Umfeld weniger, weil die Lösung nicht ganz zum Problem passt. So ist es aus naheliegenden Gründen beispielsweise recht wenig sinnvoll, ein Software-Hotline-Team mit Scrum organisieren zu wollen. Wieder andere Konzepte und Buzzwords stellten bald ihre notorische Untauglichkeit für die Praxis unter Beweis und verschwanden ebenso schnell wieder, wie sie gekommen waren.

Für erfolgreich agierende Unternehmen und Produktteams ist aber die Tatsache, ob Scrum oder Kanban, ob SOA oder Microservices, ob dieses Framework oder jenes zu verwenden und besser geeignet sei, letztlich eine Detailfrage. Die technologischen Grundlagen ändern sich in der Softwarebranche einfach zu rasant, als dass sie in irgendeiner Form als Fixpunkt dienen könnten. Oder noch prägnanter: Technologie ist sehr häufig einfach Mittel zum Zweck. Das gilt natürlich nicht pauschal und in jedem Fall, aber wenn wir uns nur einmal den Bereich der Webstandards und Web-Frameworks herausgreifen (Bild 1), wird deutlich, dass technologische Entscheidungen heute mehr denn je einem steten Wandel unterworfen sind. Früher noch eine feste Konstante, auf die man über Jahre hinaus bauen konnte, werden bestehende Ansätze und Verfahren in immer kürzeren Zeitabständen hinterfragt, revidiert oder zugunsten einer noch besser geeigneten Lösung verworfen.

Wir alle sind Bestandteil einer Branche, die in einem rasanten Entwicklungstempo voranschreitet, und dies wird sich auch auf absehbare Zeit nicht ändern. Es gilt also, andere Fixpunkte zu finden, an denen man sich orientieren kann und an denen Entscheidungen gemessen werden.

Am Ende steht immer der Kunde

Das Endziel eines handelsüblichen Softwareprojekts ist normalerweise ein zufriedener Kunde, der nicht nur dankbar seine Rechnung bezahlt, sondern darüber hinaus auch mit Folgeaufträgen winkt. Gleich an zweiter Stelle steht meist der Anwender, der ja gerade bei Webanwendungen mit dem Auftraggeber allerhöchstens eine Schnittmenge bildet.

Die Qualität einer Software aus Sicht dieser beiden Zielgruppen ist in erster Näherung durch drei Hauptziele definiert:

Funktion ist der Dreh- und Angelpunkt jeder weiteren Betrachtung. Wenn die Lösung nicht den spezifizierten Funktionsumfang abdeckt oder die eingebauten Features nicht oder nicht richtig funktionieren, kann man sich jede weitere

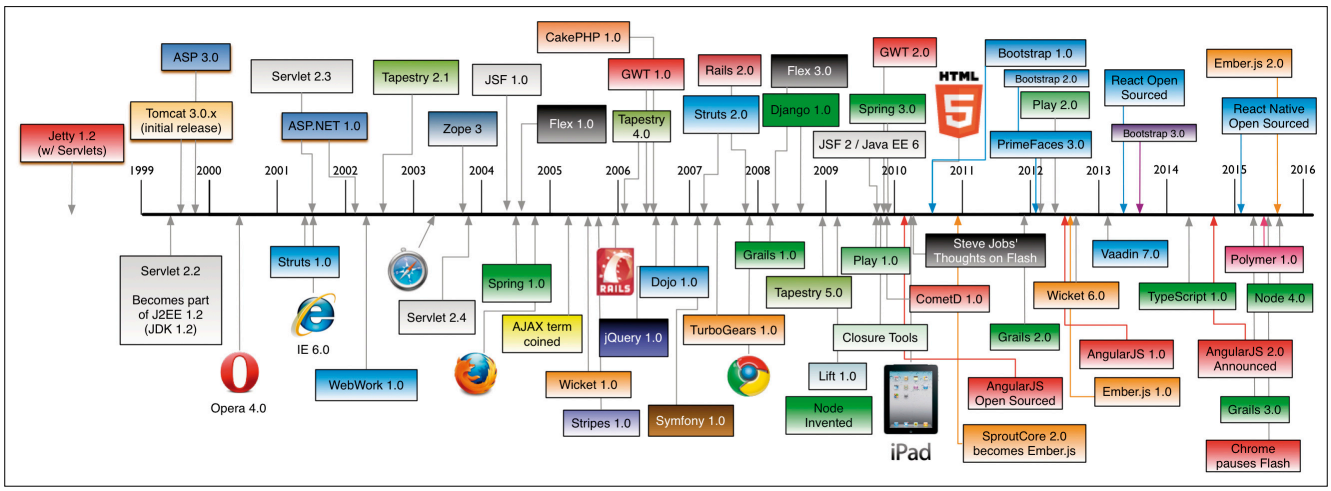


Foto: Matt Raible, Github-Repository

Entwicklungsgeschichte: Die Historie der Web-Frameworks (Bild 1)

Jeff Atwood über KISS und YAGNI

Rico Mariani von Microsoft hat in seinem Blog ein wichtiges Thema aufgegriffen:

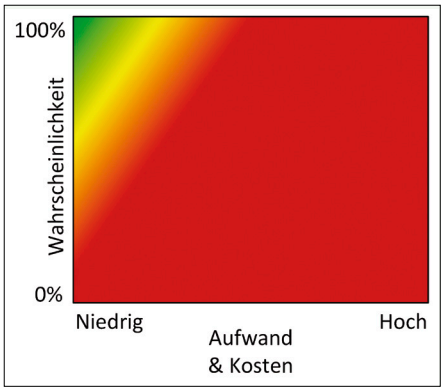
»Aber sehr oft sehe ich, dass der Code auf eine aufwendige Art und Weise geschrieben wurde, wo es auch ein einfacheres Modell getan hätte. Welcher Programmierer-Glaubensrichtung Sie auch immer anhängen, Sie werden mir zustimmen, dass komplexere Sprachabstraktionen nicht automatisch auch besseres Design bedeuten. Im Gegenteil, jedes anspruchsvollere Sprachfeature kommt mit einem Nachteil daher und muss diesen zunächst ausgleichen – durch Vorteile wie Klarheit, einfache Wartbarkeit, Performance und so weiter.

Als Entwickler denke ich, wir sind allzu oft überoptimistisch bezüglich der Wiederverwendbarkeit unserer eigenen Lösungen. Wir bauem aufwendige OOP-Frameworks um Dinge, die diesen Komplexitätslevel möglicherweise gar nicht rechtfertigen. Um diesen inneren Drang zu bekämpfen, empfehle ich die strikte Befolgung des YAGNI-Prinzips. Bauen Sie das, was Sie brauchen, wenn Sie es brauchen. Refaktorisieren Sie agggressiv. Verbringen Sie nicht zu viel Zeit damit, für grandiose, unbekannte zukünftige Szenarios zu planen. Gute Software kann in diese Anforderungen hineinwachsen.«
https://blog.codinghorror.com/kiss-and-yagni

Diskussion eigentlich sparen. Oder um Larry Osterman zu zitieren: »I can make it arbitrarily fast if I don't actually have to make it work« – die beste Performance bringt uns nicht weiter, wenn das Ergebnis nichts taugt. Womit wir auch gleich den zweiten Punkt identifiziert haben, an dem Kunde und Anwender interessiert sind. Würde Google uns das ansonsten korrekte und völlig erwartungskonforme Suchergebnis anstatt wie üblich in Bruchteilen einer Sekunde erst nach fünf Minuten oder zwei Stunde servieren, wären sie ziemlich sicher innerhalb kürzester Zeit aus dem Geschäft.

Was aber ist das dritte Ziel? Es ist im Gegensatz zu den beiden anderen ein nichtfunktionales Ziel, gleichwohl nicht weniger wichtig. Hier geht es um den finanziellen Aspekt, der sich in erster Linie darum dreht, dass das georderte Produkt dann auch im geforderten Zeitrahmen abgeschlossen wird und keinesfalls den veranschlagten Kostenrahmen sprengt. Douglas Hofstadter prägte dazu das in seinem Buch »Gödel, Escher, Bach« veröffentlichte Hofstadtersche Gesetz, dass Vorhaben stets länger dauern als ursprünglich geplant – auch dann, wenn man das Hofstadtersche Gesetz bereits bei der Planung berücksichtigt. Dem ist nicht viel hinzuzufügen, außer dass sich die Kosten nicht nur auf das Zeitbudget auswirken, sondern häufig eben auch finanziell. Das zum Zeitpunkt der Entstehung dieses Artikel noch laufende Langzeitexperiment BER bietet hier ein sehr anschauliches Beispiel, wie so etwas in der Praxis aussehen kann.

Die Forderung nach Einhaltung von finanziellem und Zeitbudget beschränkt sich dabei naturgemäß nicht auf das initiale Produkt in Version 1.0. Die Mehrzahl der Produkte wird



Kosten und Wahrscheinlichkeiten: Die schematische Darstellung der Zusammenhänge (Bild 2)

über eine gewisse Lebensdauer hinweg weiterentwickelt und gepflegt. Ob dies traditionell durch regelmäßige Releases geschieht oder ob man über rollende Releases in schneller Folge immer wieder kleine Änderungsinkremente in die laufende produktive Version einfließen lässt, ist dabei sekundär. Entscheidend sind auch hier Zeit und Kosten (Bild 2).

Jeder Entwickler wird zustimmen, dass eine gut gewartete und strukturierte Codebasis das Leben eines Wartungsprogrammierers deutlich leichter macht. Ein unstrukturierter Haufen Brownfield-Spaghetticode, wild aufeinandergetürmte Modulhalden, deren subtile Querbeziehungen und Abhängigkeiten sich erst nach längerem intensiven Studium der Quelltexte langsam erschließen, das ist wohl der Albtraum jedes Entwicklers. Natürlich wird man während der Entwicklung neuer oder beim Optimieren bestehender Features immer wieder in einer Phase stecken, wo der Code eben nicht aufgeräumt ist. Ebenso selbstverständlich wird man im Verlauf inkrementeller Entwicklungsschritte unweigerlich an den Punkt kommen, wo bestehender Code mindestens refaktorisiert werden muss, um ein gewisses Niveau zu halten. Und ebenso menschlich ist es auch, wenn letzte Bugfixes kurz vor dem Release unter Zeitdruck vielleicht nicht hundertprozentig sauber ausprogrammiert, sondern notfalls auch mal grob mit virtuellem Duct Tape zusammengeheftet werden.

So legitim die als virtuelles Duct Tape bezeichnete Notlösung ist, wenn kurzfristig andere Prioritäten bestehen, so klar dürfte auch sein, dass wir dies im Gegenzug mit technischen Schulden bezahlt haben. Handelt es sich beim Produkt lediglich um einen Wegwerf-Prototyp, spielt die Wartbarkeit als drittes Hauptkriterium nach Funktion und Performance keine wesentliche Rolle. Für jedes Produkt aber, das noch eine längere Lebenszeit vor sich hat, ist Wartbarkeit die wohl wichtigste nichtfunktionale Anforderung überhaupt.

Wartbarkeit versus technische Schulden

Schaut und hört man sich um, spielt der Wartbarkeitsaspekt in den Gesprächen lediglich eine untergeordnete Rolle, wenn er denn überhaupt diskutiert wird. Sicher liegt das zum Teil auch daran, dass sich Wartbarkeit eher schlecht als primäres Produktfeature verkaufen lässt. Nicht an den Kunden, aber auch nicht vom Entwickler an das eigene Management. Eher noch das Gegenteil: Wenn es gilt, einen bestimmten Codeteil zu refaktorisieren, erntet der geneigte Entwickler durch- ▶

aus auch mal verwunderte Blicke. Warum sollte man Zeit und Geld in etwas investieren, was doch anscheinend gar nicht kaputt ist und prima funktioniert?

Der Begriff der technischen Schulden sagt es ja bereits aus. Man nimmt hier einen Kredit auf die Zukunft auf. Wird die Codebasis in ungepflegtem Zustand hinterlassen, werden sich im Lauf der Zeit mit jeder weiteren Änderung und Erweiterung auch immer neue, für sich genommen kleine Schäden und Unsauberkeiten ansammeln, die mittelfristig dann zu eben jenem Morast führen, den wir Entwickler als matschiges Brownfield fürchten. Mit allen Seiteneffekten, die sich dadurch ergeben.

Selbst kleinere Eingriffe arten in einem solchen Codeverhau schnell in komplexe Änderungen aus. Es fällt schwer, Auswirkungen auf andere, möglicherweise betroffene Codeteile und Features abzuschätzen. Die Arbeit ist fehlerträchtig und geht infolgedessen auch deutlich langsamer von der Hand. Ganz abgesehen von den Bugs, die wir trotz aller Vorsichtsmaßnahmen und Tests eingebaut haben, weil wir eben doch irgendeine obskure Abhängigkeit übersehen haben – deren Beseitigung uns erneut in den Morast führt. Unterm Strich lässt sich zusammenfassen, dass schlecht wartbarer Code nicht nur die Laune, sondern auch die Produktivität des Entwicklers dramatisch absenkt.

Durchlaufzeiten pro Projekt

Und das ist genau die Stelle, an der das Management wieder hellhörig werden sollte. Denn geringere Produktivität heißt ja, dass pro Zeitraum weniger Produkt und somit zunächst auch weniger Umsatz erzeugt wird. Andersherum betrachtet steigen die Durchlaufzeiten pro Projekt an, unter Umständen sogar recht deutlich. Wartbarkeit ist tatsächlich ein wesentlicher Aspekt, der sich nur eben erst mittel- und langfristig bemerkbar macht. Leider ist es so, dass in unserer Welt nicht selten den Quartalsergebnissen größere Signifikanz beigemessen wird als den erst auf längere Sicht drohenden Einbußen durch technische Schulden.

Dazu kommt noch der nicht unerhebliche Einwand des YAGNI-Prinzips: Warum sollte ich heute Zeit und Geld in ein Refactoring investieren, wenn ich noch nicht mal weiß, ob ich das überhaupt benötigen werde? Natürlich ist es wirtschaftlicher Unsinn, Mittel in potenziell unnütze Dinge zu stecken. Leider ist es aber auch nicht ganz so einfach, denn die Investition in wartbaren Code ist nicht einfach irgendeine Investition, sie ist eher mit vorbeugender Instandhaltung vergleichbar. Natürlich kann man auf die regelmäßige Durchsicht kostenintensiver technischer Anlagen und Geräte verzichten, schließlich könnte sich ja nächste Woche schon ein Totalschaden ereignen. Natürlich kann man die energetische Sanierung eines in die Jahre gekommenen Bauwerkes als unnötig abwinken – schließlich weiß man ja auch nicht, ob man bei den Heizkosten überhaupt noch Mieter findet und das Haus nicht sowieso bald verkauft oder abgerissen wird. Alle drei Fälle haben aber eines gemeinsam: Es handelt sich zu einem guten Teil um selbsterfüllende Prophezeiungen. Bezüglich Wartbarkeit gilt es also wie so oft, einen wirtschaftlich sinnvollen Kompromiss zu finden, der nicht nur auf kurzfristige Ziele bedacht ist, sondern eben auch die potenziellen Auswirkungen mittelfristiger Geschäftsrisiken adäquat einpreist.

Einfachheit ist die höchste Stufe der Vollendung

Neben den bereits diskutierten Gründen gibt es mindestens noch einen weiteren Grund, der auf Wartbarkeit als solche einen großen Einfluss hat: Einfachheit. Je nach persönlicher Vorliebe kann man das nun mit obigem Zitat von Leonardo da Vinci ausdrücken, oder man greift eher zum kernigen »Keep it simple stupid«, das den einschlägigen Quellen zufolge aus der US-Navy entlehnt wurde. Die Aussage ist in beiden Fällen dieselbe: Halte die Dinge einfach, komplex wird es noch früh genug.

Für ausreichend vorhandene Komplexität sorgt nicht zuletzt ein ganz aktueller Trend, der in der Zukunft noch mehr an Bedeutung gewinnen wird. Die Rede ist natürlich von Parallelisierung, von verteilter Software und hochgradig asynchronen Abläufen. Wie wohl jeder bestätigen kann, der schon einmal eine Race Condition debuggt hat, kann die Fehlersuche in einem solchen System ziemlich schnell ziemlich komplex werden. Nicht selten beginnt es schon damit, dass sich das Verhalten in der Testumgebung zunächst gar nicht reproduzieren lässt.

Ganz gleich ob es um Algorithmen, Prozesse, Bedienoberflächen oder Software-Architektur geht, ganz allgemein gilt: Je komplexer, desto höher die Wahrscheinlichkeit, dass Fehler gemacht werden. Je schneller eine Sache zu verstehen, je

Hyperbolische oder zeitinkonsistente Diskontierung

Wir Menschen tendieren dazu, uns inkonsistent zu verhalten, wenn es um zeitlich versetzte Entscheidungen geht.

Ganz gleich ob es sich dabei um die guten Vorsätze zum Jahresanfang oder um finanzielle Entscheidungen dreht, häufig wird der schnelle kleinere Gewinn einem langfristigen Investment mit deutlich größerer Rendite vorgezogen. Das leckere Stück Erdbeertorte mit Sahne zum Kaffee ist mental ein lohnenderes Ziel als der Vorsatz, durch langfristige bewusste Ernährung zum Idealgewicht zu finden. Stünden aber sowohl das sofortige Erreichen des Idealgewichtes als auch die Leckerei beide zum gleichen Zeitpunkt zur Wahl, entschiede man sich rational. Dieser Effekt ist als Zeitinkonsistenz oder hyperbolische Diskontierung bekannt. Genau dieses überproportionale mentale Abwertung einer dauerhaft sauberen Codebasis auf lange Sicht gegenüber dem schnellen Erfolg eines wilden Hacks heute lässt sich auch unter Entwicklern beobachten.

Wie kann man dem entgegenwirken? Indem man sich diesen Effekt zunächst bewusst macht und ihm dann willentlich entgegenwirkt. Der schnelle Hack ist nicht per se schlecht, denn es mag durchaus gute Gründe dafür geben. Wirklich schlecht wäre allerdings, die Codebasis mit der Ausrede »never touch a running system« in diesem dreckigen Zustand zu belassen und damit deren Erosion Vorschub zu leisten. Der Hack von heute ist das Brownfield von morgen.

intuitiver eine Oberfläche oder ein API zu handhaben ist, je geläufiger die eingesetzten Techniken, Bezeichnungen und Methodiken sind, desto weniger fehlerträchtig ist es, damit zu arbeiten, sie für neue Anforderungen zu modifizieren, zu erweitern und Fehler darin zu beheben. Klingt einfach, ist es aber in der Praxis nicht immer. Erstens, weil den mittels Software abzubildenden Sachverhalten selbst ja bereits eine gewisse Komplexität innewohnt. Zweitens, weil die Frage, ob zum Beispiel ein Algorithmus eher komplex oder eher simpel ist, auch vom Kenntnisstand des Betrachters abhängt.

Im Lauf der Zeit haben sich ein paar Orientierungspunkte herauskristallisiert, die man gern unter Evolvierbarkeit zusammenfasst: Dazu gehören unter anderem Punkte wie lose Kopplung, die Trennung von Belangen (Separation of Concerns) und das Prinzip der geringstmöglichen Überraschung (Principle of Least Astonishment).

Wohlgemerkt: Der Ruf nach Einfachheit heißt nun aber natürlich nicht, dass etwa einem einfachen, dafür weniger leistungsfähigen Algorithmus der Vorzug gegeben werden soll gegenüber einem komplexen, ausgefeilten Algorithmus, der die Sache deutlich effizienter erledigt – das wäre weder im Sinne des Produkts noch in dem des Kunden. Natürlich sollte man sich nach Möglichkeit immer für eine dem Problem angemessene Lösung entscheiden. Angemessen heißt nichts anderes, als dass die Lösung einerseits nicht gar zu simpel gestrickt sein sollte, andererseits aber auch nicht grundlos overengineered. Natürlich kann man faktisch jede Aufgabenstellung auf beliebig aufwendige Weise lösen, sodass man für beliebige antizipierte, vielleicht eintreffende zukünftige Anforderungen bestens gerüstet sein wird. Die vielen Konjunktive deuten das Problem mit diesem Ansatz bereits an. Die Frage ist nämlich, ob das, was da konstruiert werden soll, auch tatsächlich den Projektzielen entspricht.

Welt aus Wahrscheinlichkeiten

Es könnte gut sein, dass eine bestimmte Architektur oder Technologie aus ganz anderen Gründen vorgeschlagen wurde, die mit dem Projekt gar nichts zu tun haben. Ein Indikator kann die Häufung von aktuellen Buzzwords sein, zum Beispiel REST. Erstaunlich weit verbreitet ist die Ansicht, dass jede verteilte Anwendung, die etwas auf sich hält, heutzutage RESTful sein sollte. Klassische Remote Procedure Calls (RPC) werden in diesem Mindset als veraltete Technologie aus dem letzten Jahrtausend wahrgenommen. Dass Entwickler und Programmierer mit dieser Art Entscheidungen, die nicht selten von oben aufdiktiert werden, dann ihre liebe Not haben, die Realität mit der Theorie notfalls mit Gewalt in Einklang zu bringen, kann man anschließend auf StackOverflow und Co. gut beobachten. Die gerade im IT-Bereich auffallend häufig anzutreffende Tendenz, die aktuelle Modetechnologie auf wirklich jedes Problem anwenden zu wollen, ergibt leider viel zu oft weder wirtschaftlich noch technisch Sinn.

Unglücklicherweise ist in der YAGNI-Diskussion auf allen Seiten eine ganze Menge Dogmatismus im Spiel. Die persönliche Meinung des Autors fällt recht pragmatisch aus. Risiken sind in jedem Projekt vorhanden, das liegt in der Natur der Sache. Wir leben nun mal in einer Welt aus Wahrscheinlichkeiten. Die Kunst ist, diese so auszubalancieren, dass unterm Strich das bestmögliche Ergebnis herauskommt. Folglich scheint es sinnvoll zu sein, je nach Eintrittswahrscheinlichkeit und konkretem Setup unter Umständen einen kleinen zusätzlichen Aufwand heute in Kauf zu nehmen, um sich zumindest die Option einer späteren Erweiterbarkeit bewusst durch entsprechende Designentscheidungen offenzuhalten. Hierbei sollte man aber trotzdem unbedingt immer den heute garantiert zu treibenden Aufwand gegen den zukünftigen und mit einer Wahrscheinlichkeit kleiner 100 Prozent behafteten Nutzen genau abwägen.

Das funktioniert im Prinzip nicht viel anders als eine Option an den Finanzmärkten. Gegen die sofortige Zahlung eines kleinen Betrags sichert man sich die Möglichkeit, ein bestimmtes Wertpapier in der Zukunft zu einem bereits heute festgelegten Preis zu kaufen beziehungsweise zu verkaufen. Das Entscheidungsproblem, mit dem wir es hier zu tun haben, hat nämlich nicht nur die zwei Zustände »Erweiterung nötig: ja oder nein«. Gar nicht so selten tritt nämlich auch der Fall ein, dass es zwar zukünftige Anforderungen gibt, diese aber von den ursprünglich antizipierten Annahmen mehr oder weniger deutlich abweichen. Wäre die ursprüngliche Variante umgesetzt worden, hätte man unter Umständen viel Zeit in eine letzten Endes nur teilweise geeignete oder sogar ganz untaugliche Lösung investiert. Hingegen kann sich die Entscheidung, bestimmte Erweiterungsmöglichkeiten absichtlich offenzuhalten, ohne aber bereits heute alle Eventualitäten fertig auszuprogrammieren, als die kleinere, aber wertvollere Investition in die Zukunft entpuppen.

Wie wir bereits festgestellt hatten, steht der Kunde und Anwender im Fokus unserer Bemühungen, da wir im Regel- ▶

Fehler und Zufälle sind Triebfedern der Innovation

Auf einem Technical Council sprach der 3M-Mitarbeiter Spencer Silver darüber, einen superstarken Klebstoff für den Einsatz im Flugzeugbau zu entwickeln.

Das Ergebnis war dann allerdings nur ein schwacher Klebstoff, zu der Zeit faktisch eine Lösung ohne zugehöriges Problem. Arthur Fry, ebenfalls bei 3M angestellt und außerdem Sänger im Kirchenchor, hatte ständig Probleme mit den Lesezeichen in seinem Gesangbuch. Fry bemerkte zwei wichtige Eigenschaften von Silvers Klebstoff, die ihn für Lesezeichen geeignet machten: Die Notiz war wiederverwendbar, und sie war leicht und ohne Rückstände entfernbar. So wurde die Post-it Note geboren.

3M lernte frühzeitig, wie wichtig diese Faktoren sind und verankerte dies später in der bekannten Unternehmensregel, nach der Mitarbeiter 15 Prozent ihrer Arbeitszeit an einem beliebigen Projekt ihrer Wahl arbeiten dürfen. »Ermutigen Sie die Leute zu kreativem Experimentieren. Wenn wir Zäune bauen, bekommen wir Schafe. Gebt ihnen den Raum, den sie brauchen.«
https://hbr.org/2013/08/the-innovation-mindset-in-acti-3
http://money.cnn.com/magazines/fsb/fsb_archive/2003/04/01/341016

fall ja an einer längerfristigen Geschäftsbeziehung interessiert sind. Im System agieren neben dem Kunden, dessen Vertreter in Scrum bekanntlich der Product Owner ist, auch noch andere Parteien. Jede Einzelne davon hat eigene Erwartungen und Anforderungen an den Gesamtprozess. Der Kunde erwartet ein Produkt, das seine funktionalen und nichtfunktionalen Anforderungen erfüllt. Um ihre Aufgabe erwartungskonform zu erfüllen, haben Mitarbeiter und Team zu Recht den Anspruch auf adäquate Vorbereitung, den Zugriff auf relevante Informationen und Feedback.

Um einen möglichst reibungslosen Ablauf der mit dem finalen Ziel »Produkt geliefert und vom Kunden abgenommen« zu gewährleisten, sind viele Dinge nötig, die nicht zum primären Entwicklungsprozess gehören. Konflikte müssen moderiert und gelöst werden, sowohl bezüglich technischer Ressourcen als auch bezüglich möglicher Differenzen im Team selbst. Einflüsse von außen, wie etwa die ungeplante Einforderung zusätzlicher Aufgaben durch das Team, müssen koordiniert werden. Hindernisse, die eine erfolgreiche Umsetzung der Arbeitsaufgaben behindern oder blockieren, müssen identifiziert und behoben werden.

In Scrum fällt die Aufgabe, dem Team diese sprichwörtlichen Steine aus dem Weg zu räumen, der Rolle des Scrum Master zu. In traditionell geführten Projektteams ist dafür nominell der Projektleiter in der Pflicht. Während der Scrum Master naturgemäß bereits mit entsprechenden Befugnissen ausgestattet sein sollte, müssen einem extra dazu bestellten Mitarbeiter gleichermaßen Kompetenzen und Autorität an die Hand gegeben werden, damit er seine Aufgabe auch tatsächlich erfüllen kann.

Das gilt generell. Wenn ein Mitarbeiter eine Aufgabe ohne die dazu nötigen Befugnisse zugewiesen bekommt, wird das Ergebnis im besten Fall unbefriedigend sein. Dass dabei anfangs auch Fehler passieren werden oder Dinge einfach anders angepackt werden, ist völlig normal und kein Grund zu Panik oder Überreaktion. Es reicht nicht, einfach nur Aufgaben zu übertragen, man muss darüber hinaus auch ein gewisses Vertrauen mitgeben. Es sollte sich mittlerweile herumgesprochen haben, dass Mikromanagement ebensowenig ein guter Führungsstil ist wie das völlige Fehlen jeglicher Führung, bei der Mitarbeiter und Team orientierungslos vor sich hin werkeln. Beide Extreme sind gleichermaßen kontraproduktiv.

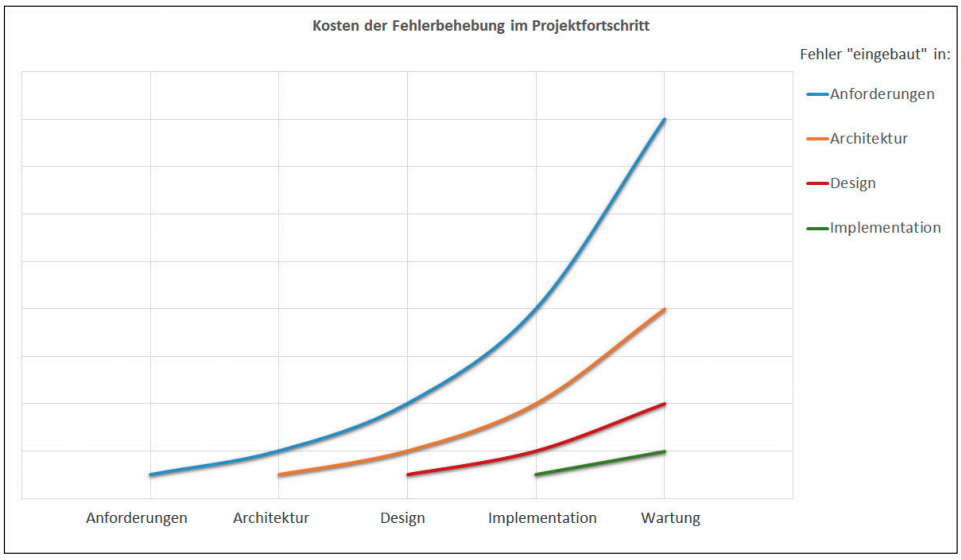
Mitarbeiter erwarten zu Recht, dass jeder, der eine Rolle im Unternehmen übernimmt, dieser Rolle auch gerecht wird, indem er sie tatkräftig mit Leben füllt. Oder um es mit Peter Drucker zu sagen: »Leadership is defined by results, not by attributes«. Gerade bei Mitarbeitern, die aus einer technischen Rolle, zum Beispiel der des Entwicklers, in eine Führungsposition hineinwachsen, beobachtet man jedoch oft einen Fokus auf die Tätigkeiten, die eher mit der ursprünglichen Position konnotiert sind als mit der Wahrnehmung des neuen Aufgabengebiets. Hier kann ein Coaching oder die Mentorschaft durch einen erfahrenen Kollegen hilfreich sein.

Man sollte sich immer wieder vor Augen halten, dass Ziele wie »erfolgreiche Produkte, hoher Umsatz und zufriedene Kunden« in einem Prozess entstehen, der erst durch die Zusammenarbeit vieler Teammitglieder möglich wird, von denen jeder seinen Bereich und sein Aufgabenfeld hat. Jedes davon trägt gleichermaßen zum Ergebnis bei. Wenn der Entwickler in der Leerlaufschleife Däumchen dreht, weil seine Hardware gerade nicht einsatzbereit ist, weil er auf wichtige Informationen wartet oder einfach die Arbeitsaufgabe nicht klar ist, dann ist es wichtig, diese Störung schnellstmöglich zu beheben. Natürlich sind die Bereitstellung einer Festplatte und die Koordinierung von Informationen, Terminen und Stakeholdern keine primär wertschöpfenden Tätigkeiten, dennoch ist jede eine unverzichtbare Aufgabe, ohne deren Erledigung die Projektziele in Gefahr geraten.

Ergebnisorientierte Kommunikation

Aber der Vorgesetzte oder der Scrum Master ist nur ein Teil dieser Medaille. Denn auch das Teammitglied selbst ist in der Verantwortung. Kommunikation funktioniert am besten, wenn sie in beiden Richtungen fließt. Wenn dem Entwickler die Arbeitsaufgabe für ein Feature nicht klar ist, weil ihm diverse Puzzleteilchen zum vollen Verständnis fehlen, muss er diese Informationen eben aktiv einfordern.

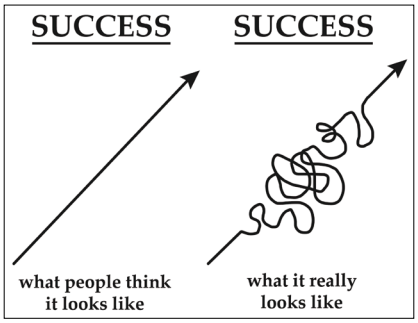
Wenn der Scrum Master nicht weiß, dass auf einer Maschine das dort dringend benötigte Programm XYZ ständig abstürzt oder die Netzwerkverbindung lahmt, müssen diese Dinge an die zuständigen Stellen weitergeleitet werden. Wenn eine kundenseitige Anforderung an technischen Hürden scheitert, die das initial geplante Budget zu sprengen



Kosten der Fehlerbeseitigung im Projektfortschritt (Bild 3)

drohen, sollte dies frühzeitig bekannt werden.

Wenn die Performance des Entwicklungsteams leidet, weil man ständig damit beschäftigt ist, Fehler einzelner Kollegen auszubessern, so ist besagter Kollege mit den zugewiesenen Aufgaben vielleicht einfach nur überfordert. Ob der Mitarbeiter nun eine Schulung bekommt, mittels Pair Programming auf einen aktuellen Wissensstand gebracht wird oder vielleicht eine völlig andere Arbeitsaufgabe erhält, das wird das Team nicht unbedingt selbst entscheiden können. Es kann und muss aber dafür sorgen, dass Hindernisse jeder Couleur notfalls entsprechend eskaliert werden, anstatt passiv auf eine wundersame Fügung zu warten.



Innovation in Theore und Praxis (Bild 4)

Zusammenfassend lässt sich festhalten, dass der Einzelne auf jeder Hierarchiestufe immer auch Mitglied eines Teams ist. Diese Ebene wird dabei nach unten im Wesentlichen durch die Möglichkeit gebildet, Aufgaben zu delegieren. Die Grenze nach oben ist charakterisiert durch die jeweiligen Entscheidungskompetenzen.

Gesunde Fehlerkultur

Zu ergebnisorientierter Kommunikation gehört auch die Etablierung einer gesunden Fehlerkultur im Unternehmen. Vielen ist die Grafik in Bild 3 über die Kosten der Fehlerbehebung in Softwareprojekten bekannt, die Steve McConnell in seinem Klassiker »Code Complete« verwendet hat. Die Grafik selbst geht auf noch ältere Studien von Barry Boehm zurück. Deren Aussagekraft wurde in den vergangenen Jahren immer mal wieder kritisiert, diskutiert und gelegentlich auch angezweifelt. Unabhängig davon, ob die Verhältnisse in der täglichen Realität nun exakt abgebildet wurden oder nicht, kann man schon rein intuitiv der Kernaussage zustimmen. Wenn die Anforderungen bereits fehlerhaft sind, das Problem jedoch erst in der Betaversion entdeckt wird, wird die Korrektur mit ziemlicher Wahrscheinlichkeit deutlich aufwendiger werden, als wenn dies bereits während des Entwurfs stattfindet. Wird das Produkt bereits produktiv eingesetzt, müssen neben der Softwarekorrektur vielleicht zusätzlich auch noch Benutzerdaten konvertiert werden, was den Aufwand weiter in die Höhe treibt.

Und genau aus diesem Grund ist es auch eine sinnvolle Idee, alle Stakeholder bereits frühzeitig ins Boot zu holen, wie es in allen Schattierungen agiler Ansätze mittlerweile zum Glück gang und gäbe ist. Die Grafik hält aber noch eine andere Aussage bereit, auch diese inzwischen State of the Art agiler Praktiken: Häufige, schnell aufeinanderfolgende Releases auch unfertiger Versionen mit kleinen Inkrementen sind besser. Die Entwickler erhalten frühzeitig Feedback, der Kunde bekommt ein Gefühl dafür, wo das Projekt steht, und kann schon damit beginnen, Features zu testen und erste praktische Erfahrungen zu sammeln. Erfahrungsgemäß führt dieses Feedback zwar wiederum zu Änderungen in den Anforderungen, was aber durchaus gewollt ist. Auch wenn diese neuen Anforderungen vielleicht nicht im ersten Meilenstein umgesetzt werden, so geben sie doch bereits die ungefähre Richtung für den weiteren Projektfortschritt vor – durchaus ein Vorteil für die weitere Planung.

Schnelle Releases mit kleinen Inkrementen reduzieren die potenziellen Auswirkungen etwaiger Fehler. Mit sinkenden Fehlerkosten steigt auch der Anreiz, Dinge bewusst einfach mal auszuprobieren. Insbesondere bei Webanwendungen hat es sich bewährt, neue Features über sogenannte Feature-Toggles abschaltbar zu implementieren. Auf diese Weise kann man auch mit geringem Aufwand ein Feature testweise für einen begrenzten Anwenderkreis freischalten.

Fail fast, fail often

Ob schnelle Releases im Web oder langsamere Releases auf dem Desktop, stets ist die Angst vor Fehlern ein schlechter Ratgeber, der die Innovationskraft des Teams und des Unternehmens hemmt. Einer Studie von 3M zufolge spielt der Zufall bei den weitaus meisten Patenten, die 3M im Lauf ihrer Unternehmensgeschichte angemeldet hat, eine durchaus signifikante Rolle. Gerade Innovation hat viel mit Ausprobieren, Fehlschlag, mit Lernen und iterativer Verbesserung zu tun. Auch das im agilen Umfeld wohlbekannte Kaizen lässt sich nicht grundlos mit »Wandel zum Besseren« übersetzen, oder etwas griffiger mit »stetige Verbesserung« (Bild 4).

Wer nach absoluter Perfektion im ersten Anlauf sucht, wird das Feature oder Produkt nie auf den Markt bringen. Etwas verbessern lässt sich schließlich immer. Dasselbe gilt, wenn ein Entwickler seine Implementation eines neuen Features oder einer algorithmischen Idee vor dem Team präsentiert. Es wird zu diesem Zeitpunkt nicht perfekt sein, das ist ihm selbst durchaus klar. Was er will, ist positives Feedback, das ihm wertvolle Hinweise für die nächste Iteration liefert. Bekommt der Entwickler aber nur Gegenwind, weil der Rest des Teams ja sowieso alles viel besser weiß, dann wird auch dies zur selbsterfüllenden Prophezeiung werden.

Wenn jedoch alle Beteiligten im Hinterkopf haben, dass wir es gerade in der Software-Entwicklung mit iterativen Vorgängen zu tun haben, wird das zarte Feature-Pflänzchen wachsen und gedeihen und vielleicht sogar ein komplett neues Produkt daraus entstehen. ■

Jens Geyer
entwickelt bei der VSX Vogel Software GmbH Lösungen, die bei Kunden weltweit im Einsatz sind. Sein Fokus liegt auf der effizienten Entwicklung hochwertiger paralleler und verteilter Anwendungen.
http://jensgeyer.net

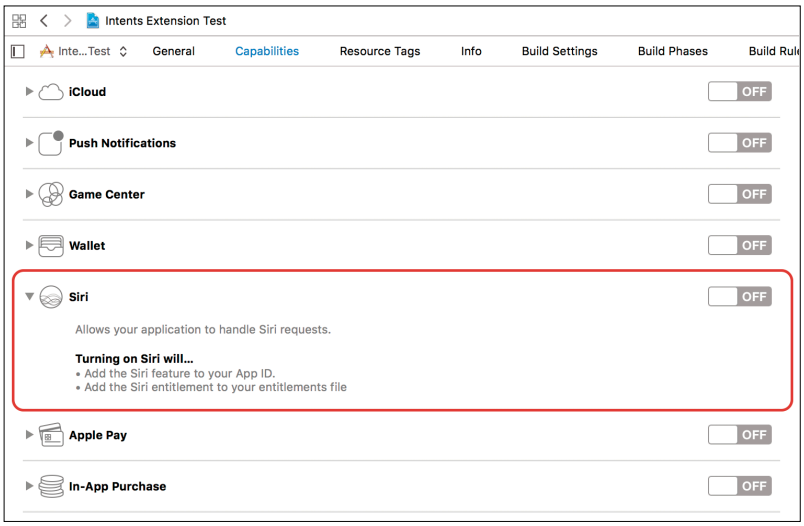
INTEGRATION VON SIRI IN IOS-APPS

Power und Usability

Wie Sie Apples Sprachassistentin in eigenen Apps nutzen.

Apples Sprachassistentin hat sich seit ihrer erstmaligen Vorstellung zusammen mit dem iPhone 4s massiv weiterentwickelt. Nicht nur hat sie in dieser Zeit eine Vielzahl neuer Befehle erlernt sowie neue Plattformen erobert, nein, inzwischen können sich auch App-Entwickler von Dritt-Anwendungen für iOS die Power und Usability von Siri zunutze machen. Dazu hat Apple zusammen mit iOS 10 zwei neue Frameworks eingeführt: Intents sowie Intents UI, beide werden auch gerne unter dem Namen SiriKit zusammengeführt (Bild 1).

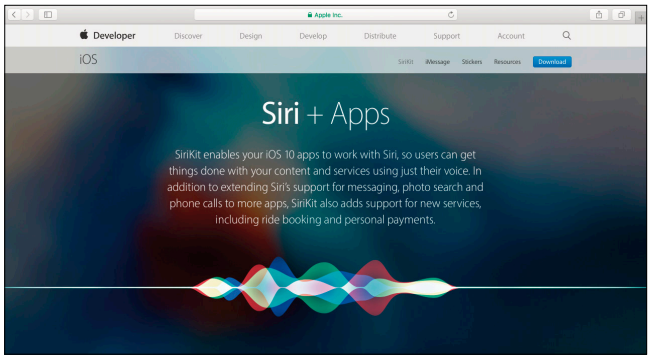
Beide bringen verschiedene Funktionen mit, um Siri bei der Steuerung und Verwendung der eigenen Apps nutzen zu können. Dabei kommt dem Intents Framework die entscheidende Rolle zu: Mit Hilfe dieses Frameworks definieren Sie, welche Aktionen sich im Zusammenspiel mit Siri über Ihre App ausführen lassen, und setzen damit die dafür benötigte Logik um.



Um Siri für iOS-Apps verwenden zu können, muss zunächst einmal die entsprechende Capability aktiviert werden (Bild 2)

Mittels des Intents UI Frameworks können Sie darüber hinaus ein optionales User Interface zur Verfügung stellen, das dem Nutzer das Ergebnis seiner Aktion noch einmal vor Augen führt, bevor die entsprechende Aktion schlussendlich ausgeführt wird. Wie beschrieben, ist ein solches Interface aber optional, alternativ steht ein Standard-UI zur Verfügung, das im Zusammenspiel mit Siri-Anfragen verwendet werden kann und von iOS selbst bereitgestellt wird, ohne dass Sie dafür sonst noch etwas tun müssen.

Bevor wir uns im Detail mit der technischen Umsetzung von Intents Extension sowie Intents UI Extensions auseinandersetzen, möchte ich Sie auf die möglichen Anwendungsbereiche aufmerksam machen, in deren Kontext Siri überhaupt verwendet werden kann. Denn Siri kann mitnichten für alle Aufgaben und Funktionen verwendet werden, die Ihnen womöglich bereits im Kopf herumschweben. Die Integration von Siri in eigene Apps ist auf einige bestimmte Anwendungsbereiche begrenzt, und diese werden direkt von Apple vorgegeben. Es ist zwar davon auszugehen, dass diese Bereiche im Lauf der Zeit und mit kommenden neuen Versionen von iOS stetig erweitert werden, dennoch ist man – wenigstens zum jetzigen Zeitpunkt – voll und ganz an sie gebunden.



Mit SiriKit ist es ab iOS 10 möglich, Apples Sprachassistentin für eigene Apps zu nutzen (Bild 1)

Mit iOS 10 kann Siri in Apps eingesetzt werden, die sich in einem oder mehreren der folgenden Anwendungsbereiche und Aufgabengebiete bewegen:

- Messaging,
- Fotos,
- Zahlungen,
- Workouts,
- Restaurant-Reservierungen,
- Fahrtbuchungen,
- CarPlay,
- VoIP-Telefonie.

Dies sind jene Bereiche, in denen Siri in eigenen Anwendungen verwendet werden kann. Dabei stellt Apple für jeden dieser Bereiche mehrere verschiedene Aktionen (auch als sogenannte Intents bezeichnet) bereit, die vom Nutzer ausgeführt und in den eigenen Apps implementiert werden können. Das ist insofern wichtig, als die Verwendung von Siri – wie beschrieben – auf diese Bereiche und Intents beschränkt

ist; andere Möglichkeiten der Verwendung, Nutzung und Implementierung von Siri stehen schlicht nicht zur Verfügung. Welche Intents dabei genau zur Verfügung gestellt werden und welche Klassen beziehungsweise Protokolle Sie zur Implementierung eines solchen Intents benötigen, erfahren Sie bald. Zunächst betrachten wir aber das grundlegende Erstellen einer neuen Intents Extension beziehungsweise Intents UI Extension sowie das Vorbereiten eines iOS-Projekts auf die Nutzung von Siri.

Vorbereiten eines Xcode-Projekts

Damit eine iOS-App mit Siri interagieren kann, müssen zunächst einige grundlegende Vorkehrungen getroffen werden. Dabei gilt es zunächst, die Siri-Capability für das betreffende iOS-Target zu aktivieren. Durch diesen Schritt wird die Unterstützung von Siri für die zugehörige App ID aktiviert und zugleich ein passendes Entitlements File erstellt und Ihrem Projekt hinzugefügt.

Diese Capability können Sie setzen, indem Sie in Ihrem Projekt das zugehörige iOS-Target auswählen, dort in den Reiter *Capabilities* wechseln und anschließend den Switch im Bereich *Siri* aktivieren (Bild 2).

Auch die *Info.plist*-Datei Ihres iOS-Projekts müssen Sie um ein neues Schlüssel-Wert-Paar ergänzen. Als Schlüssel kommt dabei *NSSiriUsageDescription* zum Einsatz, als Wert wird ein String erwartet. Dieser String soll dem Nutzer erklären, wozu die App Siri verwenden möchte. Da Siri von einer App nur dann genutzt werden kann, wenn der Nutzer explizit der Verwendung zustimmt, ist es wichtig, ihn in diesem String klar und deutlich auf die Nützlichkeit von Siri in der zugrunde liegenden App aufmerksam zu machen (Bild 3).

Mit diesen erfüllten Grundvoraussetzungen muss nun abschließend Ihre App noch offiziell um die Erlaubnis des Nutzers fragen, Siri verwenden zu dürfen. Nur wenn der Nutzer dem zustimmt, kann die Logik Ihrer Intents Extension beziehungsweise Intents UI Extension überhaupt zum Einsatz kommen.

Zu diesem Zweck steht Ihnen die Klasse *INPreferences* des Intents Frameworks zur Verfügung. Diese verfügt über eine Typmethode namens *requestSiriAuthorization(_:)*, deren Deklaration so aussieht:

```
class func requestSiriAuthorization(_ handler:
(INSiriAuthorizationStatus) -> Void)
```

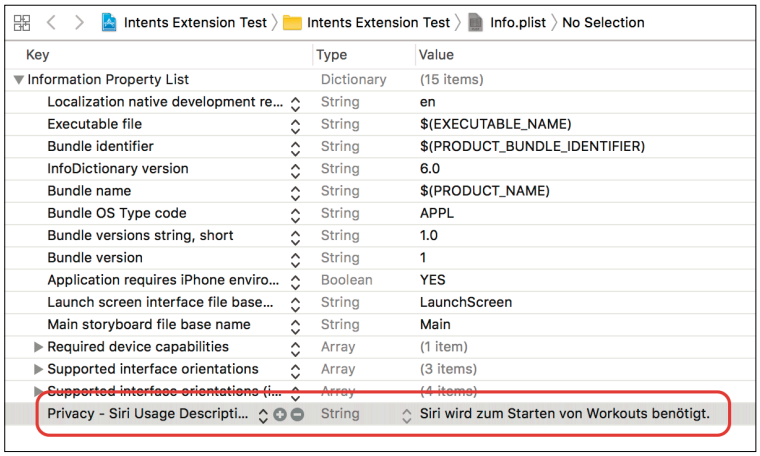
Wenn Sie diese Methode an irgendeiner Stelle innerhalb Ihrer iOS-App aufrufen, wird das System daraufhin eigenständig einen Alert anzeigen, in dem der Nutzer darüber informiert wird, dass die App Zugriff auf die Verwendung von Siri erhalten möchte (dabei ist in diesem Alert auch der String enthalten, den Sie in der *Info.plist*-Datei für den Schlüssel *NSSiriUsageDescription* definiert haben). Dieser Alert wird dabei für jede App nur einmalig angezeigt, weitere Aufrufe der Methode *requestSiriAuthorization(_:)* führen also zu keinen weiteren Alerts für den Nutzer.

Als Ergebnis liefert Ihnen diese Methode ein Closure zurück und übergibt Ihnen den aktuellen Status der Siri-Nutzung in Form eines Parameters vom Typ *INSiriAuthorizationStatus*. Dieser Status gibt Aufschluss darüber, ob eine App für die Verwendung von Siri freigegeben wurde oder nicht. Diese Information kann auch zu jeder Zeit abgefragt werden, indem die Typmethode *siriAuthorizationStatus()* auf die Klasse *INPreferences* aufgerufen wird.

Bei *INSiriAuthorizationStatus* handelt es sich dabei um eine Enumeration, die insgesamt über vier verschiedene Werte verfügt. Diese Werte und und ihre jeweiligen Status werden im Folgenden erläutert:

- *authorized:* Der Nutzer hat der App die Verwendung von Siri gestattet.
- *denied:* Der Nutzer hat der App die Verwendung von Siri nicht gestattet.
- *restricted:* Die App darf Siri nicht verwenden. Im Unterschied zu *denied* kann dieser Status beisipelsweise dann auftreten, wenn Siri auf dem entsprechenden iOS-Gerät gänzlich deaktiviert ist und überhaupt nicht genutzt wird.
- *notDetermined:* Es wurde noch nicht über eine mögliche Verwendung von Siri für die App entschieden.

Erst, wenn Sie den Nutzer mit Hilfe der Typmethode *requestSiriAuthorization(_:)* gefragt haben, ob Ihre App Siri verwenden darf, und der Nutzer dem zugestimmt hat, kann Siri auch tatsächlich genutzt werden. Sind diese Grundlagen erfüllt, können Ihre Intents Extensions beziehungsweise Intents UI Extensions auch ihre Arbeit verrichten.



Mit Hilfe des Schlüssels NSSiriUsageDescription beschreiben Sie, wofür Ihre App den Zugriff auf Siri benötigt (Bild 3)

Erstellen von Intents Extensions und Intents UI Extensions

Wie bereits eingangs erläutert, stehen mit iOS 10 zwei neue Frameworks zur Verfügung, um Siri in eigenen Apps zu nutzen: Intents und Intents UI. Ebenso verhält es sich mit den zugehörigen Extensions, die benötigt werden, um Siri in eigenen Apps zu nutzen. Eine neue Intents Extension ist das Herzstück, möchte man Siri in den eigenen Anwendungen verwenden. Diese Extension nutzt die Funk- ▶

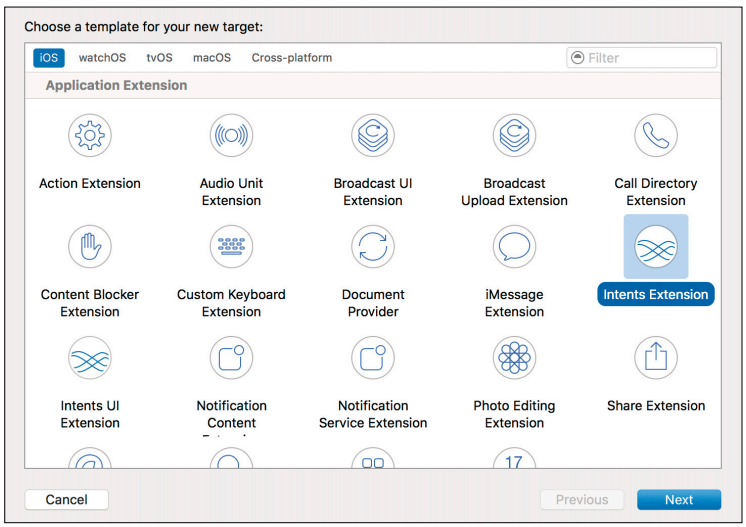
tionen des Intents Frameworks, um darüber die benötigte Logik zu implementieren.

Damit wird auch direkt deutlich, dass die Umsetzung zur Verwendung von Siri ausschließlich in einer solchen Intents Extension stattfindet, nicht im eigentlichen und zugrunde liegenden iOS-Projekt selbst. Möchte man zusätzlich noch die Oberfläche der Sprachsteuerung an die eigene App anpassen, wird darüber hinaus eine zusätzliche Intents UI Extension benötigt, doch dazu später mehr.

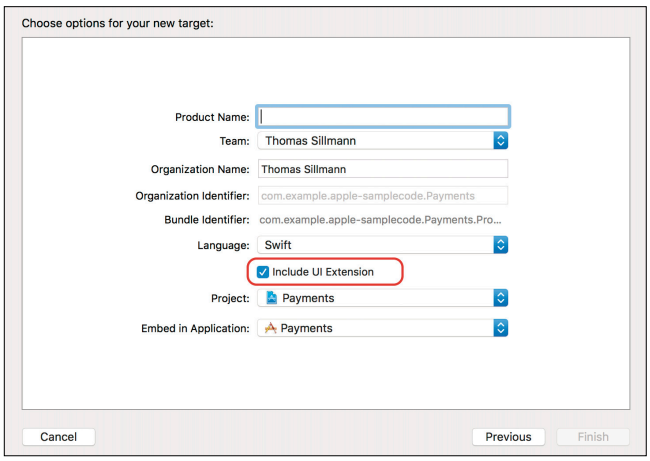
Um ein bestehendes iOS-Projekt um eine Intents Extension oder auch Intents UI Extension zu ergänzen, wählt man zunächst das eigentliche Projekt aus und klickt in der Target-Übersicht auf die Schaltfläche *Add Target…* In der sich öffnenden Template-Übersicht zum Hinzufügen eines neuen Targets muss nun im oberen Bereich die Plattform *iOS* ausgewählt und anschließend in den Bereich *Application Extension* gescrollt werden. Dort stehen dann passende Vorlagen namens *Intents Extension* und *Intents UI Extension* zur Erstellung zur Verfügung (Bild 4).

Nach einem Klick auf *Next* folgt die typische Grundkonfiguration der neuen Extension. Wie bei allen anderen Extensions auch sind die wichtigsten Einstellungen der gewünschte Product Name (wie lautet der Name der Extension im eigenen Projekt?) sowie die Auswahl des Projekts, zu dem die Extension hinzugefügt werden soll (beispielsweise wenn Sie mit einem Workspace arbeiten, der mehrere Projekte enthält) und die Auswahl der iOS-App in Form des entsprechenden Targets, für das die Extension verwendet werden soll. Bei Erstellung einer neuen Intents Extension können Sie darüber hinaus auf Wunsch den Haken bei der Checkbox *Include UI Extension* setzen, um parallel zur neuen Intents Extension auch gleich eine zusätzliche Intents UI Extension zu erstellen (Bild 5). In diesem Fall wird für die Intents UI Extension derselbe Name verwendet wie für die Intents Extension und ihr lediglich zusätzlich das Suffix *UI* angehängt.

Nach einem Klick auf Finish wird sodann die neue Extension erstellt und Ihrem Projekt hinzugefügt. Eine neue Intents



Eine Intents Extension ist die Grundlage zur Nutzung von Siri in eigenen Apps (Bild 4)



Sie können parallel zu einer neuen Intents Extension auch direkt eine zusätzliche Intents UI Extension erstellen (Bild 5)

Extension setzt sich dabei aus zwei Dateien zusammen: einer automatisch von Xcode erzeugten Klasse *IntentHandler* sowie einer *Info.plist*-Datei. Eine Intents UI Extension dagegen verfügt über eine *IntentViewController*-Klasse, ein Storyboard sowie ebenfalls eine *Info.plist*-Datei (Bild 6).

IntentHandler-Klasse

Betrachten wir nun zunächst einmal die von Xcode automatisch erzeugte Klasse *IntentHandler* einer neuen Intents Extension. Diese ist abgeleitet von der Klasse *INExtension*. Diese Klasse erbt direkt von *NSObject* und bringt selbst keine speziellen Funktionen mit. Sie dient als Deklaration des Einstiegspunkt Ihrer Intents Extension und ist gleichzeitig konform zum *NSIntentHandlerProviding*-Protokoll. Dieses Protokoll dient dazu, alle Funktionen zu deklarieren, die benötigt werden, um eine korrekte Funktionsweise der Siri-Unterstützung zu gewährleisten. Zum jetzigen Zeitpunkt stellt dieses Protokoll genau eine Methode bereit, die zugleich als *Required* gekennzeichnet ist und daher innerhalb der Klasse *IntentHandler* zwingend implementiert werden muss: *handler(for:)*. Die vollständige Deklaration der Methode sieht so aus:

```
func handler(for intent: INIntent) -> Any?
```

Diese Methode übergibt einen Intent in Form eines *INIntent*-Objekts. Dieser Intent entspricht der Aufgabe, die der Nutzer mittels Siri angefragt hat und die nun von Ihrer App ausgeführt werden soll. Zu diesem Zweck liefern Sie innerhalb dieser Methode ein beliebiges Objekt zurück, das sich um die Verarbeitung des gewünschten Intents kümmert.

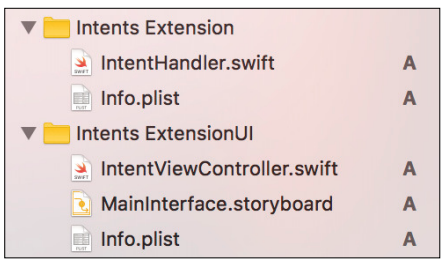
Bevor wir uns tiefergehend mit der Implementierung von Siri-Anfragen innerhalb einer Intents Extension beschäftigen, muss zuvor eine andere zentrale Frage geklärt werden: Welche Intents gibt es überhaupt, die in eigenen Apps verarbeitet werden? Und wie definiert man innerhalb einer Intents Extension, welche dieser Intents unterstützt werden?

Wie bereits zu Beginn dieses Artikels angeschnitten, kann Siri nur im Zusammenspiel mit bestimmten Bereichen und Aufgabengebieten genutzt und verwendet werden. Für jeden dieser Bereiche stehen mehrere verschiedene sogenannte Intents bereit, die einer konkreten Aufgabe entsprechen, die für diesen Bereich mit Hilfe von Siri umgesetzt werden kann.

Für alle diese Intents stehen im Intents Framework jeweils spezialisierte Klassen und Protokolle bereit. Über diese wird geregelt, um was für einen Intent es sich handelt und welche Informationen darüber übertragen und welche Aktionen ausgeführt werden können. Dabei unterteilen sich diese Elemente in folgende drei Bereiche:

- **Intent:** Ein Intent wird vom System mittels Siri erzeugt und an die Extension übergeben. Ein Intent enthält alle Informationen für eine spezifische Anfrage, die mittels Siri zusammengesetzt wurde. Sie kann beispielsweise beim Versenden einer Textnachricht den Empfänger sowie die eigentliche Nachricht beinhalten. Jeder Intent wird über eine eigene separate Subklasse von *INIntent* abgebildet.
- **Handler:** Für jeden Intent gibt es ein zugehöriges Handler-Protokoll. Dieses Protokoll dient dazu, den Intent zu prüfen und womöglich weiter zu optimieren. Wenn der Nutzer beispielsweise eine Nachricht versenden möchte, in seinem Intent aber kein Empfänger angegeben ist, kann der Handler auf dieses Fehlen eines Empfängers aufmerksam machen und dem System melden, dass es diese Information benötigt.
- **Response:** Ein Response ist eine Art Antwort auf die verschiedenen Funktionen des Handler und gibt an, ob ein Intent alle benötigten Informationen enthält oder nicht. Für jeden Intent existiert eine eigene individuelle Subklasse von *INIntentResponse*.

Der Handler macht dabei in dieser Aufstellung ein weiteres wichtiges Merkmal bei der Verwendung von Siri in eigenen Apps deutlich: Man selbst hat nie direkten Zugriff auf die Konversation, die der Nutzer mit Siri führt. Siri verarbeitet die Eingaben zu einem Intent und reicht diesen an die Extension weiter. Diese prüft den Intent und gibt eine passende Response zurück. Diese Response wiederum wird vom System selbst verarbeitet, und daraufhin wird selbsttätig die Kommunikation mit dem Nutzer weitergeführt, ohne dass die Extension selbst konkret darauf Einfluss nehmen kann. Bei der Verwendung von Siri ist die eigene Extension also eher als Dirigent zu sehen; ein direkter Zugriff auf die Sprachassistentin oder gar auf Konversationen ist nicht möglich.



Xcode erzeugt zusammen mit einer neuen Intents Extension und Intents UI Extension einige Standarddateien (Bild 6)

Wie beschrieben, gibt es für für jeden Intent exakt eine passende INIntent-Subklasse, ein passendes Handler-Protokoll und eine passende INIntentResponse-Subklasse. Tabelle 1 führt für jeden der zu Beginn genannten unterstützten Bereiche deren verfügbare Intents sowie die zugehörigen Klassen und das jeweilige Handler-Protokoll auf.

Unterstützte Intents registrieren

Jeder Intent, der von einer Intents Extension unterstützt wird, muss in der *Info.plist*-Datei der Extension in Form des Namens der jeweiligen INIntent-Subklasse hinzugefügt werden. Für diesen Zweck finden sich zwei passende Schlüssel unterhalb der *NSExtension*- sowie *NSExtensionAttributes*-Schlüssel namens *IntentsSupported* und *IntentsRestrictedWhileLocked* (Bild 7).

IntentsSupported ist dabei der wichtigere der beiden Schlüssel. Ihm wird ein String-Array als Wert zugewiesen, wobei dieses Array den Namen jeder INIntent-Subklasse enthält, deren Intent in der Extension unterstützt werden soll. Innerhalb von *IntentsRestrictedWhileLocked* werden all jene Intents aufgeführt, deren Funktion nur dann ausgeführt werden soll, wenn der Nutzer sein iOS-Gerät entsperrt hat. Auch hier wird für jeden entsprechenden Intent der zugehörige Name der passenden INIntent-Subklasse angegeben.

▼ NSExtension	Dictionary	(3 items)
▼ NSExtensionAttributes	Dictionary	(2 items)
▼ IntentsRestrictedWhileLocked	Array	(0 items)
▼ IntentsSupported	Array	(3 items)
Item 0	String	INSendMessageIntent
Item 1	String	INSearchForMessagesIntent
Item 2	String	INSetMessageAttributeIntent
NSExtensionPointIdentifier	String	com.apple.intents-service
NSExtensionPrincipalClass	String	$(PRODUCT_MODULE_NAME).IntentHandler

Die Info.plist-Datei einer Intents Extension enthält passende Schlüssel zur Angabe der unterstützten Intents (Bild 7)

Somit gilt es, diese beiden Schlüssel mit den gewünschten Intents zu füllen, bevor diese mit der entsprechenden Intents Extension auch genutzt werden können.

Resolve, Confirm, Handle

In den bisherigen Abschnitten wurden die Grundlagen und essenziellen Vorbereitungen zum Verwenden von Siri für eigene iOS-Apps vorgestellt und erläutert. Nun geht es an den eigentlichen Kern der Umsetzung innerhalb der Intents Extension. Die von Xcode automatisch generierte *IntentHandler*-Klasse wurde dabei bereits kurz vorgestellt. Ihre Aufgabe liegt in der Implementierung des *INIntentHandlerProviding*-Protokolls, das die Methode *handler(for:)* definiert.

Diese Methode entspricht dem Startpunkt Ihrer Siri-Implementierung und Sie nutzen typischerweise den übergebenen *intent*-Parameter vom Typ *INIntent*, um darüber auf die INIntent-Subklasse zu schließen, zu der die gestellte Anfrage gehört. Denn die Methode *handler(for:)* wird genau dann ▶

aufgerufen, wenn der Nutzer einen Befehl über Siri abgesetzt hat, der für Ihre App bestimmt ist. Und sofern Sie nicht für all Ihre Intents ein- und dieselbe Funktion ausführen, müssen Sie zunächst einmal mittels Type Casting prüfen, um welchen Intent es sich genau handelt.

Anschließend geben Sie ein passendes Objekt zurück, das konform zu dem zum Intent passenden Handler-Protokoll ist. Dieses Objekt kümmert sich dann um die weitere Kommunikation und Auswertung der eigentlichen Anfrage.

Im folgenden Listing sehen Sie ein Beispiel dazu, das eine Intents Extension für Workouts darstellen soll. So wird der Intent dort gegen die verschiedenen verfügbaren INIntent-Subklassen für Workouts geprüft und je nach Intent ein Objekt einer zugehörigen Handler-Klasse zurückgegeben. Wie beschrieben, müssen diese Klassen konform zum jeweils zugehörigen Handler-Protokoll des jeweiligen Intents sein:

```
class IntentHandler: INExtension {
  override func handler(for intent: INIntent) -> Any? {
    if intent is INStartWorkoutIntent {
      return StartWorkoutIntentHandler()
    } else if intent is INPauseWorkoutIntent {
      return PauseWorkoutIntentHandler()
    } else if intent is INResumeWorkoutIntent {
      return ResumeWorkoutIntentHandler()
    } else if intent is INEndWorkoutIntent {
```

Tabelle 1: Verfügbare Intents

INIntent-Subklasse	INIntentResponse-Subklasse	Handler-Protokoll
INSendMessage-Intent	INSendMessage-IntentResponse	INSendMessageIntent-Handling
INSearchFor-MessagesIntent	INSearchForMessa-gesIntentResponse	INSearchForMessage-sIntentHandling
INSetMessage-AttributeIntent	INSetMessageAttri-buteIntentResponse	INSetMessageAttribute-IntentHandling
INSearchForPho-tosIntent	INSearchForPhotos-IntentRespsone	INSearchForPhotos-IntentHandling
INStartPhoto-PlaybackIntent	INStartPhotoPlay-backIntentResponse	INStartPhotoPlay-backIntentHandling
INSendPay-mentIntent	INSendPayment-IntentResponse	INSendPaymentIntent-Handling
INRequestPay-mentIntent	INRequestPayment-IntentResponse	INRequestPayment-IntentHandling
INStartWorkout-Intent	INStartWorkout-IntentResponse	INStartWorkoutIntent-Handling
INPauseWork-outIntent	INPauseWorkout-IntentResponse	INPauseWorkoutIntent-Handling
INResumeWork-outIntent	INResumeWorkout-IntentResponse	INResumeWorkout-IntentHandling
INEndWork-outIntent	INEndWorkout-IntentResponse	INEndWorkoutIntent-Handling
INCancelWork-outIntent	INCancelWorkout-IntentResponse	INCancelWorkout-IntentHandling
INGetUserCur-rentRestaurant-Reservation-BookingsIntent	INGetUserCurrentRes-taurantReservation-BookingsIntent-Response	INGetUserCurrent-RestaurantReservation-BookingsIntent Handling
INGetRestau-rantGuestIntent	INGetRestau-rantGuestIntent Response	INGetRestaurant-GuestIntentHandling
INGetAvailable-Restaurant-Reservation-Booking-DefaultsIntent	INGetAvailableRes-taurantReservation-BookingDefaults-IntentResponse	INGetAvailableRestau-rantReservationBoo-kingDefaultsIntent-Handling
INGetAvailable-Restaurant-Reservation-BookingsIntent	INGetAvailableRes-taurantReservation-BookingsIntentRes-ponse	INGetAvailableRestau-rantReservationBoo-kingsIntentHandling
INBookRestau-rantReserva-tionIntent	INBookRestaurant-ReservationIntent-Response	INBookRestaurant-ReservationIntent-Handling
INListRide-OptionsIntent	INListRideOptions-IntentResponse	INListRideOptions-IntentHandling
INRequest-RideIntent	INRequestRide-IntentResponse	INRequestRideIntent-Handling
INGetRideStatus-Intent	INGetRideStatus-IntentResponse	INGetRideStatusIntent-Handling
INSetAudio-SourceInCarIntent	INSetAudioSourceIn-CarIntentResponse	INSetAudioSource-InCarIntentHandling
INSetClimate-SettingsInCar-Intent	INSetClimateSet-tingsInCarIntent-Response	INSetClimateSettings-InCarIntentHandling
INSetDefroster-SettingsInCar-Intent	INSetDefrosterSet-tingsInCarIntent-Response	INSetDefrosterSettings-InCarIntentHandling
INSaveProfile-InCarIntent	INSaveProfileInCar-IntentResponse	INSaveProfileInCar-IntentHandling
INSetProfile-InCarIntent	INSetProfileInCar-IntentResponse	INSetProfileInCarIntent-Handling
INSetHeatSet-tingsInCarIntent	INSetHeatSettings-InCarIntentResponse	INSetHeatSettings-CarIntentHandling
INSetRadio-StationIntent	INSetRadioStation-IntentResponse	INSetRadioStation-IntentHandling
INStartAudio-CallIntent	INStartAudioCall-IntentResponse	INStartAudioCallIntent-Handling
INStartVideo-CallIntent	INStartVideoCall-IntentResponse	INStartVideoCallIntent-Handling
INSearchCall-HistoryIntent	INSearchCallHistory-IntentResponse	INSearchCallHistory-IntentHandling

```
        return EndWorkoutIntentHandler()
      } else if intent is INCancelWorkoutIntent {
        return CancelWorkoutIntentHandler()
      }
      return nil
    }
}
```

Die jeweils zurückgegebenen Objekte innerhalb der Methode *handler(for:)* sind nun für die weitere Bearbeitung der vom Nutzer getätigten Anfrage verantwortlich, wobei das jeweilige Handler-Protokoll eine zentrale Rolle spielt. Jedes Handler-Protokoll eines Intents definiert verschiedene Methoden, um eine Anfrage auszuwerten und zu verarbeiten. Diese Methoden unterteilen sich dabei in jedem Fall bei jedem Handler-Protokoll in eine von drei Phasen:

- Resolve,
- Confirm,
- Handle.

Jede dieser Phasen kann über spezifische Methoden des jeweiligen Handler-Protokolls abgebildet werden (konkrete Beispiele dazu sehen wir gleich). Im Folgenden stelle ich Ihnen alle diese drei Phasen im Detail vor und erläutere, wie Sie diese Phase mit Hilfe Ihres Handler-Objekts abbilden können.

Resolve-Phase

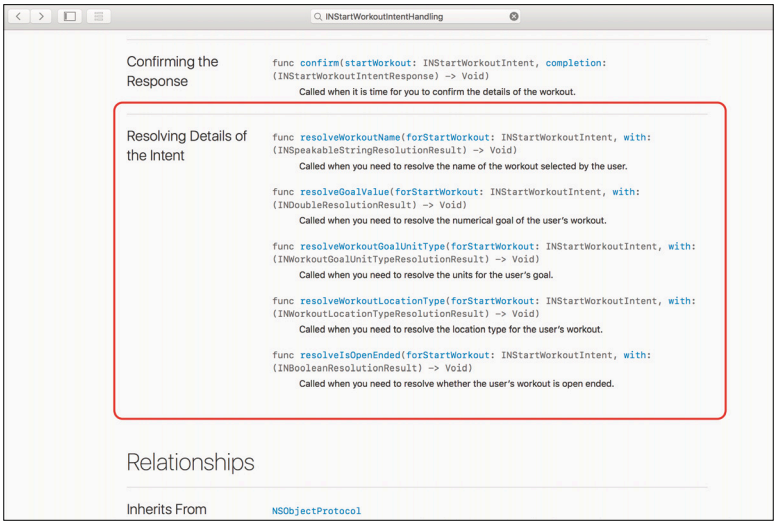
In der Resolve-Phase wird die genaue Aufgabe definiert, die der Nutzer mittels Siri für eine App durchführen möchte. Diese Phase dient dazu, alle notwendigen und optionalen Fragen zu klären, um am Ende über eine vollständige Anfrage zu verfügen, die von der App wie gewünscht bearbeitet werden kann.

Nehmen wir beispielsweise an, eine Messaging-App möchte Siri nutzen, um Nachrichten zu versenden. Gibt der Nutzer nun über Siri lediglich an, wie die Nachricht lautet, so würde diese Information alleine noch nicht ausreichen, um erfolgreich eine Nachricht verschicken zu können, da der zugehörige Empfänger noch fehlt. Genau diese Details werden in der Resolve-Phase gelöst.

Dabei stehen im zugehörigen Handler-Protokoll des zugrunde liegenden Intents verschiedene solcher Resolve-Methoden bereit, die alle die einzelnen Aspekte des Intents überprüfen und abfragen. Gleichzeitig wird Ihnen in diesen Methoden das aktuelle Intent-Objekt übergeben. Ihre Aufgabe besteht also darin, innerhalb der jeweiligen Resolve-Methode die aktuelle Konfiguration des Intent-Objekts zu prüfen und anschließend ein Feedback zu geben, ob die erhaltene Information korrekt, oder aber fehlerhaft, unvollständig oder in anderer Weise unbrauchbar ist.

Betrachten wir einmal die Resolve-Phase anhand eines konkreten Beispiels auf Basis eines Intents zum Starten eines Workouts (also auf Basis der Klasse *INStartWorkoutIntent*). Das zugehörige Handler-Protokoll *INStartWorkoutIntentHandling* definiert insgesamt fünf verschiedene optionale Methoden für die Resolve-Phase zum Starten eines Workouts (Bild 8).

Jede dieser fünf Methoden dient dabei dazu, genau eine spezifische Information für diese spezielle Anfrage zu überprüfen und ein Feedback darüber zu geben, ob die aktuell im Intent hinterlegte Information so in Ordnung ist oder nicht. Beispielsweise ist da die Methode *resolveWorkoutName(forStartWorkout:with:)*. Diese dient dazu, den Namen des zu startenden Workouts zu überprüfen. Gibt es das vom Nutzer genannte Workout überhaupt in der App, und falls nicht, hat er womöglich ein anderes gemeint? Um dieses Problem zu lösen, übergibt die Methode als ersten Parameter das zugehörige Intent-Objekt (in diesem Fall ist es vom Typ *INStartWorkoutIntent)*. Über dessen Attribute können Sie nun innerhalb der Methode feststellen, ob Ihre App etwas mit dem Namen des Workouts anfangen kann oder nicht. So verfügt die Klasse *INStartWorkoutIntent* über eine passende Property namens *workoutName*, die genau diese Information enthält. Äquivalent dazu verhält es sich auch bei der Resolve-Methode *resolveIsOpenEnded(forStartWorkout:with:)*. Über diese können Sie prüfen, ob der Nutzer das Workout offen (also ohne konkretes Endziel) durchführen möchte oder nicht. Auch hier erhalten Sie das Intent-Objekt als Parameter und können so im Fall des Startens eines Workouts die Property *isOpenEnded* der Klasse *INStartWorkoutIntent* nutzen, um die aktuelle Information zu dieser Option auszulesen.



Die Handler-Protokolle eines Intents bringen jeweils verschiedene Methoden für die Resolve-Phase einer Anfrage mit (Bild 8)

Umgekehrt bedeutet das auch, dass Siri die Anfrage eines Nutzers direkt in ein zugehöriges Intent-Objekt verpackt und dieses über die verschiedenen Methoden der Handler-Protokolle übergibt. Die Arbeit zur Bearbeitung einer Anfrage mittels Siri beginnt also für uns App-Entwickler frühestens in der vorgestellten Resolve-Phase, wobei wir immer bereits konfigurierte Intent-Objekte enthalten. Unsere Aufgabe besteht dann darin, mit Hilfe der verschiedenen Resolve-Metho- ▶

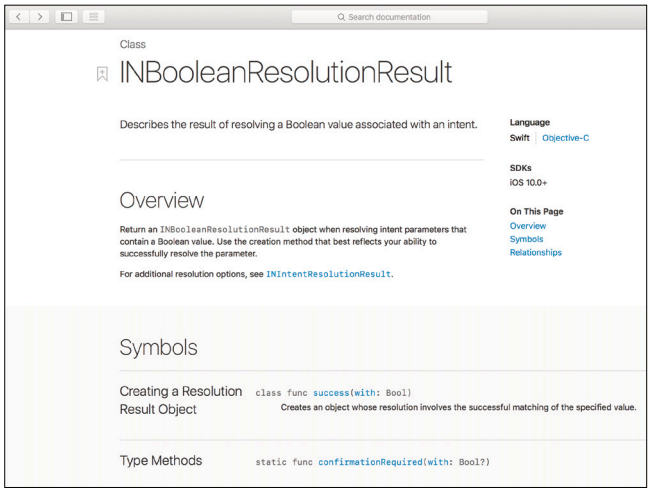
den diesen Intent zu prüfen und Feedback zu geben, ob Informationen fehlen oder unvollständig beziehungsweise vollständig sind.

Feedback geben in der Resolve-Phase

Dazu muss nun noch eine zentrale Frage geklärt werden: Wie gibt man dem System Feedback darüber, ob die im Intent übergebenen Informationen in Ordnung oder unvollständig sind? Zu diesem Zweck steht innerhalb des Intents Frameworks die Klasse *INIntentResolutionResult* bereit. Jede Resolve-Methode der Handler-Protokolle übergibt neben dem Intent-Objekt als zweiten Parameter ein Closure, das innerhalb der Methode aufgerufen werden muss und wiederum als Parameter ein Objekt dieser Klasse *INIntentResolutionResult* oder einer ihrer Subklassen erwartet. Darüber teilen Sie dann dem System mit, ob Siri noch einmal beim Nutzer bezüglich einer bestimmten Information nachfragen muss oder ob alles in Ordnung ist.

Betrachten wir dazu als Beispiel einmal die Resolve-Methode *resolveIsOpendEnded(forStartWorkout:with:)* des *INStartWorkoutIntentHandling*-Protokolls. Das Closure erwartet hier als Parameter ein Objekt der Klasse *INBooleanResolutionResult* (wie beschrieben handelt es sich dabei um eine Subklasse von *INIntentResolutionResult)*. Je nachdem, wie Sie die Instanz konfigurieren, die Sie über dieses Closure zurückliefern, steuert Siri die weitere Kommunikation mit dem Nutzer. Dazu bringen sowohl die Klasse *INIntentResolutionResult* als auch Ihre Subklassen verschiedene Typmethoden mit, über die Sie direkt neue Instanzen dieser Klassen mit dem gewünschten Ergebnis erstellen.

Beispielsweise erzeugen Sie mit der Typmethode *needsValue()* ein Ergebnisobjekt, das Siri mitteilt, dass für das abgefragte Element der entsprechenden *Resolve*-Methode noch der gewünschte Wert fehlt und notwendig ist, während mit *notRequired()* darauf hingewiesen wird, dass für dieses Element zur Bearbeitung der Anfrage sowieso kein Wert benötigt wird. Welche Ergebnisse Sie über die jeweilige Ergebnisklasse zurückliefern können, entnehmen Sie direkt der Dokumentation von Apple (Bild 9).



Die Dokumentation von Xcode gibt Aufschluss darüber, welche Ergebnisse eine entsprechende Ergebnisklasse für eine Resolve-Methode zurückgeben kann (Bild 9)

Im folgenden Listing sehen Sie einmal ein Beispiel für eine Implementierung der Resolve-Methode *resolveWorkoutName(forStartWorkout:with:)* des *INStartWorkoutIntentHandling*-Protokolls:

```
func resolveWorkoutName(forStartWorkout intent:
INStartWorkoutIntent, with completion: @escaping
(INSpeakableStringResolutionResult) -> Void) {
  guard let workoutName =
  intent.workoutName?.spokenPhrase else {
    return completion
    (INSpeakableStringResolutionResult.needsValue())
  }
  if workoutName == "Joggen" {
    let workoutSpeakableString = INSpeakableString
    (identifier: "Joggen", spokenPhrase: "Joggen",
    pronunciationHint: nil)
    return completion
    (INSpeakableStringResolutionResult.success
    (with: workoutSpeakableString))
  }
  return completion
  (INSpeakableStringResolutionResult.unsupported())
}
```

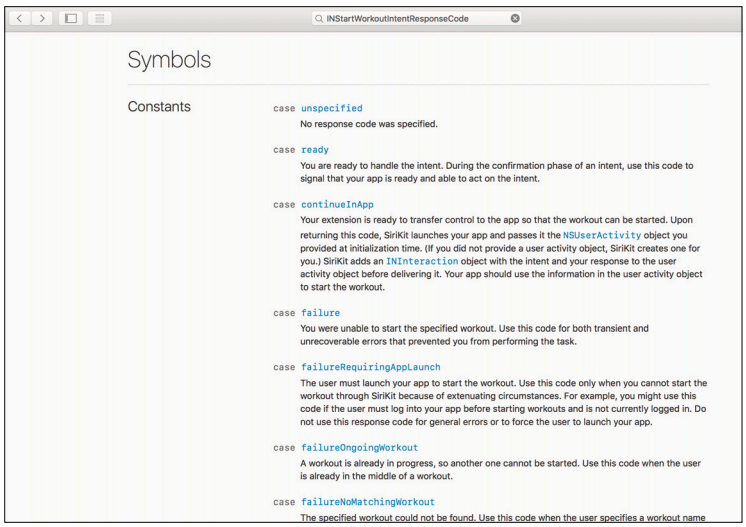
Darin wird zunächst geprüft, ob für den Namen des Workouts überhaupt ein Wert existiert. Falls nicht, wird dieser mittels *needsValue()* angefordert.

Der Einfachheit halber wird der Wert anschließend gegen ein einziges Workout namens *Joggen* geprüft (wobei hier natürlich weitere aufgeführt werden könnten). Entspricht das übergebene Workout diesem Namen, so wird eine Erfolgsmeldung zurückgegeben. Trägt das Workout einen anderen Namen (und kann es somit nicht an dieser Stelle ausgewertet werden), geben wir diese Information mit Hilfe von *unsupported()* zurück, womit dem Nutzer mitgeteilt wird, dass sein genannter Workout-Name nicht mit der App funktioniert und er einen anderen vergeben muss.

Allen Methoden der *Resolve*-Phase in allen Handler-Protokollen ist gemein, dass diese immer optional sind und daher nicht implementiert werden müssen. In zwei Fällen sollten Sie aber wenigstens einen Teil von ihnen in Ihrer Extension integrieren.

Einerseits sind die Resolve-Methoden dann wichtig, wenn für die korrekte Funktionalität einer Anfrage bestimmte Parameter zwingend benötigt werden (wie beispielsweise der Name des Workouts, das gestartet werden soll), und andererseits ist es ebenfalls sinnvoll, Methoden zu implementieren, deren Wert optional ist oder gar nicht gebraucht wird und das direkt mit einem entsprechenden Ergebnisobjekt dem System mitzuteilen.

Im letzteren Fall wird Siri dann nämlich gar nicht erst nach dieser Information fragen, womit die Nutzung Ihrer Intents Extension mit Siri für den Nutzer deutlich komfortabler und schneller vonstatten gehen kann.



Subklassen von *INIntentResponse* verfügen über jeweils unterschiedliche und spezifische Informationen, um über den Status einer Nutzeranfrage zu informieren (Bild 10)

Sobald alle implementierten Methoden der Resolve-Phase erfolgreich durchlaufen wurden und ein positives Ergebnis zurückliefern, beginnt die Confirm-Phase. Diese ist pro Handler-Protokoll mit einer einzigen Methode deklariert, die darüber hinaus optional ist.

Die Confirm-Phase dient dazu, dem Nutzer noch einmal die Aktion vor Augen zu führen, die ausgeführt wird, wenn er die Confirm-Phase entsprechend bestätigt. Es ist die letzte Chance für den Nutzer, den Vorgang noch abzubrechen, bevor die eigentliche Aktion über die Intents Extension in der nächsten Phase ausgeführt wird.

Für einzelne Vorgänge ruft das System automatisch von sich aus solch einen Confirm-Dialog auf, selbst wenn Sie keine zugehörige Confirm-Phase in Ihrer Extension implementiert haben (das ist beispielsweise bei Bezahlvorgängen der Fall). Davon abgesehen empfiehlt Apple ohnehin, die zugehörige Methode der Confirm-Phase des Handler-Protokolls immer zu implementieren.

Jede *Confirm*-Methode eines Handler-Protokolls übergibt zwei Parameter: Zum einen ist da wieder der zugrunde liegende Intent als Objekt der zugehörigen INIntent-Subklasse, zum anderen ein Closure, das am Ende der Methode aufgerufen werden muss und als Parameter ein Ergebnisobjekt erwartet, bei dessen Typ es sich um die zum Intent passende Subklasse von *INIntentResponse* handelt. Jede dieser Subklassen verfügt über einen eigenen Initializer, um ein Objekt dieser Klasse zu erstellen und ein Feedback über den Status der Anfrage zu geben.

Für den Start eines Workouts können Sie so beispielsweise angeben, dass es losgehen kann, dass das genannte Workout nicht gefunden wurde, dass womöglich bereits ein anderes Workout läuft und daher kein neues gestartet werden kann, dass der Nutzer die Anfrage in der iOS-App direkt fortsetzen muss oder dass ein sonstiger Fehler aufgetreten ist.

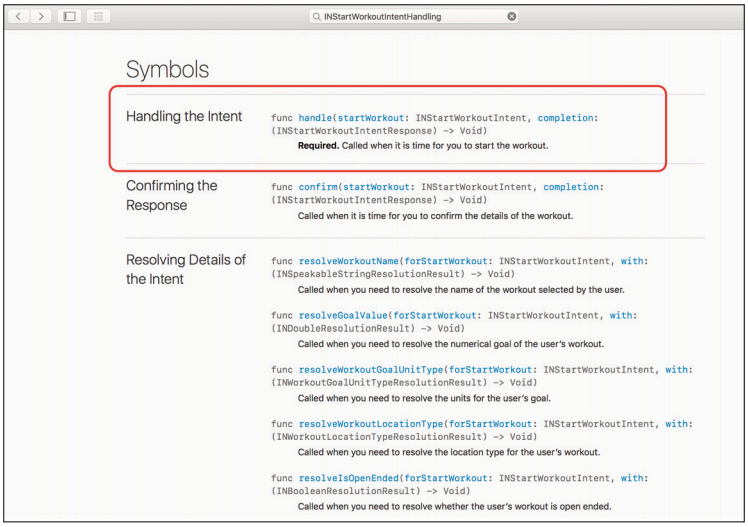
Welche Response-Optionen dabei pro Intent genau zur Verfügung stehen, entnehmen Sie der zugehörigen Dokumentation für die jeweilige INIntentResponse-Subklasse (Bild 10).

Im folgenden Listing sehen Sie ein einfaches Beispiel für die Implementierung einer Confirm-Phase für den Intent zum Starten eines Workouts. Dabei wird in jedem Fall als Response zurückgegeben, dass der Start des Workouts bereit ist:

```
func confirm(startWorkout intent:
INStartWorkoutIntent, completion: @escaping
(INStartWorkoutIntentResponse) -> Void) {
  return completion(INStartWorkoutIntentResponse
  (code: .ready, userActivity: nil))
}
```

Nach erfolgreichem Abschluss der Confirm-Phase kommt es schlussendlich zur Handle-Phase. In dieser Phase wird die vom Nutzer gesprochene und gewünschte Aktion nun endgültig durchgeführt, und das ist auch genau die Stelle, an der Ihre Extension nun alles Notwendige tut, um die zur Aktion passende Logik in Ihrer App umzusetzen (Workout starten, Workout beenden, Überweisung tätigen und so weiter).

Genau wie in der Confirm-Phase erhalten Sie dabei über die zugehörige Handle-Methode einerseits Zugriff auf das zugrunde liegende Intent-Objekt als auch auf ein Closure, das Sie am Ende der Methode in jedem Fall aufrufen und dabei ein Objekt des zugehörigen Response-Objekts übergeben müssen. Es handelt sich dabei um ein Objekt derselben INIntentResponse-Subklasse wie in der Confirm-Phase auch. Gleichzeitig ist diese Handle-Methode die einzige Methode eines jeden Handler-Protokolls, die als *required* gekennzeichnet ist und somit in jedem Fall in ihrem Handler-Objekt implementiert werden muss (Bild 11). Das folgende Listing zeigt ein einfaches Beispiel zur Implementierung der ▶

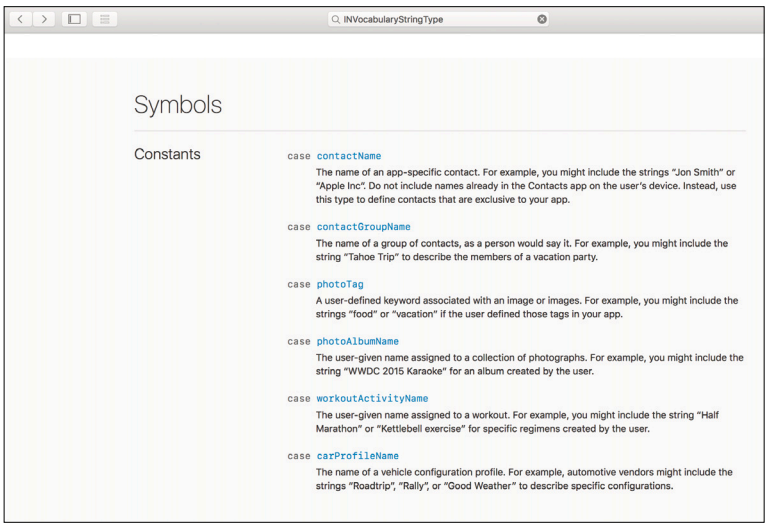


Die Handle-Methode ist die einzige Methode eines jeden Handler-Protokolls, die nicht optional und somit Pflicht ist (Bild 11)

Handle-Methode zum Starten eines Workouts. Dabei wird als Response die Information zurückgegeben, dass der Vorgang innerhalb der zugehörigen iOS-App fortgesetzt werden soll:

```
func handle(startWorkout intent:
INStartWorkoutIntent, completion: @escaping
(INStartWorkoutIntentResponse) -> Void) {
  return completion(INStartWorkoutIntent
  Response
  (code: .continueInApp, userActivity: nil))
}
```

Auch hier gilt, dass jeder Typ für die verschiedenen Response-Objekte unterschiedliche Antwortmöglichkeiten in Form sogenannter Codes bereitstellt. Wie bereits im Abschnitt zur Confirm-Phase beschrieben, gibt dabei die Dokumentation von Xcode Aufschluss darüber, welche dieser Codes es pro Response-Typ gibt und welche Funktion diese erfüllen. Gemein ist dabei allen ein optionales User-Activity-Objekt vom Typ *NSUserActivity*, das dazu genutzt werden kann, weitere Informationen an die zugrunde liegende iOS-App weiterzureichen, damit diese beispielsweise beim Start aus Siri heraus alle nötigen Informationen besitzt, um zu wissen, was zu tun ist.



Die Enumeration INVocabularyStringType definiert für alle verfügbaren Vokabular-Kategorien passende Werte (Bild 12)

App-Vokabular definieren

In manchen Anwendungsfällen ist es sinnvoll, eigenes App-spezifisches Vokabular zu definieren, das der Nutzer dann direkt bei der Kommunikation mit Siri zum Starten einer Aktion für eine App verwenden kann. Dabei kann es sich beispielsweise um die Namen verschiedener Workouts in einer Workout-App handeln, oder um die verfügbaren Kontakte in einer Messaging-App.

Damit Siri ein solches App-spezifisches Vokabular versteht, muss es ihr beigebracht werden. Dafür gibt es zwei Möglichkeiten, die davon abhängig sind, ob das Vokabular App-basierend ist (für jeden Nutzer also gleichermaßen gilt, wie eben die genannten Namen von verfügbaren Workouts einer Workout-App), oder ob es User-basierend ist (also nur pro spezifischem Nutzer gilt, wie beispielsweise bei Kontaktnamen in einer Nachrichten-App).

Für letzteren Fall spielt die Klasse *INVocabulary* aus dem Intents Framework eine zentrale Rolle. Diese Klasse ist als Singleton designt, dessen Instanz über die Typmethode *shared()* abgerufen werden kann. Mit Hilfe der Methode *setVocabularyStrings(_:of:)* erlaubt diese Klasse das Hinzufügen von nutzerspezifischen Begriffen zu Siris Wortschatz.

Dabei gilt es zu beachten, dass dieser Wortschatz nur dann über diese Methode erweitert werden kann, wenn die entsprechenden Begriffe zu einer der folgenden Kategorien gehören:

- Kontaktnamen,
- Kontaktgruppen,
- Tags für Fotos,
- Namen von Fotoalben,
- Namen von Workouts,
- Profilnamen von Fahrzeugen (für CarPlay).

Ist das der Fall und gibt es nutzerspezifischen Wortschatz in wenigstens einem dieser Bereiche, kann dieser Wortschatz Siri über die Klasse *INVocabulary* übermittelt werden. Wie so etwas aussehen kann, zeigt das folgende Listing:

```
let workoutVocabulary = NSOrderedSet(array:
["Joggen", "Bahnenschwimmen", "Walken"])
let sharedVocabulary = INVocabulary.shared()
sharedVocabulary.setVocabularyStrings
(workoutVocabulary, of: .workoutActivityName)
```

Dort wird das Singleton der Klasse *INVocabulary* dazu genutzt, spezifische, von einem Nutzer definierte Workout-Namen über die Methode *setVocabularyStrings(_:of:)* bekannt zu machen. Der erste Parameter entspricht dabei einem *NSOrderedSet* der einzelnen Begriffe, während über den zweiten die Kategorie definiert wird, zu denen die zuvor genannten Begriffe gehören. Diese Kategorien werden dabei über Werte der Enumeration *INVocabularyStringType* definiert, in der sich für jede der zuvor genannten Kategorien ein passender Wert findet. Für die Namen von Workouts lautet dieser *workoutActivityName* (Bild 12).

Neben diesem nutzerspezifischen Vokabular ist auch – wie bereits beschrieben – das Setzen eines App-spezifischen Vokabulars möglich. Zu diesem Zweck wird eine passende PLIST-Datei innerhalb des iOS-Projekts erstellt und dort hinzugefügt, die den Namen *AppIntentVocabulary* tragen muss. Dieser PLIST-Datei müssen auf oberster Ebene zwei Schlüssel vom Typ *Array* zugewiesen werden:

- ParameterVocabularies,
- IntentPhrases.

Letzterer Schlüssel *IntentPhrases* ist dabei optional und kann somit auch weggelassen werden. Er dient dazu, Beispiele zur

Nutzung von App-spezifischem Vokabular in einer kompletten Siri-Abfrage darzustellen, damit Siri so womöglich zugehörige Nutzeranfragen besser verstehen kann. Dazu setzt sich ein Element in diesem *IntentPhrases*-Array immer aus einem Dictionary zusammen, das zwei Schlüssel besitzt: Der Schlüssel *IntentName* verweist auf den Namen der INIntent-Subklasse, für dessen Intent ein Beispielsatz für Siri angegeben werden soll, während der zweite Schlüssel *IntentExamples* einem Array entspricht, in dem ein oder mehrere Strings Beispielaussagen zur Nutzung des App-spezifischen Vokabulars mit Siri enthalten (beispielsweise etwas wie *Siri, starte ein Joggen-Workout in MyWorkoutApp*, um das App-spezifische Workout *Joggen* der MyWorkoutApp zu starten).

Wie beschrieben, sind diese Beispiele des *IntentPhrases*-Schlüssels aber optional und dienen lediglich zur Optimierung der Verwendung von Siri.

Essenziell hingegen ist die Verwendung des ersten Schlüssels *ParameterVocabularies*, über den das eigentliche App-spezifische Vokabular definiert wird. Jedes Element dieses Arrays ist ein Dictionary mit zwei Schlüsseln: *ParameterNames* und *ParameterVocabulary*.

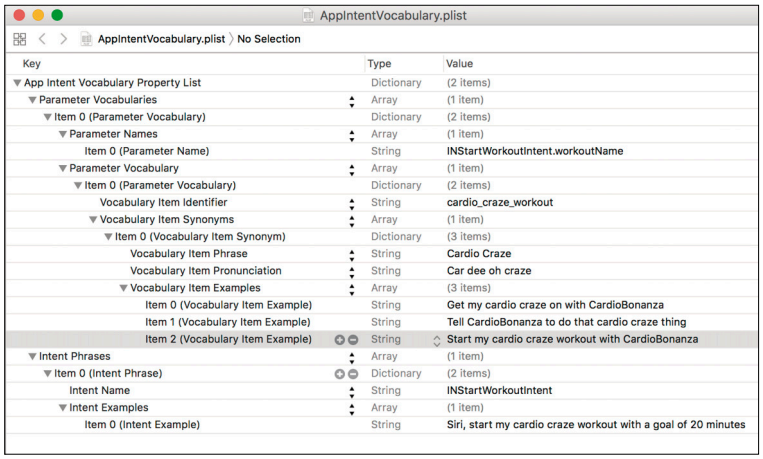
Bei *ParameterNames* handelt es sich um ein Array, in dem all jene Intents inklusive deren zugehöriger Property aufgeführt werden, zu denen Sie App-spezifisches Vokabular hinterlegen möchten. Dabei ist zu beachten, dass Sie ein solches App-spezifisches Vokabular nur für die folgenden Intents und Properties setzen können:

- INRequestRideIntent.rideOptionName,
- INStartWorkoutIntent.workoutName,
- INPauseWorkoutIntent.workoutName,
- INResumeWorkoutIntent.workoutName,
- INEndWorkoutIntent.workoutName,
- INCancelWorkoutIntent.workoutName.

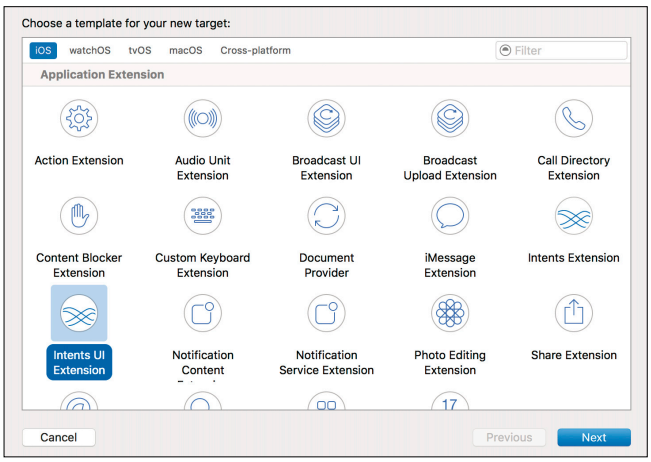
Der zweite Schlüssel *ParameterVocabulary* beschreibt wiederum ein Array, das sich ebenfalls aus ein oder mehreren Dictionaries zusammensetzt. Der erste Schlüssel eines jeden solchen Elements lautet *VocabularyItemIdentifier* und ist ein von Ihnen festgelegter eindeutiger Bezeichner für das entsprechende Vokabular. Siri wird diesen an Ihre App weiterreichen, wenn es die entsprechende Phrase in einer Nutzeranfrage entdeckt. Bei dem zweiten Schlüssel namens *VocabularyItemSynonyms* handelt es sich um ein weiteres Array von Dictionaries. Jedes Element eines solchen Dictionarys verfügt dabei wenigstens über einen Schlüssel namens *VocabularyItemPhrase*.

Dieser erwartet als Wert einen String, der dem gewünschten Vokabular in seiner Originalschreibweise entspricht. Optional kann über den Schlüssel *VocabularyItemPronunciation* ein weiterer String angegeben werden, der denselben Begriff in Lautschrift darstellt, damit Siri ihn besser verstehen kann. Ebenfalls optional ist ein dritter Schlüssel namens *VocabularyItemExamples*, der ein Array von Strings als Wert erwartet, wobei jeder String ein gesprochenes Beispiel der Verwendung eben jenes spezifischen Vokabulars enthält.

Ein vollständiges Beispiel einer solchen *AppIntentVocabulary.plist*-Datei von Apple sehen Sie in Bild 13.



Ein Beispiel einer *AppIntentVocabulary.plist*-Datei (Bild 13)



Eine Intents UI Extension lässt sich einfach über die entsprechende Target-Vorlage in Xcode erstellen (Bild 14)

Intents UI Extension

Zum Abschluss dieses Artikels betrachten wir nun noch einmal die UI-Seite der Integration von Siri in eigenen iOS-Apps. Bis hierhin haben wir alle Facetten und Techniken kennengelernt, die uns das Intents Framework zu Verfügung stellt, um eigene Aktionen mit Hilfe von Siri in iOS-Apps auszuführen und dabei auch eigenes App-spezifisches oder User-spezifisches Vokabular zu verwenden. Dabei wird bei der Kommunikation zwischen Siri und dem Nutzer in der Regel immer das Standardinterface von iOS angezeigt, ohne dass wir Entwickler groß Einfluss darauf nehmen können.

Die einzige Möglichkeit für uns, mittels eines eigenen angepassten User Interfaces einzugreifen, ist nach Ende der Resolve-Phase (wenn also die grundlegenden Parameter eines Intents geklärt sind). Genau hier setzt die sogenannte Intents UI Extension an, die immer nur im Zusammenspiel mit einer Intents Extension Sinn ergibt, da sie für sich alleine genommen nicht funktioniert. ▶

Eine neue Intents UI Extension kann entweder direkt im Zusammenspiel mit einer Intents Extension oder separat erstellt werden (Bild 14). Diese setzt sich sodann aus drei Dateien zusammen:

- *IntentViewController*-Klasse,
- *MainInterface*-Storyboard,
- *Info.plist*-Datei.

Dabei gilt es – genau wie bei der Intents Extension auch – die von der Intents UI Extension unterstützten Intents innerhalb der *Info.plist*-Datei einzutragen. Dazu steht unterhalb der Schlüssel *NSExtension* und *NSExtensionAttributes* ein weiterer Schlüssel namens *IntentsSupported* bereit, der sich aus einem Array von Strings zusammensetzt. Für jeden Intent, den die Intents UI Extension unterstützen soll, muss diesem Array der zugehörige Name der jeweiligen INIntent-Subklasse hinzugefügt werden (Bild 15).

Oberfläche des View-Controllers

Die restliche Arbeit findet innerhalb der Klasse *IntentViewController* sowie des zugehörigen Storyboards statt. Die Oberfläche des View-Controllers kann dabei nach Belieben über das Storyboard gestaltet werden, während die zugehörige Logik in der gekoppelten *IntentViewController*-Klasse Platz findet.

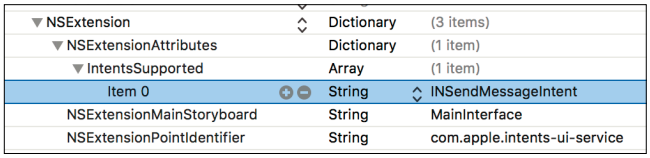
Es handelt sich dabei um eine ganz klassische Subklasse von *UIViewController*, die lediglich zusätzlich konform zum *INUIHostedViewControlling*-Protokoll ist. Dieses Protokoll definiert die als required gekennzeichnete Methode *configure(with:context:completion:)*, deren vollständige Deklaration im folgenden Listing aufgeführt ist:

```
func configure(with interaction: INInteraction!,
context: INUIHostedViewContext, completion: ((CGSize) ->
Void)!)
```

Diese Methode wird dann innerhalb des View-Controllers gefeuert, sobald dieser angezeigt wird und somit ein Intent vom Nutzer abgesetzt wurde. Dabei übergibt diese Methode in ihrem ersten Parameter *interaction* vom Typ *INInteraction* alle wichtigen und benötigten Informationen zum Intent, darunter den eigentlichen Intent selbst in Form der Property *intent*. Diese Informationen können dazu genutzt werden, die Inhalte und das Aussehen der eigenen Oberfläche entsprechend anzupassen. Der zweite Parameter *context* vom Typ *INUIHostedViewContext* gibt Aufschluss darüber, in welchem Zusammenhang der Intent über Siri abgesetzt wurde. Aktuell stehen über diese Enumeration zwei Werte zur Verfügung:

- *siriSnippet:* die Standardanzeige von Siri,
- *mapsCard:* Anzeige von Siri aus der Karten-App heraus.

Der letzte Parameter *completion* stellt ein Closure dar, das am Ende der Methode auf jeden Fall aufgerufen werden muss. Als Parameter erwartet es eine Größe in Form eines *CGSize*-Objekts, über das Sie definieren, auf welcher Fläche Ihre View angezeigt werden soll. Damit können Sie den benötigten Platz für Ihre Ansicht entweder dynamisch verkleinern oder vergrößern.



Auch bei der Intents UI Extension müssen die unterstützten Intents innerhalb der zugehörigen *Info.plist*-Datei hinterlegt werden (Bild 15)

Fazit

Siris Debüt in eigenen iOS-Apps wirkt stimmig und durchdacht. Apple hat zwar klare Szenarien definiert, die ausschließlich im Zusammenspiel mit Siri in iOS-Apps verwendet werden dürfen und können, andererseits macht das aber auch die Implementierung übersichtlich und stimmig. Das Konzept, mittels dreier Phasen (Resolve, Confirm, Handle) Nutzeranfragen zu beantworten und dabei ein bestmögliches Ergebnis zu liefern, weiß zu überzeugen und lässt sich einfach auf verschiedenste Intents übertragen, ohne bei jedem neuen Intent immer wieder prüfen zu müssen, wie dieser nun korrekterweise implementiert wird.

Man darf davon ausgehen, dass Apple den Siri-Support in kommenden iOS-Versionen ausbauen und schrittweise um weitere Intents ergänzen wird. Der grundlegende Aufbau in Form der Frameworks Intents Extension und Intents UI Extension wird aber wahrscheinlich so bestehen bleiben. Wer also schon immer einmal Funktionen einer iOS-App mit Siri ansprechen wollte, hat nun mit iOS 10 endlich die Chance dazu. Fällt die App in eine der unterstützten Kategorien, kann es direkt an die entsprechende Implementierung gehen. ■

Links zum Thema

- SiriKit Programming Guide
  https://developer.apple.com/library/prerelease/content/documentation/Intents/Conceptual/SiriIntegrationGuide
- Referenz zum Intents Framework
  https://developer.apple.com/reference/intents
- Referenz zum IntentsUI Framework
  https://developer.apple.com/reference/intentsui

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

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, Christian Bleske, Olena Bochkor, Jens Geyer, Thomas Hafen, Tam Hanna, Bernhard Lauer, Ralf Lieser, Florence Maurice, Frank Pientka, Michael Rohrlich, Markus Schraudolph, Marco Schulz, Katharina Sckommodau, Thomas Sillmann, Frank Simon, Andreas Sommer

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

Mediaberatung und Content-Marketing-Lösungen
Thomas Wingenfeld
Telefon: (089) 741 17-124
E-Mail: sales@nmg.de

Klaus Ahlering
Telefon: (089) 741 17-125
E-Mail: sales@nmg.de

Stellenanzeigen / Anbieterverzeichnisse / Guides
Juliane Roschke
Telefon: (089) 741 17-283
E-Mail: sales@nmg.de

Disposition
Marita Brotz
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

JUGENDSCHUTZ

Bedingt zulässig

Manche Webangebote benötigen einen Jugendschutzbeauftragten.

Der Jugendschutz ist nachvollziehbarerweise ein hohes Gut in der deutschen Rechtsordnung. Dies manifestiert sich in zahlreichen Einzelvorschriften, wie etwa in den Paragrafen 104 und 106 des Bürgerlichen Gesetzbuches (BGB) zur nicht vorhandenen beziehungsweise lediglich eingeschränkten Geschäftsfähigkeit von Minderjährigen.

Es gibt aber auch zwei Gesetze, die ausschließlich dem Schutz Jugendlicher dienen, nämlich das Jugendschutzgesetz (JuSchG) und der Staatsvertrag über den Schutz der Menschenwürde und den Jugendschutz in Rundfunk und Telemedien (Jugendmedienschutz-Staatsvertrag, kurz: JMStV). Durch diese Regelungen sollen Personen bis 14 Jahre und Personen zwischen 14 und 18 Jahren vor schädlichen Einflüssen auf ihre Entwicklung und Erziehung bewahrt werden.

Dieser Schutz zielt nicht nur, aber besonders auch auf elektronische Informations- und Kommunikationsmedien, in erster Linie also auf das Internet ab. Die Funktion eines Jugendschutzbeauftragten stellt in gewisser Weise ein Bindeglied zwischen den Betreibern von entsprechenden Internetangeboten und deren Nutzern dar.

Betroffene Webangebote

Da nicht jede Website das Erfordernis zur Bestellung eines Jugendschutzbeauftragten mit sich bringt, stellt sich die Frage, um welche Angebote es im Kern geht. Dazu liefert der JMStV klare Antworten. Es sind drei Kategorien zu unterscheiden, nämlich absolut unzulässige Angebote, bedingt zulässige Angebote sowie entwicklungsbeeinträchtigende Angebote.

Unter die Kategorie der absolut unzulässigen Angebote (§ 4 JMStV) fallen Websites, wenn sie

- Propagandamittel im Sinne des Strafgesetzbuches darstellen, deren Inhalt gegen die freiheitliche demokratische Grundordnung oder den Gedanken der Völkerverständigung gerichtet ist,
- Kennzeichen verfassungswidriger Organisationen verwenden,

Praxis-Tipp

Ein Jugendschutzbeauftragter fungiert insbesondere als Ansprechpartner für die Nutzer sowie als Berater des Anbieters. Er soll bei Fragen der Herstellung, des Erwerbs, der Planung und der Gestaltung von Angeboten Entscheidungshilfen geben. Er ist daher bei allen Entscheidungen zur Wahrung des Jugendschutzes angemessen und rechtzeitig zu beteiligen und über das jeweilige Angebot vollständig zu informieren.

Die Bundesprüfstelle für jugendgefährdende Medien (BPjM) ist zuständig für die Indizierung von Träger- und Telemedien mit jugendgefährdendem Inhalt (Bild 1)

- zum Hass gegen Teile der Bevölkerung oder gegen eine nationale, rassische, religiöse oder durch ihr Volkstum bestimmte Gruppe aufstacheln, zu Gewalt- oder Willkürmaßnahmen gegen sie auffordern oder die Menschenwürde anderer dadurch angreifen, dass Teile der Bevölkerung oder eine vorbezeichnete Gruppe beschimpft, böswillig verächtlich gemacht oder verleumdet werden,
- eine unter der Herrschaft des Nationalsozialismus begangene Handlung in einer Weise, die geeignet ist, den öffentlichen Frieden zu stören, leugnen oder verharmlosen,
- grausame und sonst unmenschliche Gewalttätigkeiten gegen Menschen in einer Art schildern, die eine Verherrlichung oder Verharmlosung solcher Gewalttätigkeiten ausdrückt oder die das Grausame oder Unmenschliche des Vorgangs in einer die Menschenwürde verletzenden Weise darstellt; dies gilt auch bei virtuellen Darstellungen,
- als Anleitung zu einer rechtswidrigen Tat dienen,
- den Krieg verherrlichen,
- gegen die Menschenwürde verstoßen,
- Kinder oder Jugendliche in unnatürlich geschlechtsbetonter Körperhaltung darstellen,
- pornografisch sind und Gewalttätigkeiten, den sexuellen Missbrauch von Kindern oder Jugendlichen oder sexuelle Handlungen von Menschen mit Tieren zum Gegenstand haben.

Angebote sind hingegen als bedingt zulässig einzustufen, wenn sie

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

- in sonstiger Weise pornografisch sind,
- in den Teilen A und C der Liste nach § 18 JuSchG aufgenommen sind oder mit einem in diese Liste aufgenommenen Werk ganz oder im Wesentlichen inhaltsgleich sind,
- offensichtlich geeignet sind, die Entwicklung von Kindern und Jugendlichen oder ihre Erziehung zu einer eigenverantwortlichen und gemeinschaftsfähigen Persönlichkeit unter Berücksichtigung der besonderen Wirkungsform des Verbreitungsmediums schwer zu gefährden.

Online-Angebote der genannten Art sind allerdings nur dann zulässig, wenn von Seiten des Anbieters sichergestellt ist, dass sie nur Erwachsenen zugänglich gemacht werden, also eine geschlossene Benutzergruppe existiert. Zur Einrichtung einer solchen bedarf es eines hinreichenden Altersverifikationssystems (AVS).

Der erwähnte § 18 JuSchG enthält übrigens die Regelung zur Führung einer Liste mit jugendgefährdenden Medien durch die zuständige Bundesprüfstelle (Bild 1).

Umsetzung in der Praxis

Bei solchen Angeboten, die gemäß JuSchG für Kinder oder Jugendliche der jeweiligen Altersstufe nicht freigegeben sind, wird die Eignung zur Beeinträchtigung der Entwicklung vermutet. Wenn also beispielsweise eine Video-DVD mit der Einstufung »Freigegeben ab 18 Jahren« gekennzeichnet ist, so wird vermutet, dass ihre Verbreitung an Kinder und Jugendliche unter 18 Jahren entwicklungsbeeinträchtigend sein wird. Der Gesetzgeber macht daher sehr strenge Vorgaben an die Online-Verbreitung von solchen Inhalten. Um diese Pflichten zu erfüllen, muss der betreffende Anbieter insbesondere

- durch technische oder sonstige Mittel sicherstellen, dass die Wahrnehmung des Angebots durch Kinder oder Jugendliche der betroffenen Altersstufe unmöglich oder wesentlich erschwert wird, oder
- die Zeit, in der die Angebote verbreitet oder zugänglich gemacht werden, so wählen, dass Kinder oder Jugendliche der betroffenen Altersstufe üblicherweise die Angebote nicht wahrnehmen.

Die Zeitspanne für eine Verbreitung liegt zwischen 23 und 6 Uhr beziehungsweise bei Angeboten mit einer entwicklungsbeeinträchtigenden Wirkung auf Kinder oder Jugendliche unter 16 Jahren zwischen 22 und 6 Uhr. Bei Filmen, die unter 12 Jahren nicht freigegeben sind, ist bei der Wahl der Sendezeit dem Wohl jüngerer Kinder Rechnung zu tragen.

Ist bei bestimmten Angeboten eine entwicklungsbeeinträchtigende Wirkung nur auf Kinder (Personen unter 14 Jahren) zu befürchten, kann der Online-Anbieter seine Verpflichtung erfüllen, wenn das Angebot getrennt von für Kinder bestimmten Angeboten verbreitet wird oder abrufbar ist.

Bestellung des Beauftragten

Alle geschäftsmäßigen Anbieter von den erwähnten Inhalten trifft grundsätzlich die Pflicht, einen Jugendschutzbeauftragten zu bestellen, sofern die Inhalte allgemein zugänglich sind. Das ist dann der Fall, wenn sie von einem unbestimmten Personenkreis abgerufen werden können. Auch eine geschlossene Benutzergruppen kann unter diesen Begriff fallen, wenn die Teilnahme an dieser von durch jedermann erfüllbaren Kriterien abhängig gemacht wird, wie etwa der Zahlung eines Mitgliedsbeitrags.

Geschäftsmäßig handelt ein Anbieter, der beabsichtigt, Angebote in gleicher Art und Weise wiederholt anzubieten, sie also zu einem dauerhaften Bestandteil seiner Tätigkeit gezählt werden können. Es kommt insoweit nicht auf eine Gewinnerzielungsabsicht oder einen tatsächlichen Gewinn an.

Anbieter mit insgesamt weniger als 50 Mitarbeitern oder nachweislich weniger als 10 Millionen Zugriffen (nicht Nutzer) im Monatsdurchschnitt eines Jahres können auf die Bestellung verzichten, wenn sie sich einer Einrichtung der Freiwilligen Selbstkontrolle, wie zum Beispiel der FSM, anschließen und diese zur Wahrnehmung der Aufgaben des Jugendschutzbeauftragten verpflichten.

Ein Jugendschutzbeauftragter muss generell die erforderliche Fachkunde besitzen. Damit ist ein gewisses Mindestmaß an juristischer Fachkenntnis im Bereich des Jugendschutzrechts gemeint. Es kann sich um einen Angestellten des jeweiligen Unternehmens, aber auch um einen externen Dienstleister, wie etwa einen Rechtsanwalt, handeln. Der Inhaber oder auch ein Geschäftsführer des Unternehmens scheidet für diese Funktion hingegen aufgrund von potenziellen Interessenskonflikten aus.

Beim Verstoß gegen eine bestehende Bestellungspflicht kann ein Ordnungsgeld von bis zu 500.000 Euro verhängt werden. Das Gleiche gilt für den Fall, dass der Jugendschutzbeauftragte keine ausreichende Fachkunde besitzt oder er sein Amt nicht ernsthaft wahrnimmt. ■

Michael Rohrlich
ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.
www.ra-rohrlich.de

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Marktstudie

Es fehlt an Personal und Know-how

Freiberufler erwarten für sich selbst bei der Digitalisierung in erster Linie Chancen. Für die Unternehmen schätzen sie die Lage allerdings weit kritischer ein. Das zeigt die aktuelle Marktstudie »Digitalisierung in Deutschland aus Sicht der IT-Freiberufler« von Solcom. Die Umfrage hat ergeben, dass die befragten IT-Freiberufler im Zuge der immer weiter fortschreitenden Digitalisierung den Trends Big Data, Mobility und Cloud Computing die größten Chancen einräumen. Gleichzeitig sieht eine große Mehrheit von über 80 Prozent diese Entwicklung als Chance für die deutsche Wirtschaft. Ähnlich verhält es sich bei der Einschätzung der persönlichen Auswirkungen: Acht von zehn erwarten darin Vorteile.

Für die Lage der Unternehmen fällt die Einschätzung weit weniger positiv aus. Nicht einmal jeder Zwanzigste der befragten Freiberufler sieht die Unternehmen reif für die Digitalisierung, mehr als die Hälfte hält sie für nur in geringem Maße vorbereitet. Zur Bewertung, dass Unternehmen den Trend der Digitalisierung nicht erkannt haben, passt auch die Einschätzung der Umfrageteilnehmer, dass die Realisierungsgeschwindigkeit von Digitalisierungsprojekten hinterherhinkt. Als größtes Hindernis bei der Digitalisierung der deutschen Wirtschaft nennen die befragten Freiberufler mit 86,2 Prozent in erster Linie das fehlende Personal beziehungsweise Know-how. Auf die Frage nach den größten Trends setzten zwei Drittel der Umfrageteilnehmer Big Data ganz nach oben. Zehn Prozentpunkte dahinter liegt das Thema Mobility, dicht gefolgt von den Themen Cloud Computing und Internet der Dinge (Bild 1). Autonomes Fahren, Industrie 4.0 sowie semantische beziehungsweise kognitive Technologien werden unter Freiberuflern nicht ganz so hoch gehandelt.

Tabelle 1 zeigt, wie häufig ausgewählte Technologien in Stellenanzeigen genannt werden. Ganz oben rangiert hier das Cloud Computing, gefolgt von MySQL und HTML5. Erst dann folgt in dieser Auswertung der in der Datenbank von Jobkralle.de gespeicherten Stellenanzeigen der Begriff Big Data.

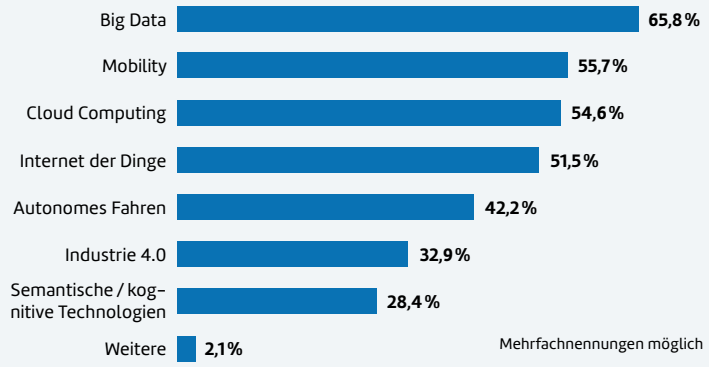
Die allmonatliche Erhebung der Jobangebote für Webentwickler sowie Entwickler von Apps für Mobilgeräte zeigte eine weiterhin sehr hohe Nachfrage, die sich aber auf wenige Bundesländer konzentriert. Zwei Drittel der Job-Ausschreibungen suchen Mitarbeiter für Bayern, NRW, Baden-Württemberg und Hessen (Bild 2). In der Städte-Auswertung kommt Berlin mit 740 Treffern bei der Suche auf Jobkralle.de auf den zweiten Rang hinter München (923 Treffer) und vor Frankfurt am Main (589 Treffer). Die weiteren Plätze belegen Hamburg, Köln und Stuttgart.

Tabelle 1: Technologien

Rang	Technologie	Anteil *
1	Cloud	19,7 %
2	MySQL	10,8 %
3	HTML5	9,9 %
4	Big Data	9,2 %
5	SharePoint	7,3 %
6	Android	6,9 %
7	iOS	6,7 %
8	Microsoft SQL Server	5,3 %
9	CSS3	4,8 %
10	AngularJS	4,2 %
11	Windows 10	3,9 %
12	Responsive Web	2,8 %
13	NoSQL	2,5 %
14	WPF	2,5 %
15	Azure	2,3 %
16	WCF	1,2 %

* Prozentualer Anteil der Treffer

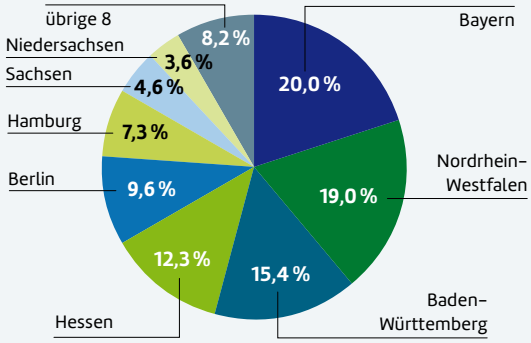
Die größten Trends



Big Data liegt in der Liste der größten Trends ganz vorne. Autonomes Fahren und Industrie 4.0 werden nicht so hoch gehandelt (Bild 1)

web & mobile developer 12/2016 Quelle: Solcom Marktstudie

Jobs für Webentwickler



Zwei Drittel der Webentwickler-Jobs gibt es in Bayern, NRW, Baden-Württemberg und Hessen (Bild 2)

web & mobile developer 12/2016
Quelle: Eigene Erhebungen, jobkralle.de

Zahl des Monats

In Deutschland gibt es **709.033 Entwickler**, sagt die weltweit größte Programmierplattform StackOverflow. Die Daten basieren auf dem Verhalten der Nutzer dieser Plattform – ausgewertet mit Big-Data- und Machine-Learning-Algorithmen. Quelle: StackOverflow

Software Engineering

Uni-Projekt EVELIN

An der Hochschule Coburg arbeiten Pädagoginnen mit Informatikern im Verbundprojekt EVELIN gemeinsam mit fünf weiteren bayerischen Hochschulen daran, das Studium im Bereich Software-Entwicklung zielgerichtet und kompetenzorientiert weiterzuentwickeln. Die Studierenden trainieren überfachliche Fähigkeiten wie Gesprächsführung und Teamarbeit – und zwar nicht im Labor, sondern im fachlichen Kontext. Informatik-Professor Dieter Landes hat gemeinsam mit der Pädagogin Dr. Yvonne Sedelmaier Software-Unternehmen befragt, was sie von ihren zukünftigen Mitarbeitern erwarten. Die Antworten waren eindeutig: Neben der fachlichen Qualifikation spielen soziale und kommunikative Kompetenzen eine wichtige Rolle. Landes und Sedelmaier möchten ihre Studierenden darauf vorbereiten und entwickeln die Hochschullehre im Software Engineering weiter. EVELIN steht für Experimentelle Verbesserung des Lernens von Software Engineering. Etwa 30 Wissenschaftlerinnen und Wissenschaftler der Hochschulen Coburg, Aschaffenburg, Kempten, Landshut, Neu-Ulm und Regensburg sind an dem Projekt beteiligt.

www.hs-coburg.de

ifo

Homeoffice weiter im Trend

39 Prozent der Unternehmen bieten ihren Beschäftigten die Möglichkeit, gelegentlich zu Hause zu arbeiten. Das ergibt sich aus der aktuellen ifo-Randstad-Personalleiter-Befragung. Vor allem größere Unternehmen mit mehr als 500 Beschäftigten bieten das Büro zu Hause an. Hier liegt der Anteil bei 65 Prozent. Bei Firmen mit weniger als 50 Mitarbeitern sind es dagegen nur 29 Prozent. In allen Größenklassen wurde das Angebot aber in den vergangenen Jahren ausgebaut. Und 23 Prozent der Unternehmen wollen Homeoffice in den kommenden Jahren verstärkt nutzen. Die Heimarbeit bedeutet aber keine völlige zeitliche Freiheit, sie ist in 72 Prozent der Unternehmen mit häufiger Anwesenheit im Büro verbunden. In 43 Prozent der Unternehmen (Mehrfachnennungen möglich) sind längere Präsenzphasen erforderlich. Nur in 26 Prozent der Fälle gibt es keinerlei Anwesenheit in der Firma (Bild 3).

In den Unternehmen, die kein Homeoffice anbieten, antworteten 63 Prozent der Personaler, die Anwesenheit der Mitarbeiter in der Firma sei zwingend erforderlich. Dort, wo das nicht der Fall war, begründeten 41 Prozent der Personalleiter ihre Ablehnung mit erschwerter Kommunikation bei Homeoffice, 31 Prozent mit der IT-Sicherheit und 25 Prozent mit dem Datenschutz.

www.ifo.de

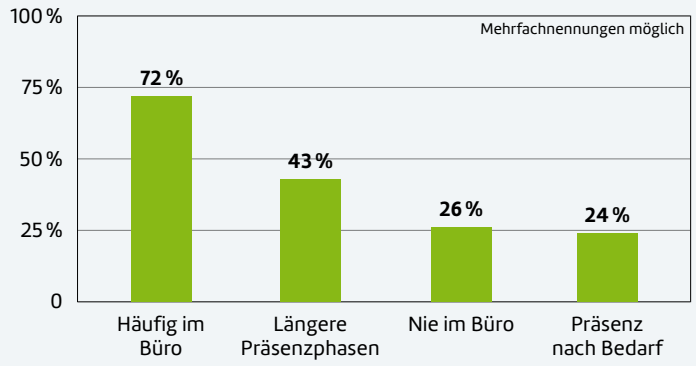
StackOverflow

Berlin: Hauptstadt des Start-up-Booms

In Berlin arbeiten laut StackOverflow derzeit 81.868 Software-Entwickler, darunter 4994 Mobile-Entwickler (Bild 4). Besonders interessant: Im vergangenen Jahr wurde in Berlin alle 20 Minuten ein Start-up gegründet, so besagt es das Global Startup Ecosystem Ranking. Damit wies die Bundeshauptstadt den größten Wachstumsfaktor aller 20 betrachteten Start-up-Ökosysteme auf. In den vergangenen fünf Jahren haben Start-ups an der Spree Tausende Jobs geschaffen. Firmen wie Rocket Internet, Zalando, Zanox oder Wooga sind hier zu nennen, aber auch Auctionata, Delivery Hero, Home24 oder ResearchGate. Zusätzlich eröffnen viele international agierende Unternehmen strategische Zweigstellen in Berlin: Amazon, Ebay, Microsoft, Twitter, Uber, Visa oder Zendesk.

http://stackoverflow.com

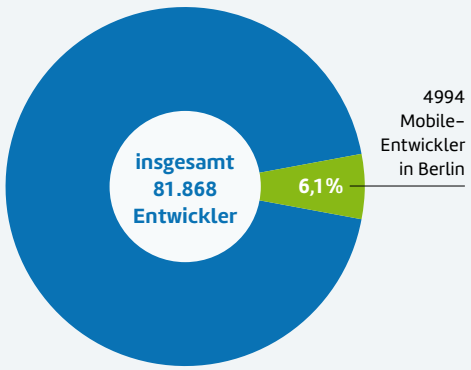
Homeoffice



Nur ein Viertel der im Homeoffice tätigen Mitarbeiter muss gar nicht im Unternehmen präsent sein (Bild 3)

web & mobile developer 12/2016 Quelle: ifo-Randstad-Personalleiter-Befragung

Entwickler in Berlin



Der Anteil der Mobile-Entwickler ist in Berlin höher als in Gesamtdeutschland (Bild 4)

web & mobile developer 12/2016 Quelle: StackOverflow

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

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 1/2017 erscheint am 8. Dezember 2016

Neue Version von TypeScript

Statische Typisierung mag altbacken sein – sie hilft aber bei der Fehlervermeidung. Microsoft betreut mit der Programmiersprache TypeScript einen JavaScript-Dialekt, der Douglas Crockfords Sprache um ebendieses Konstrukt erweitert und diverse weitere Hilfsmittel zur sauberen Strukturierung des Programmcodes bietet. Die zweite Version der Sprache, die jetzt erschienen ist, steht im Zeichen der Kompatibilität: Neben einer Vielzahl von neuen Features zur Steigerung des Entwicklerkomforts arbeitete man im Hause Microsoft auch daran, TypeScript mit mehr existierendem JavaScript-Code kompatibel zu machen. Dieser Aspekt des Sprachdesigns ist insofern relevant, als TypeScript auf Seiten des Browsers keine neue Arbeitsumgebung voraussetzt. Die Sprache wird stattdessen vor der Ausführung in normales JavaScript transpiliert.

TYPO3 CMS, Routing und REST

Moderne Websites stellen heutzutage nicht mehr nur hübsche Seiten in HTML/CSS dar, sondern liefern auch Inhalte und Daten in maschinenlesbaren Formaten aus. JSON hat sich hierbei als schlanke, aber leistungsstarke Syntax etabliert. Unter Verwendung des Architekturstils REST lassen sich anspruchsvolle Webservices entwickeln. Hierbei werden Daten zum Beispiel im Backend eines Content-Management-Systems (CMS) verwaltet.

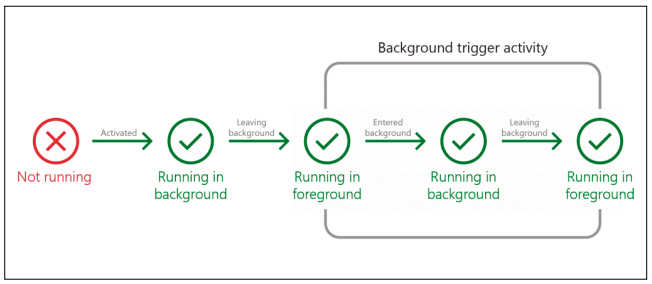
Vue.js – MVVM ganz einfach

Die Bibliothek Vue.js gewinnt seit einiger Zeit Aufmerksamkeit in der JavaScript-Szene. Sie ermöglicht die Erzeugung von MVVM-basierten Apps und kommt dabei aber mit weniger als 20 KByte aus. Von Vue.js gibt es sowohl eine Entwicklungs- als auch eine Produktivversion. Die Entwicklungsversion ermöglicht die Ausgabe von Informationen und Warnungen. Die für den Produktiveinsatz vorgesehene Version unterstützt diesen Modus nicht.

Responsive SVG-Icons

Das skalierbare Vektorgrafik-Format SVG (Scalable Vector Graphics) ist wie geschaffen für responsive Webseiten. Die meisten Webbrowser können ohne Erweiterungen einen Großteil des Sprachumfangs darstellen. Damit es mit den SVG-Icons klappt, gibt es jedoch einiges zu beachten. Der Artikel beschreibt, wie man SVG-Icons erstellt, verbessert und skaliert. Weitere Themen sind Responsive Icons, Browserprobleme und nützliche Tools.

dotnetpro



Ausgabe 12/2016 ab 17. November 2016 am Kiosk
Windows 10 gibt es inzwischen auf vielen verschiedenen Geräten. Wer für diese entwickeln will, sollte die Universal Windows Platform (UWP) einsetzen. Ein Schwerpunkt zeigt ausführlich, wie man für die UWP entwickelt.
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

Stellenmarkt

dotnetpro + web & mobile Developer

- 25.800 Exemplare Gesamtauflage
- 25.300 Newsletter-Empfänger
- 66.600 PI'S

.NET Architektur HTML5/JavaScript iOS/Android

Kontakt:
Klaus Ahlering, Thomas Wingenfeld • Tel. 089/74117-125 • sales@nmg.de

DWX Developer Week 2017

26.-29. Juni 2017, Messe Nürnberg

- **Treffen Sie** Ihre Zielgruppe auf einer der größten Entwickler-Konferenzen Europas
- **Profitieren Sie** von umfangreichen Marketingaktivitäten
- **Gestalten Sie** das Progamm selber mit und nehmen Sie am Call for Papers teil

developer-week.de/Ausstellung

Diese Kunden vertrauen uns:

adesso · adorsys · AIT · andrena objects · Compuware APM · ASTRUM IT · best-blu · byteAgenten · cellent · CENTIGRADE · centron · codecentric · combit · Configit · DATEV · DevExpress · ECS Webhosting · embarc · embarcadero · ERGOSIGN · evosoft · Fast Reports · GfK · GFU Cyrus AG · GOODGAME STUDIOS · HANSER · HeiReS · hemmersbach · ILNUMERICS · IBM · InnoGames · innoQ · intel Software · InterFace AG · INTERNETONE · ITECH PROGRESS · JetBrains · MATHEMA · MAXIMAGO · Microsoft · mitp · NOSERENGINEERING · www.oose.de · PayPal Developer · Progress · redgate · SafeNet · Saxonia Systems · SDX · SENACOR · sevenval · SOS Software Service · SPARX SYSTEMS · Telerik · TEXT CONTROL · THEOBALD SOFTWARE · ThingWorx · UID · UNIRENT

developer-week.de **#dwx17** **DeveloperWeek**