



VisualStudio1.de

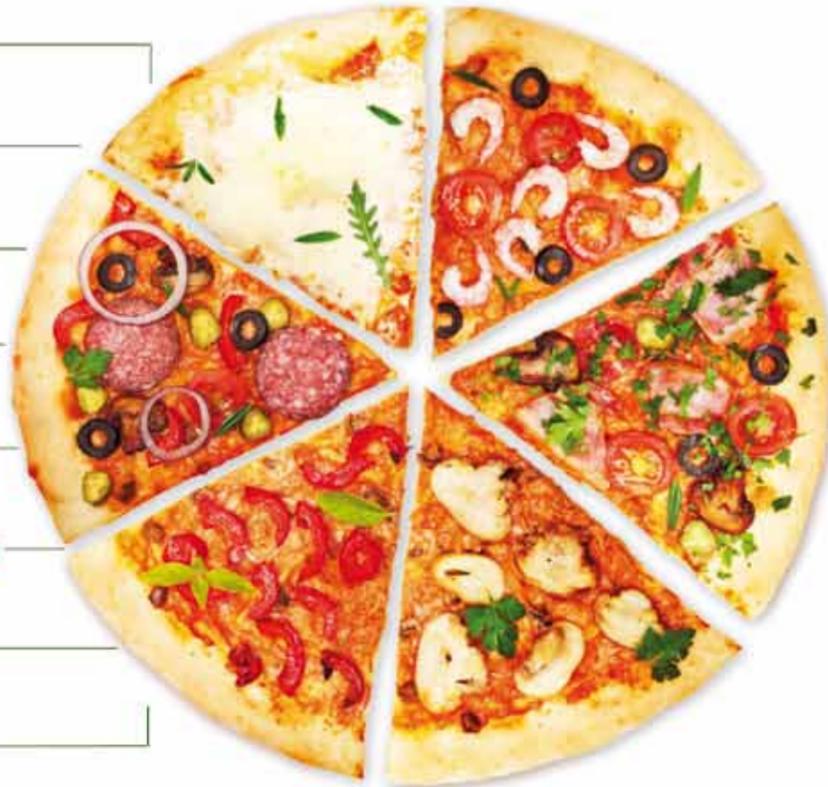
- Dem Windows Subsystem für Linux auf der Spur
- Display Templates
- C# – 7.0 Ein Einblick
- Windows Remote Arduino
- F# – Das unbekannte Wesen
- Force Feedback Programming



Learn .NET

... better than pizza

- ① .NET CORE
- ② C# / C++
- ③ WPF
- ④ XAMARIN
- ⑤ ASP.NET
- ⑥ ARCHITECTURE
- ⑦ UWP
- ⑧ TFS



© Valentina Ruzumova | Dreamstime.com

FRESH DELIEVERY

BERLIN // DRESDEN // LEIPZIG
 DÜSSELDORF // KÖLN // FRANKFURT
 MÜNCHEN // NÜRNBERG // BURGHAUSEN
 KARLSRUHE // STUTTGART // WIEN

ORDER NOW !

+49-8677-988 90
 +43 1 236 01932
 @PPEDV
 PPEDVAG

PPEDV.DE/DOTNET

NO WORDS TO SAY

Auch wenn gerne anderes behauptet wird, im Leben eines Menschen sind Geburt und Tod die zwei einzig wirklich wichtigen Ereignisse. In meiner Facebook Freundesliste befinden sich beinahe ein Dutzend Namen, die nicht mehr unter uns weilen. Die reale Existenz erlischt, die virtuelle bleibt. Nur selten nutzt jemand proaktiv den social Media Stream für sein persönliches Finale. Wenn es dann doch vorkommt, handelt es sich um amoklaufende Psychopaten.

So war ich überrascht und betroffen als ich den heutigen Tweet in meinem Stream zu Gesicht bekam.



Der mir völlig unbekannte Pieter Hintjens hat sich entschieden seinem Krebsleiden ein Ende zu setzen. Offensichtlich war Peter ein Mensch der der Welt was zu sagen hatte, als Konferenz Sprecher, als Blogger und auf Twitter. Nur am Ende, kurz vor seinem wichtigsten Ereignis, Stille.

Wie schnell und stark hat sich die Welt doch gewandelt. Noch leben Frauen und Männer, die niemals einen Computer benutzt und nie den eigenen Namen gegoogelt haben. Die digitale Transformation der Realität in die virtuelle Welt ist nicht nur im Gange, sondern im Laufschrift unterwegs.

Man fragt sich, wie dies weiter gehen wird. Was wird in 100 Jahren sein, in Anbetracht, dass das Internet gerade mal 30 Jahre alt ist.

Eine nicht zu überschätzende Entwicklung nimmt die künstliche Intelligenz ein. Nicht nur das mit BOT Frameworks und Machine Learning konkrete Lösungen für den gemeinen Softwareentwickler bereit stehen. Das Zeug ist dank Cloud auch weltweit vernetzt, wie ein großes zentrales Gehirn. Es weiß alles und hat unbegrenzte Kapazität zu lernen. Drängen sich da nicht parallelen zu diversen Weltuntergangs Blockbuster Movies auf? Maschinen beginnen wie Menschen zu handeln und Menschen beginnen wie Software zu werden. Leicht gruselig? Könnte eine Halloween Folge der Simpsons werden oder Realität.

No more words to say

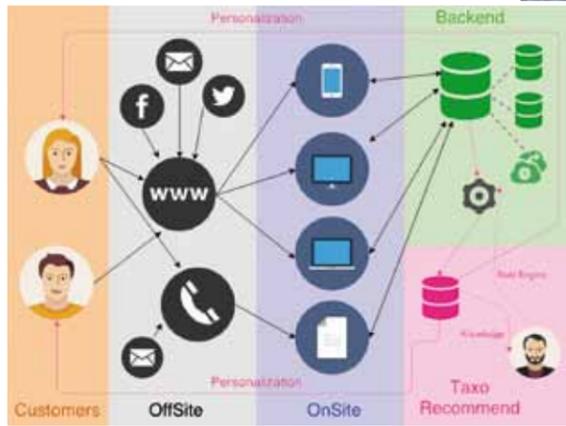
Ihr
HANNES PREISHUBER





52

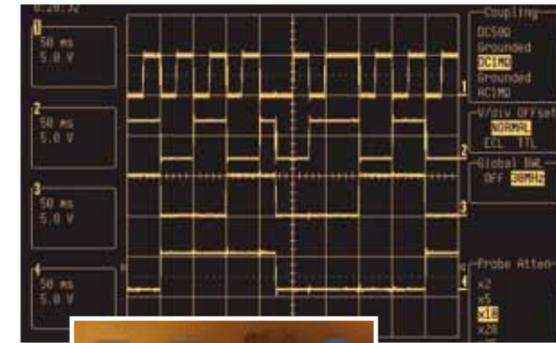
45



30



28



62



36



Editorial No words to say	3	Dem Erfolg auf der Spur	24
Inhalt	4	Test der ersten Generation von Microsofts Augmented Reality Brille.	
Autoren	6	TaxoRecommend	30
News	8	Multi-Channel B2B Präferenzanalyse System auf Basis der MS Dynamics CRM Datenstruktur und multiplen Taxonomien	
Crowdfunding	12	F# - Das unbekannte Wesen	36
C# 7.0 Ein Einblick	14	Funktionale Programmiersprachen als Alternative zu objektorientierten	
C# wird beständig weiterentwickelt. Der Artikel zeigt, wie es weitergehen könnte.		Marktanteile von Windows 10 sinken	41
Comic	18	Force Feedback Programming	42
Display Templates	19	Clean Code leicht gemacht	
Eigene SharePoint-Anzeigenvorlagen erstellen		Windows Remote Arduino	45
		Managed Code ist für Echtzeitaufgaben ungeeignet. Mit Windows Remote Arduino geht man einen neuen Weg.	
		Web-APIs, die sie bisher vermutlich nicht kannten	50
		Ein Einblick in weniger bekannte APIs, die aber für den einen oder anderen durchaus Sinn machen.	

Comic	51	Warum brauchen wir einen .NET Standard?	79
Von der Natur lernen	52	Was zum Schmunzeln	83
Genetische Algorithmen ermöglichen das effiziente Annähern an die optimale Lösung.		Besseres Arbeitsklima durch Verwendung neuer Rhetorik	
Die kniffligen Fälle beim Testen: Das automatisierte Testen von Softwaresystemen ist Stand der Technik. Jeder tut es. Oder sollte es zumindest.....	56	Windows Server Container	84
Starke Zielgruppe – Die Herausforderung für Entwickler: Die IT für die Generation 65+.....	62	Windows Server Container sind extrem praktisch wenn es darum geht Ressourcen zu sparen und so viele Anwendungen wie möglich zu virtualisieren.	
Visual Studio Preview 4 Was bringt es?	67	Neue Bücher	86
Vor einigen Tagen erschienen das vierte Preview der Version „15“		Kolumne	
Dem Windows Subsystem für Linux auf der Spur	72	Best of NuGet: WriteableBitmapEx.....	89
Das WSL ist eine von Microsoft entwickelte Technologie, um Anwendungen, nativ unter Windows laufen lassen zu können, die für Linux gebaut wurden.		Microsoft's NuGet-Paketmanager öffnet die Welt des Drittanbietercodes für Jedermann	

DIE AUTOREN DIESER AUSGABE

HEIKO ANGERMANN ist Dozent und Wissenschaftler im Bereich "Metadaten Management" und "Content Management" an der University of the West of Scotland. Er ist Betreiber des Beratungshauses "HA-ECM" (Internet: www.ha-ecm.com; E-Mail: angermann@ha-ecm.com), sowie Autor, Sprecher und Trainer im Themenfeld Microsoft SharePoint, war als SharePoint Administrator für einen Microsoft Gold Partner tätig und leitete internationale SAP-Projekte im Bereich Produkt-Daten-Management.



NAEEM RAMZAN ist Dozent an der University of the West of Scotland und Autor von "Social Media Retrieval". Er ist in europaweite Industrie- und Forschungsprojekte im Themenfeld "Social Media" involviert und im Bereich "Netzwerke" tätig.

FABIAN DEITELHOFF

lebt und arbeitet in Dortmund, der Metropole des Ruhrgebiets. Er studiert derzeit den Masterstudiengang Informatik mit dem Schwerpunkt Biomedizinische Informatik an der Hochschule Bonn-Rhein-Sieg in Sankt Augustin. Seine Schwerpunkte liegen in der Entwicklung von Visual Studio Erweiterungen, der Analyse und Beschreibung von Open Source Frameworks sowie im Rapid Prototyping. Beruflich ist er als freier Autor, Trainer, Sprecher und Softwareentwickler im .NET Umfeld tätig. Blog: www.fabiandeitelhoff.de, E-Mail: Fabian@FabianDeitelhoff.de, Twitter: [@FDeitelhoff](https://twitter.com/FDeitelhoff).



STEFAN LIESER (<http://lieser-online.de>) ist Informatiker aus Leidenschaft und arbeitet als Trainer/Berater/Autor/Entwickler. Er ist „gerne Lerner“ und sucht ständig nach Verbesserung und neuen Wegen, um die innere Qualität von Software sowie den Entwicklungsprozess zu verbessern. Gemeinsam mit Ralf Westphal hat er die Clean Code Developer Initiative (<http://clean-code-developer.de>) ins Leben gerufen. Mit der CCD School (<http://ccd-school.de>) bietet er Trainings und Beratung rund um das Thema Clean Code an.



TIM BOROWSKI studierte Informatik an der TU Darmstadt und entwickelt von klein auf Software mit .NET und C#. Er spezialisierte sich auf 3D Entwicklung und .NET im Microsoft Umfeld. E-Mail: timbo93@linux.com



GOLO RODEN ist Gründer und CTO der the native web GmbH, eines auf native Webtechnologien spezialisierten Unternehmens. Er hat das erste deutschsprachige Buch zu Node.js geschrieben und wurde von Microsoft dreifach als MVP ausgezeichnet. Webseite: www.goloroden.de, E-Mail: webmaster@goloroden.de

MAXIMILIAN SCHWEIGERDT

ist Dev. Trainer bei der ppedv AG. In Schulungen und Kursen gibt er sein Wissen an die Kursteilnehmer weiter. Die Schwerpunkte liegen dabei im Bereich der Webprogrammierung wie z. B.: HTML und CSS, JS + Libs sowie TypeScript. Als leidenschaftlicher Mikrocomputer-Bastler beschäftigt er sich in seiner Freizeit mit dem MSP und Arduino. E-Mail: maximilians@ppedv.de



ANNETTE HEIDI BOSBACH betreut Legacy-systeme der Tamoggemon Holding k.s. und ist zudem für die Bearbeitung politischer und militärischer Aufträge aus dem Osten und aus Afrika zuständig.



TAM HANNA hat diverse Anwendungen und Spiele für Symbian, PalmOS und bada entwickelt. Der studierte Elektrotechniker betreibt mit seinem Team eine Gruppe von Online Newsdiensten für Mobilcomputer-Techniker und Power User. www.youtube.com/user/MrTamhan, E-mail: tamhan@tamoggemon.com

DR. ECKHARD HAUENHERM

ist Inhaber der Firma Dr. Hauenherm (www.hauenherm.de). Er berät Unternehmen bei der Planung und Umsetzung von Strategien und Lösungen zum Informationsmanagement. Er arbeitet vorwiegend mit Microsoft-Plattformen wie SharePoint, Exchange und Active Directory mit Fokus auf Projektmanagement-Lösungen.



WALTER SAUMWEBER hat langjährige Erfahrung als Entwickler, Berater und Dozent. Er ist Autor von zahlreichen Fachbüchern und Beiträgen in Computer-Fachzeitschriften. Seine Tätigkeitsschwerpunkte sind die Realisierung von Unternehmenslösungen in Client-Server-Umgebungen und Workflow-Anwendungen.



OLENA BOCHKOR

studierte Betriebswirtschaftslehre u.a. mit dem Schwerpunkt Wirtschaftsinformatik. Weitere Informationen zu diesen und anderen Themen der IT finden Sie unter <http://it-fachartikel.de>.



HANNES PREISHUBER

ist CEO der ppedv AG und Microsoft-Experte (MCSD, MCAD, MCT) mit Schwerpunkt auf Web-Technologien. Er ist Sprecher, Trainer und Autor rund um Development-Themen.



DR. VEIKKO KRYPCZYK

studierte und promovierte Betriebswirtschaftslehre mit dem Schwerpunkt Wirtschaftsinformatik. Er arbeitet als Fachautor und ist begeisterter Programmierer.



STEFAN OBER absolvierte seine Ausbildung ursprünglich als IT-Systemelektroniker und arbeitete daraufhin kurze Zeit als Software-Tester. Mittlerweile vermittelt er sein Wissen als Trainer bei der ppedv AG, hauptsächlich im Bereich Windows Server. E-Mail: StefanO@ppedv.de

DAVID C. THÖMMES studierte Medieninformatik an der FH Kaiserslautern und entdeckte schon früh seine Leidenschaft für die Mensch-Computer-Interaktion und das Software Engineering. Als Senior Software & UX Engineer sowie Geschäftsführer von Shapefield gilt seine Passion heute der benutzerzentrierten Gestaltung sowie der technischen Entwicklung eindrucksvoller Benutzeroberflächen.



RALF WESTPHAL (ralfw.de) ist freiberuflicher Berater, Projektbegleiter, Referent, Autor und Trainer für Themen rund um Softwarearchitektur und die Organisation von Softwareteams. Er ist Mitgründer der Initiative „Clean Code Developer“ (CCD) für mehr Softwarequalität (clean-code-developer.de), propagiert kontinuierliches Lernen mit der CCD School (ccd-school.de) und möchte mit ich-verspreche.org zu mehr Zuverlässigkeit motivieren. Sie erreichen ihn über seinen Blog <http://ralfw.de/blog> oder Twitter: [@ralfw](https://twitter.com/ralfw).

Pixel – das erste komplett selbst entwickelte Smartphone von Google

Erstmalig gelang es Google, Smartphones mit eigener Soft- sowie Hardware auszustatten und somit auf die Unterstützung von anderen Hersteller wie beispielsweise HTC, LG und Motorola zu verzichten. Die Google Smartphones, das Pixel und das Pixel XL, versprechen eine hervorragende Kamera wie auch eine lange Akkulaufzeit. Zudem haben sie den Google Assistant integriert sowie das Betriebssystem Android Nougat (7.1) installiert. Erhältlich sind die beiden Smartphone-Varianten in den Speichergrößen 32 GB und 128 GB. Die Hauptkamera auf der Smartphone-Rückseite arbeitet mit 12 Megapixel und gestattet Slow-Motion aufnahmen, während die Frontkamera mit 8 Megapixel arbeitet. Ein Fingerabdrucksensor wie auch ein USB-Typ-C-Anschluss gehört ebenfalls zu den Ausstattungsmerkmalen. Ab 20. Oktober beginnt der Verkauf der beiden Smartphones in Deutschland. Erhältlich ist das Pixel ab einen Verkaufspreis von 759,95€ und das Pixel XL ab 869,95€. *Quelle: technikneuheiten.com*



Das neue Google Pixel

Bildquelle: Google

PostgreSQL 9.6

Die neueste Version der PostgreSQL 9.6 bietet verschiedene Neuerungen

an. Zu den wichtigsten zählen die parallelen Queries, die Volltextsuche nach Phrasen sowie die verbesserte Aufräumfunktion autovacuum. Nennenswert ist dabei das

nicht nur bei PostgreSQL ein Versionswechsel stattfand, sondern auch das Backup-Tool Barman nun zeitgleich auf die Version 2.0 upgedatet wurde. *Quelle: heise.de*

Dell XPS 13 – nun als Developer-Edition verfügbar



XPS 13 (Bildquelle: Dell)

In Europa wie auch in den USA steht nun Dells XPS 13 mit Kaby-Lake-Chips auch als sogenannte Developer-Edition mit Ubuntu 16.04 LTS zur Verfügung. Anzumerken ist, dass die kleinste Variante von Dells XPS 13 jedoch nur in den USA angeboten wird.

TREIBER-UNTERSTÜTZUNG

Die Notebook Nutzung sollte kein Problem darstellen, da alle notwendigen Hardwaretreiber bereits eingepflegt wurden. Dadurch sollen die Notebooks auch mit anderen aktuellen Linux-Distributionen genutzt werden können. *Quelle: golem.de*



Adblockerverbot?!

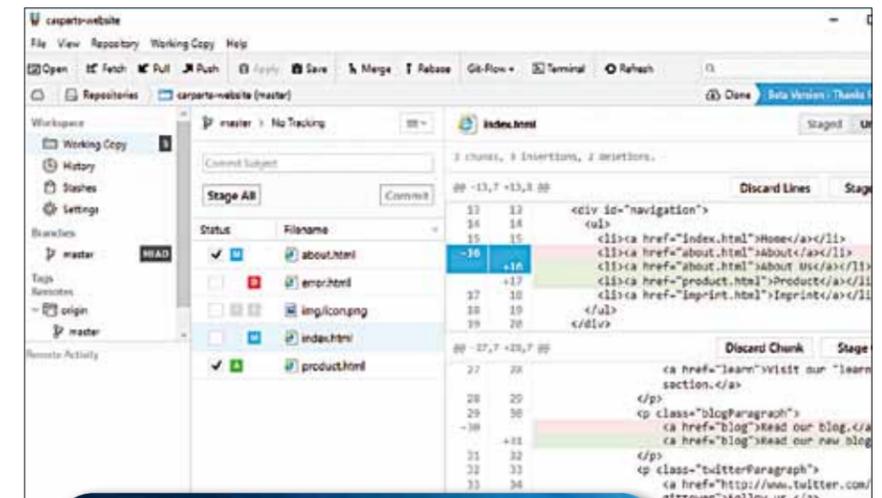
In ihrem Abschlussbericht forderte die Kommission aus Bund und Länder zur Medienkonvergenz dazu auf ein gesetzliches Adblockerverbot zu überprüfen. Es besteht jedoch laut Ansichten von di Fabio und Kuhlmann derzeit keine Verpflichtung des Gesetzgebers, Medienbestände zu schützen. Ohne eine existenzielle Gefährdung der Medien ist also ein

Eingreifen des Gesetzgebers nicht erforderlich. Sollte trotzdem versucht werden ein Adblockerverbot in die Wege zu leiten, wird dies nach Ansichten von di Fabio und Kuhlmann nicht einfach werden, da Kuhlmann beispielsweise davon ausgeht, dass eine Kollision mit den Grundrechten stattfinden könne. *Quelle: golem.de*

Die Public Beta Phase erreicht der Windows Git-Client Tower

Quelle:ournova.com

Die öffentliche Beta-Phase von Tower for Windows, einem neuen Git-Client für die Plattform, wurde vom Software-Entwicklerournova am 10. Oktober angekündigt. Die Software steht mittlerweile zum kostenlosen Download für alle Interessierten zur Verfügung. In der geschlossenen Beta-Phase war die Resonanz überwältigend. Jetzt ist man auf die Rückmeldungen zur Public-Beta gespannt, zudem freut man sich mit Hilfe der Nutzer, Tower zum besten Git-Client für Windows zu entwickeln. So die Aussage von Tobias Günther, CEO vonournova. Ziel ist, dass das Arbeiten am Desktop mit dem Versionskontrollsystem Git einfacher und komfortabler wird. Auf der Tower-Website kann die Beta-Version kostenlos heruntergeladen werden: www.git-tower.com/windows



Video-Kanal für Softwareentwickler

Das Videoportal Twitch erweiterte seine Palette um einen Kanal. Dieser Kanal wendet sich dem Thema Softwareentwicklung zu und trägt den passenden Namen „#programming“. Dort sollen Online-Kurse, Vorträge, Hackathons und Gesprächsthemen rund um das Thema Softwareentwicklung ausgestrahlt werden. Dieser Kanal ist jedoch

nicht die einzige Weiterentwicklung von Twitchs Engagement außerhalb der reinen Games-Community, denn bereits seit beträchtlicher Zeit gibt es den sogenannten „#GameDev“-Kanal für Spieleentwickler. *Quelle: heise.de*

Microsoft: Workplace as a Service



Bildquelle: Irene Nadler (Microsoft)

Kleineren Firmenkunden wird durch „Workplace as a Service“ die Möglichkeit geboten mit individuell konfigurierbaren Bundles der aktuellsten Hard- und Software ihre Arbeitsplätze auszustatten. Unternehmen müssen also nur noch die modernste Hardware mit den individuellen Anforderungen des digitalen Arbeitsplatzes kombinieren. Um „Workplace as a Service“ zu nutzen muss man einen „WaaS“-Reseller mieten. Vermietet werden die speziell konfigurierten Bundles für 12, 24 oder 36 Monate. Durch dieses Modell haben Firmen die Möglichkeit flexibler auf aktuelle Trends zu reagieren. Zusätzlich profitieren sie

beispielsweise von einer aktuellen und sicheren IT. Durch dieses Modell wird den Kunden auch der Zugriff auf „Surface as a Service“ gestattet. Über den ALSO Cloud Marketplace wird Firmenkunden angeboten das „SaaS“-Angebot zu konfigurieren, bereitzustellen und zu verwalten. Anzumerken ist, dass bei „WaaS“ die Hardware-Versicherung inkludiert ist. Bei Beschädigung oder Verlust wird das Surface innerhalb von 24 Stunden ersetzt. Interessierte Unternehmen können sich auf der Website www.microsoft.de/workplace-as-a-service über das Angebot informieren. *Quelle: news.microsoft.com*

Sony Xperia Ear

Mehr als ein drahtloser Ohrhörer ist genau die richtige Bezeichnung für das Sony Xperia Ear. Es reagiert nämlich auf Sprachbefehle und Kopfnicken und ermöglicht dem Nutzer somit eine Erleichterung in allen Bereichen sowie einen Einblick in neue Kommunikationswege. Über eine Bluetooth-Verbindung können die Befehle an das verbundene Smartphone weitergeleitet werden.

Per Abruf informiert das Sony Xperia Ear seinen Nutzer über die neuesten Informationen. Mittels Sprachsteuerung kann der Nutzer dem Sony Xperia Ear Befehlen, seine Texte zu diktieren, eine Telefonverbindung herzustellen, Informationen zu sammeln oder einfach zu navigieren. Außerdem kann das Sony Xperia Ear Nachrichten vorlesen und durch Benutzerreaktion auf diese Antworten.

Ausgeliefert wird das Sony Xperia Ear ab Mitte November 2016. Vorbestellen kann man es sich jedoch im Sony Mobile Online-Store für 199,- Euro (UVP).

Quelle: technikneuheiten.com



Sony Xperia Ear

Bildquelle: Sony Mobile

Google Daydream

Google brachte Virtual Reality mit seinem Produkt Google Daydream eine Entwicklungsstufe weiter. Für hochauflösendes Virtual Reality wird mit Google Daydream eine Mindestanforderung geschaffen. Dabei wird von Smartphones, VR-Headsets und Controllern gesprochen. Google Daydream kann kabellos mit dem Smartphone verbunden werden. Zudem ist das Produkt laut Google um 30 Prozent leichter als konkurrierende VR-Brillen. Ein anderes Feature ist die Ausstattung des Controllers mit Bewegungssensoren. Interessant ist, dass für Google Daydream-Geräte der PlayStore als VR-Version erscheint. Erhältlich ist das Produkt ab November für 79 Dollar in variierenden Farben.

Quelle: chip.de



Google Daydream

Bildquelle: Google

Whatsapp-Konkurrent Immmr gestartet



Immmr

Bildquelle: Deutsche Telekom

Das Wirtschaftsmagazin Capital berichtet, dass die Deutsche Telekom den bereits im Februar vorgestellten Whatsapp-Konkurrenten Immmr in diesen Wochen erstmalig in der Slowakei startet. Der Netzbetreiber verspricht, dass das in Berlin entwickelte Angebot zu einem globalen Produkt wird. Sollte das Messenger-Angebot bei der Tochter Slovak Telekom funktionieren, so soll der Service daraufhin auch in Kroatien verfügbar sein. Telefonate, Nachrichten und persönliche Kontakte im Browser oder der App seien mit dem Messenger Immmr durch Geräteunabhängigkeit immer verfügbar. Um Immmr verwenden zu können muss man sich mit seiner Mobilfunknummer beim Service registrieren. *Quelle: golem.de*

JavaScript: jQuery Foundation nun unter Linux Foundation

Die jQuery Foundation wurde von der Linux Foundation unter dem Namen JS Foundation als neues Unterprojekt übernommen. Client- sowie serverseitige JavaScript-Anwendungen sollen gefördert werden. Die Organisation will sicherstellen, dass die zentralen Open-Source-Projekte dauerhaft weiterentwickelt werden. Wichtig ist, dass die JS Foundation nicht von vorne beginnen muss. Das heißt es werden Projekte aus der alten jQuery Foundation übernommen, wie beispielsweise ESLint, Grunt und jQuery. Als Starthilfe für neue Projekte soll ein



Mentorship-Programm dienen. Auf der Website der JS Foundation kann beispielsweise eine Auflistung aktueller Projekte aufgefunden werden. *Quelle: heise.de*

► CROWDFUNDING

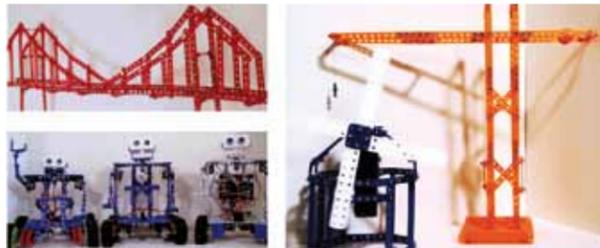
CamBuddy Pro: der erste All-In-One DSLR Smart Controller

Mit dem CamBuddy Pro kann man seine DSLR Kamera vom Smartphone aus steuern. Im Zusammenspiel von CamBuddy und der Joopic App hat man die Möglichkeit die Kamera komplett vom Smartphone aus zu steuern, sei es beispielsweise die Blende, die ISO oder weitere Einstellungen. Selbst das ultimative Selfie kann man aus einer Entfernung von bis zu 100 Metern aufnehmen. Dies sind nur einige Features welche das Produkt ermöglicht. Anzumerken ist, dass man bis zu 128 Kameras auf einem Smartphone steuern kann.

Woher: Houston, USA via kickstarter.com
Ziel: Mitte November 2016
Finanzierungsziel: \$ 30.000



Tinkathing – Baukastensystem



Tinkathing ist ein Baukastensystem, welches junge Investoren dazu inspiriert beispielsweise Spielzeugroboter oder neue Konstruktionen zu erbauen. Kinder können dann mit diesem Roboter spielerisch die Grundlagen der Programmierung, der Elektronik und der Mechanik erlernen. Besonders zielt dieses System darauf ab, sich neues Wissen durch spaßige Aktivitäten anzueignen.

Woher: Berlin, Deutschland via Kickstarter.com
Ziel: Ende November 2016 · Finanzierungsziel: 100.000 €



Air Charge – schwebendes Ladegerät für iPhone 7 und 6



Das Air Charge ist das einzige drahtlose schwebende iPhone-Ladegerät, das das Smartphone sozusagen „im Flug“ per Induktion aufladen kann und sich dabei auch noch dreht.

Woher: San Francisco, USA via kickstarter.com
Ziel: Mitte November 2016
Finanzierungsziel: \$ 5.000



Solartab C das weltchnellste Solarladegerät

Das Solartab C ist das erste Solarladegerät mit einem USB-C Anschluss sowie zwei USB Anschlüssen. Es bietet einen 6,5W Sonnenkollektor, wie auch eine Batterie mit 9,000mAh. Anzumerken ist, dass das Gerät wasserfest ist.

Woher: San Francisco, USA via indiegogo.com
Ziel: Mitte November 2016
Finanzierungsziel: \$ 50.000



The Full Stack Developer Bundle Lernen durch den Bau von über 40 Apps

Dieser Kurs zielt darauf ab, die Kursbesucher mit dem notwendigen Wissen auszustatten, um aus ihnen vollständige Webentwickler zu machen. Das Wissen wie man beispielweise HTML/CSS Layouts gestaltet, mit NoSQL Datenbanken arbeitet oder Ruby on Rails verwendet, wird in diesem Kurs vermittelt. Es steht jedoch

ein viel breiteres Spektrum zur Verfügung als die beispielsweise genannten Inhalte. Laut Beschreibung soll mit diesem Kurs ein besserer Programmierer werden.

Woher: Orlando, USA via kickstarter.com
Ziel: Anfang November 2016
Finanzierungsziel: \$ 1.500

MagBolt der erste Anschluss für iPhone7 und Android

Der MagBolt ist der erste 2-in-1 Anschluss für Apple und Android Geräte. Es ermöglicht einem durch Drehen des Anschlusses entweder ein Apple Gerät oder ein Android Gerät aufzuladen.

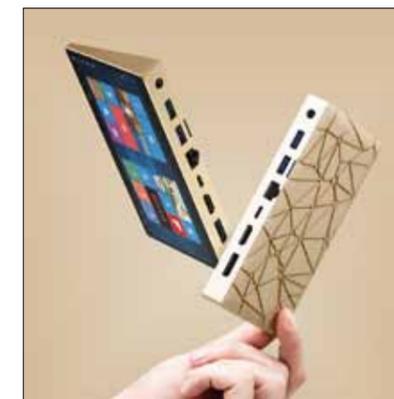
Woher: Los Angeles, USA via kickstarter.com
Ziel: Ende November 2016
Finanzierungsziel: \$ 12.000



Ockel Sirius A der vielseitigste mini PC der Welt

Der Ockel Sirius A ist ein vollwertiger Windows 10 PC, jedoch mit dem Unterschied, dass dieser die Größe eines Smartphones besitzt. Anzumerken ist dabei, dass er genauso leistungsfähig ist wie ein Desktop-PC. Eigenschaften welche dieses Gerät so einzigartig machen sind beispielsweise die extreme Vielseitigkeit, die Größe und die gebotene Funktionalität. Der Ockel Sirius A besitzt zwei USB 3.0 Anschlüsse, einen DisplayPort, einen HDMI Anschluss, einen USB Typ-C Anschluss sowie einen LAN – Anschluss.

Woher: Den Haag, Niederlande via indiegogo.com
Ziel: Anfang November 2016 · Finanzierungsziel: \$ 100.000



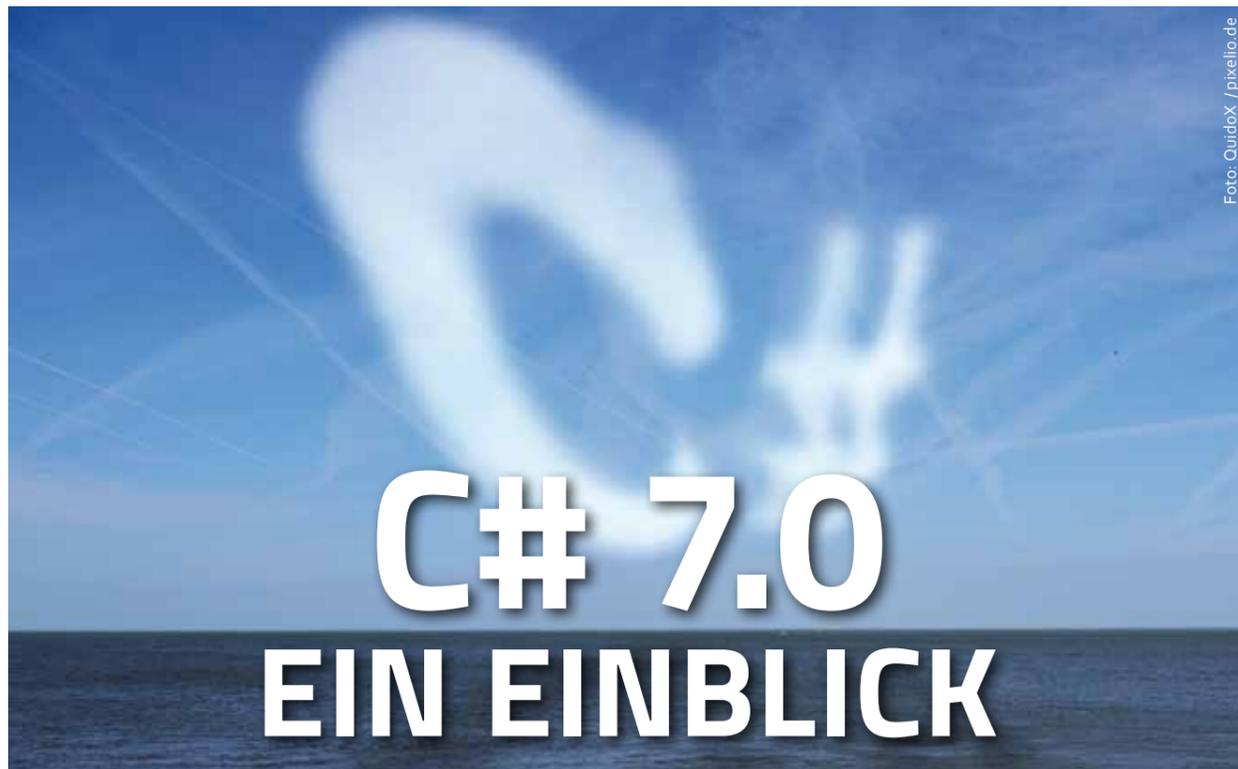
VoCore2 der weltkleinste Wireless Router

Der VoCore2 ist ein münzengroßer Open Source Linux Computer, sowie ein funktionsfähiger WLAN-Router. Er dient mehreren Funktionen, wie beispielsweise als VPN-Gateway zur Netzwerksicherung wie auch als

private Cloud um Fotos, Videos und dergleichen zu speichern.

Woher: Taiyuan, China via indiegogo.com
Ziel: Anfang November 2016
Finanzierungsziel: \$ 6.000





C# wird beständig weiterentwickelt. Der Artikel zeigt, wie es weitergehen könnte.

Autor: FABIAN DEITELHOFF

Nicht nur das Ökosystem rund um .NET und C#, wie zum Beispiel Visual Studio, wird konstant weiterentwickelt. Auch die Sprachen C# und Visual Basic .NET selbst erfahren bei jeder Version mehr oder weniger große Änderungen. Zum einen werden so Wünsche der Community entsprochen, zum anderen bleibt die Zeit nicht stehen und Anpassungen an Programmiersprachen sind notwendig. Wo die Reise in Bezug auf C# 7.0 hingeht, was geplant ist und was in den Sternen steht, zeigt dieser Artikel.

In der Welt der Softwareentwicklung ist nichts von Dauer. Weiterentwicklungen gibt es an jeder Ecke. Von kleinen bis großen Änderungen, von vermeintlich bahnbrechenden Entwicklungen bis hin zu unbemerkten Änderungen ist alles dabei. Oftmals beziehen sich solche Änderungen lediglich auf das Ökosystem selbst, das sich um eine Programmiersprache entwickelt hat. Neue Features in Frameworks und Bibliotheken zum Beispiel oder Verbesserungen bei Entwicklungsumgebungen wie Visual Studio.

Aber auch an Programmiersprachen selbst, wie zum Beispiel C# und Visual Basic .NET, geht die Zeit nicht spurlos vorbei. Anforderungen der Community, geänderte Zielrichtungen und Veränderungen bei anderen Programmiersprachen machen Anpassungen aller Art

notwendig. Insbesondere funktionale Aspekte aller Art hatten in der Vergangenheit großen Einfluss auf Programmiersprachen wie C# und Java.

DER IST-ZUSTAND

In den vergangenen Jahren wurde vermehrt beobachtet, wie Features und Designansätze zwischen verschiedenen Programmiersprachen ausgetauscht wurden. Nicht eins zu eins ohne Anpassungen, was zwangsläufig geschehen muss, da die jeweiligen Programmiersprachen ganz grundsätzlich verschiedene Ansätze verfolgen. Allerdings sind die Anpassungen dennoch so markant, dass sie ins Auge stechen.

Eine wichtige Quelle für Neuerungen sind die funktionalen Programmiersprachen mit ihren funktionalen Ansätzen beziehungsweise Designentscheidungen. So macht es den Eindruck, dass C# mehr und mehr Funktionen von F# übernimmt. Aber auch Skriptsprachen, die für besonders kompakten Syntax und spezifische Features stehen, haben einen Einfluss.

Schon in C# 6 sind daher viele Features hinzugekommen, die bereits vorhandene Dinge teils vereinfachten oder neue Funktionen hinzugefügt haben, die von der Community herbeigesehnt wurden. Die Vereinfachungen kommen in der Regel dadurch zustande, dass weniger Code als vorher notwendig ist, um eine identische Anweisung oder ein identisches Feature auszudrücken. Die neuen Sprachfeatures in C# 6 sind unter anderem:

- Initialisierungen für Eigenschaften (auto-properties)
- Ausdrücke oder Funktionen in Lambda-Ausdrücken
- Static in using-Anweisungen
- Der Null-Conditional Operator
- String-Interpolation
- Exception-Filter
- Await in Catch- und Finally-Blöcken

Dieser Auszug von Features in C# 6 macht deutlich, die vollständige Liste ist unter [1] zu finden, dass viele der neuen Funktionen kleinere Verbesserungen, lang erwartete Features oder Fehlerbehebungen sind. Bei C# 7.0 wird deutlich, dass F#, beziehungsweise funktionale Sprachen an sich, einen deutlich größeren Einfluss gehabt zu haben scheinen.

TECHNISCHE VORAUSSETZUNGEN

Bei C# 6 ist es Visual Studio 2015, das als technische Voraussetzung gilt, um von den Sprachfeatures profitieren zu können. Mit Visual Studio 2015 lassen sich alle Features ausprobieren beziehungsweise einsetzen, die gegenüber der Vorgängerversion neu hinzugekommen sind. Bei C# 7.0 ist es die Nachfolgeversion von Visual Studio, die aktuell noch unter dem Namen Visual Studio „15“ firmiert. Beim Schreiben des Artikels ist die Preview 4 [2] vom 22. August 2016 aktuell, die über den gleichen Link kostenfrei heruntergeladen werden kann.

In dieser Preview muss zuvor ein Symbol in den Build-Eigenschaften eines Projekts gesetzt werden, damit tatsächlich alle neuen Features von C# 7.0 ohne Einschränkungen ausprobiert werden können. Abbildung 1 zeigt dieses Symbol mit Namen `__DEMO__`. Fehlt es, kann Code mit den neuen Features nicht kompiliert werden. Ärgerlicherweise tauchen auch keinerlei Fehlermeldungen auf, die darauf hinweisen. Es funktioniert schlicht und ergreifend nicht. Mit dem zusätzlichen Symbol ist es allerdings kein Problem.

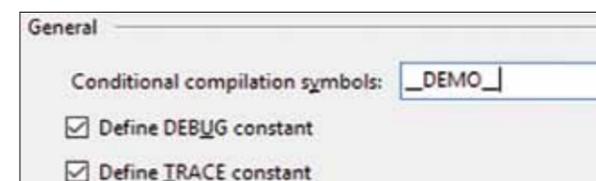


Abbildung 1: Das zusätzliche Symbol, um C# 7.0 Features testen zu können.

ENTWICKLUNG DER NEUEN SPRACHVERSIONEN

Die Entwicklung aller neuen Funktionen für C# 7.0 und Visual Basic .NET 15 findet über die Roslyn-Plattform statt. Das bedeutet auch, dass die Roslyn Plattform, deren Code auf GitHub [3] zur Verfügung steht, ein sehr guter Startpunkt für Informationen zu den neuen Sprachfeatures ist. Insbesondere für all diejenigen weitergehenden Informationen, die in einem Einführungsartikel wie diesem hier nicht behandelt werden können.

Ganz allgemein gibt es zwei Arten von neuen Sprachfeatures. Zum einen die für die nächste Version fest eingeplanten Funktionen, die in diesem Abschnitt näher beschrieben sind. Darüber hinaus gibt es aber schon jetzt Features, die zwar aus Sicht des Roslyn-Teams interessant sind und an denen durchaus gesteigertes Interesse besteht, die allerdings erst in der nächsten Version zu erwarten sind. Mangels sinnvoller Versionsnummer für die nächste Version wird diese als C# 7.0 +1 beziehungsweise Visual Basic .NET +1 bezeichnet.

Zusätzlich werden die Sprachfunktionen zudem in die beiden Kategorien mit starkem (strong) Interesse und einigem (some) Interesse eingeordnet. Ganz allgemein gilt, dass es keine Versprechungen gibt, dass es ein Feature tatsächlich in eine bestimmte Version schafft. Neue Funktionen werden laufend verändert, verschoben, verworfen oder tatsächlich implementiert. Ein Blick in das Roslyn Repository auf GitHub lohnt daher allemal. Dort ist auch erklärt, wie eine Mitarbeit an der neuen Sprachversion möglich ist, da viel Feedback von der Community eingeholt wird, während sich die neuen Funktionen in der Entwicklung befinden.

AKTUELL EINGEPLANTE FUNKTIONEN

Für die initiale Version von C# 7.0 sind bisher acht Features eingeplant, die sich im „Finishing“-Status befinden und in den Master Branch eingepflegt wurden. Das sind gute Anzeichen dafür, dass es diese Funktionen tatsächlich in die neue Version von C# schaffen. Die eingeplanten Funktionen sind die folgenden:

- Binary Literals
- Digit Separators
- Local Functions
- Type Switch
- Ref Returns
- Tuples
- Out var
- ValueTask
- Address of Static (Feature-Spezifikation)

Einige dieser neuen Funktionen sind kleinere Ergänzungen zum C# Sprachumfang. Andere haben wiederum einen deutlich größeren Einfluss auf die Art und Weise, wie Software in C# beziehungsweise Visual Basic .NET geschrieben werden kann. Zur ersten Kategorie gehören die beiden erstgenannten Features.

Binäre Literale [4] erlauben es zukünftig, Binärdaten direkt im Code unterzubringen und vom Compiler interpretieren zu lassen. Syntaktisch und semantisch sind sie identisch zu hexadezimalen Literalen, mit dem großen Unterschied, dass nicht die Buchstaben x oder X zum Einsatz kommen, sondern b oder B. Ein binäres Literal lässt sich wie folgt definieren:

```
int nineteen = 0b10011;
```

Die Variable `nineteen` enthält anschließend den interpretierten Wert 19. Dieses Feature ist sinnvoll für Enumerationen als Flag und im Bildungsbereich.

Das zweite Feature geht Hand in Hand mit den binären Literalen. Hintergrund ist, dass durch die neue Schreibweise ebendiese binären Literale sehr viel Platz in Anspruch nehmen können. Durch die Trennzeichen für Ziffern (Digit Separators [5]) ist es möglich, die Daten in Pakete aufzuteilen, die dadurch deutlich besser lesbar sind. Dieses Feature erlaubt zum Beispiel die folgenden Schreibweisen:

```
int bin2 = 0b1001_1010;
int hex = 0x1b_a0_44_fe;
int dec = 33_554_432;
int weird = 1_2__3;
```

Diese Beispiele machen deutlich, dass das lediglich ein Feature für die Lesbarkeit ist und ansonsten keinerlei Einfluss hat. C# 7.0 folgt hier dem Beispiel von Java, das ebenfalls den Unterstrich als Trennzeichen nutzt. Erlaubt ist das Zeichen überall in einem binären Literal, mit Ausnahme am Anfang und am Ende einer Ziffernfolge.

Des Weiteren führt C# 7.0 die sogenannten lokalen Funktionen [6] ein. Sie erlauben es, Funktionen innerhalb einer anderen Funktion zu definieren, also die innere Funktion auf den Sichtbarkeitsbereich der äußeren zu beschränken. Ein Anwendungsfall ist es zum Beispiel, private

Listing 1: Eine lokale Funktion

```
static void Main(string[] args)
{
    int LocalAdd(int a, int b)
    {
        return a + b;
    }

    var result = LocalAdd(10, 20);
}
```

Listing 2: Ein Tuple als Rückgabewert einer Methode.

```
private static (int sum, int count) Calculate(List<int> values)
{
    var s = 0;
    var c = 0;

    ...

    return (s, c);
}

var tuple = Calculate(...);
Console.WriteLine($"Sum: {tuple.sum}, count: {tuple.count}");
```

Methoden zu ersetzen, die zuvor lediglich von einer einzigen anderen Methode aufgerufen wurden. Eben um dorthin einen Teil der Logik für eine bestimmte Aufgabe auszulagern. Diese Methode kann nun als lokale Funktion implementiert werden, umso deutlich zu machen, wozu der Code tatsächlich gehört. Ob das die Lesbarkeit tatsächlich stark erhöht, ist sicherlich von Fall zu Fall zu entscheiden. Listing 1 zeigt das Vorgehen für lokale Funktionen an einem minimalen Beispiel.

Aktuell gibt es Diskussionen darüber, ob die verwendete Syntax, um eine lokale Funktion zu definieren, gut gewählt ist und wo eine lokale Funktion definiert werden soll. Denn es kann sich durchaus problematisch auf den Lesefluss auswirken, wenn im Code eine lokale Funktion auftaucht, die einen Wert zurückliefert. Das enthaltene `return`-Statement ist ansonsten ein gutes Zeichen dafür, dass der aktuelle Ablauf an dieser Stelle verlassen wird. Nun ist zwingend darauf zu achten, ob es sich um eine lokale Funktion handelt und die `return`-Anweisung nur diese verlässt. Zusätzlich ist zu berücksichtigen, dass lokale Funktionen private Methoden überschreiben. Auch hier kann es zu Missverständnissen kommen. Damit es zu keinen Problemen beim Lesen von Variablen kommt, lokale Funktionen könnennämlich auf die Variablen des übergeordneten Sichtbarkeitsbereichs zugreifen, müssen diese Variablen vor dem Aufruf der lokalen Funktion einen zugewiesenen Wert besitzen. Ansonsten kommt es zu Fehlermeldungen.

Und auch Tuples [7] scheinen in C# 7.0 Einzug zu finden. Ein Feature, das schon aus diversen anderen Programmiersprachen bekannt ist. Dahinter verbirgt sich die Möglichkeit eine Menge von typisierten Werten temporär zu einer Gruppe

zusammenzufassen. Ohne das diese Gruppe ein eigenes Konzept beziehungsweise einen eigenen Typ-Namen benötigt. Ein häufiger Anwendungsfall ist das Gruppieren von Ergebnissen eines Methodenaufrufs. Es besteht aktuell zwar die Möglichkeit eine Methode mit `out`-Parametern auszustatten, dass funktioniert allerdings nicht mit Methoden, die mit dem `async` Schlüsselwort definiert sind. Zudem müssen bei `out`-Variablen die Variablen zunächst definiert werden, anschließend ist die Methode aufzurufen, um die Variablen zu füllen. Erst danach kann mit den Werten weitergearbeitet werden. Die aktuelle Syntax-Definition für Tuples sieht eine Syntax ähnlich zu der Gruppierung von Methoden-Parametern vor. Also eine durch Klammern eingeschlossene Menge von Typdefinitionen mit Namen, die durch Komma voneinander getrennt sind.

Listing 2 zeigt ein Beispiel für eine Methode mit einem Tuple als Rückgabewert und eine Möglichkeit, diese Methode aufzurufen und das Ergebnis weiter zu verarbeiten.

Alternativ kann ein Tuple, anstatt es in einer Variable zu speichern, beim Aufruf einer Methode auch direkt in seine Bestandteile zerlegt werden. Beim vorherigen Listing zum Beispiel durch den folgenden Aufruf:

```
(var sum, var count) = Calculate(...);
```

Die vorgestellten Features sind lediglich ein Ausschnitt aus den neuen Funktionen, die für die nächste Version von C# und Visual Basic .NET geplant sind. Eine vollständige Liste ist unter [8] zu finden.

EIN AUSBLICK

Diese Liste beherbergt zudem eine Übersicht über diejenigen Funktionen, die für die übernächste Version, also für das angesprochene C# +1 und Visual Basic .NET +1, vorgesehen sind. Wieder ohne Garantie, dass es auch tatsächlich so kommt. Die für diese Version eingeplanten Funktionen sind:

- Async Main
- Source Generation
- Throw Expr
- Private protected
- Non-null Ref Types
- Better Betterness
- Records

■ With Exprs

■ Pattern Matching

Einige dieser Features befinden sich gerade im Zustand des Prototyping. Andere wiederum befinden sich erst in der Phase der Spezifikation.

Das Feature `Async Main` [9] ist beispielsweise ein kleineres aus dieser Liste. Aktuell ist es nämlich nicht möglich, `Main`-Methoden, die als Einstieg einer Anwendung dienen, mit dem `async` Schlüsselwort zu markieren. Werden darin allerdings andere asynchrone Methoden aufgerufen, muss an irgendeiner Stelle in der `Main`-Methode manuell dafür gesorgt werden, dass der Wechsel von `asynchron` zu `synchron` erfolgt. Das liegt in der Natur der Dinge. Eine Möglichkeit ist zum Beispiel der folgende Code, der einen Aufruf der `GetAwaiter` Methode enthält:

```
DoWork().GetAwaiter().GetResult();
```

Dieses Vorgehen ist aufwändig und fehleranfällig. Asynchrone `Main`-Methoden könnten dieses Problem beheben. Im Hintergrund muss der Compiler für dieses Feature etwas mehr Code generieren, da die `Common Language Runtime (CLR)` keine asynchronen Methoden als Einstieg einer Anwendung akzeptiert.

Mit auf der Liste der zukünftigen Features stehen ebenfalls die sogenannten `Records` [10]. Sie sind eine Mischung aus Klassen und Strukturen und ganz praktisch ausgedrückt lediglich eine Ansammlung von Eigenschaften. Denn es tritt im praktischen Einsatz des Öfteren auf, dass auch eine Klasse lediglich eine Sammlung von Eigenschaften ist, die mit Daten befüllt werden. Das Schreiben einer Klasse erfordert aber eine teilweise nicht unerhebliche Menge an Code. `Records` vereinfachen das, in dem die Syntax in C# deutlich gestrafft wurde und im Hintergrund zusätzlich notwendiger Code automatisch erzeugt wird. So wird aus dem folgenden Beispiel

```
class Point(int X, int Y, int Z);
```

eine vollständige Klasse vom Compiler erzeugt. Diese enthält nicht nur die Eigenschaften, einen sinnvollen Konstruktor mit den Parametern zum Belegen der Eigenschaften mit Daten, sondern auch Getter-Methoden, eine Implementierung von `IComparable<Point>`, sowie eine

Listing 3: Der "is" Operator im Pattern Matching

```
// Ohne Pattern Matching
var v = expr as Type;
if (v != null)
{
    ...
}

// Mit Pattern Matching
if (expr is Type v)
{
    ...
}
```

`Tostring` Methode, die alle Daten zurückliefert. Damit sind `Records` hervorragend geeignet, um Daten aufzunehmen und zu transportieren. Zum Beispiel in `Client-/Server`-Anwendungen, ohne dass viele reine Datenklassen manuell implementiert werden müssen.

Ein deutlich größeres Feature, das abschließend angesprochen werden soll, ist `Pattern Matching` [11]. Es ist eine der Funktionen, die mit am stärksten in der `Community` diskutiert werden und vormals in die nächste Version von C# Einzug halten sollte. Allerdings aufgrund des großen Diskussionsbedarfs und der vielen Stellen, an denen das Feature Implikationen auf die Sprache hat, verschoben wurde.

Zum Beispiel gehört zum `Pattern Matching` eine Erweiterung des „`is`“ Operators. Dieser wird so modifiziert, dass ein Test gegen ein Muster möglich ist. Beispielsweise unter `Zuhilfenahme` des `Typ-Patterns`. Damit lässt sich prüfen, ob ein Ausdruck von einem bestimmten Typ ist. Statt den „`as`“ Operator zu nutzen und anschließend auf `ungleich Null` zu prüfen, kann der „`is`“ Operator das in einer Zeile erledigen, wie das Beispiel in Listing 3 zeigt.

In beiden Fällen steht im `Scope` der `if`-Abfrage, also in den Klammern, die Variable `v` zur weiteren Verwendung zur Verfügung. Eine detaillierte Auflistung der `Pattern Matching` Features sind im bereits verlinkten Dokument [11] auf `GitHub` zu finden. Gerade dieses Feature ist aber mit Vorsicht zu genießen, da sich noch Änderungen ergeben können. `Pattern Matching` war einer der großen Punkte, die zunächst vollständig in C# 7.0 aufgehen sollten. Nach umfangreichen Diskussionen wurde es dann aber so gelöst, dass es ein `Pattern Matching „Light“`

in C# 7.0 geschaffen hat und der Rest in der Folgeversion nachgeschoben werden soll.

AKTUELLE BEDENKEN

Oftmals ist in Diskussionen bei `Twitter` und `Co.` herauszuhören, dass sowohl C# als auch `Visual Basic .NET` immer komplexer werden. Von der Sprache, konkret auf C# bezogen, wie sie im Jahr 2002 erstmals erschienen ist, ist mittlerweile nicht mehr viel übriggeblieben. Die Liste der zusätzlich eingeführten Sprachelemente ist lang.

Auf der einen Seite mag das abschreckend wirken. Viele fragen sich, ob sie immer noch C# lernen würden oder es Anfängern empfehlen sollen, die mit der Programmiersprache heute beginnen möchten. Es besteht das diffuse Gefühl, dass C# etwas von seiner Leichtigkeit verloren hat. Von der klaren Syntax und den überschaubaren Features. Manche sind auch einfach nur froh, die Sprache nicht von Grund auf erlernen zu müssen. Eben aufgrund der gefühlten oder begründeten, erhöhten Komplexität.

Auf der anderen Seite ist das allerdings ganz klar ein normaler Prozess. Programmiersprachen entwickeln sich. Neue Zielrichtungen und Anforderungen kommen hinzu, alte verschwinden oder wandeln sich mit der Zeit. Eine Sprache, die sich nicht wandelt und weiterentwickelt ist möglicherweise schon tot. Denn umso häufiger beziehungsweise intensiver eine Programmiersprache im Einsatz ist, desto mehr Anwendungsfälle sollen damit abgedeckt werden. Diese Anwendungsfälle sind es, die neue Anforderungen an die Sprache stellen. Das ist vergleichbar mit einer natürlichen Sprache. Auch diese entwickelt sich mit der Zeit und den Generationen, die sie sprechen.

Vielleicht ist es daher gar nicht so erstaunlich, dass C# und `Visual Basic .NET` mit der Zeit immer mehr Sprachelemente bekommen haben. Weit aus mehr Beachtung sollte dagegen die Tatsache bekommen, dass sich auch stark objektorientierte Sprachen wie C# immer weiter den funktionalen Vertretern annähern. Das gesamte Thema `Pattern Matching` ist nur eines der Beispiele, die stark von Sprachen wie F# beeinflusst worden sind. Hier stellt sich die Frage, ob die Zukunft in funktionalen Sprachen

liegt, wie weit sich objektorientierte Sprachen den funktionalen „Konkurrenten“ noch annähern mögen?

FAZIT

Denn diese Annäherung ist es letztlich, die darüber entscheidet, ob C#, um bei diesem Beispiel zu bleiben, für neue Projekte noch in Erwägung gezogen werden sollte? Müssen, gerade Neulinge, immer mehr funktionale Konzepte lernen, ist der Umstieg oder Einstieg mit Sprachen wie F# vielleicht von Anfang an einfacher. Vielleicht lohnt es sich irgendwann mehr, direkt mit einer funktionalen Sprache zu beginnen und später die fehlenden objektorientierten Konzepte „nachzulernen“, die im Vergleich zu C# und Co. fehlen? Aktuell ist das noch kein allzu verbreitetes Thema, da sich die Programmiersprachen noch hinreichend stark unterscheiden. Bei der aktuellen Entwicklung bleibt die Frage, wie lange das noch so sein wird.

Bis dahin können wir gespannt sein, welche Features es tatsächlich in C# 7.0 schaffen, ob sich noch größere Änderungen ergeben und was C# 8.0 für uns bereithält.

LINKS UND QUELLEN

- [1]: Neue Sprachfeatures in C# 6: <https://goo.gl/hK16j3>
- [2]: Release-Notes von Visual Studio „15“: <http://urlg.de/XMr>
- [3]: Das Roslyn Projekt auf GitHub: <http://urlg.de/XMs>
- [4]: Binary Literals in C# 7.0: <http://urlg.de/XMt>

- [5]: Digit Separators in C# 7.0: <http://urlg.de/XMu>
- [6]: Lokale Funktionen in C# 7.0: <http://urlg.de/XMv>
- [7]: Tuples in C# 7.0: <http://urlg.de/XMw>
- [8]: Übersicht des Features Status für C# 7.0: <http://urlg.de/XMx>
- [9]: Async Main in C# 7.0: <http://urlg.de/XMy>
- [10]: Records in C# 7.0: <http://urlg.de/XMz>
- [11]: Pattern Matching in C# 7.0: <https://goo.gl/JlkcZ3>

SharePoint konferenz

20. - 22. Februar 2017 in Erding
<http://www.sharepointkonferenz.de/2017>

Passt der Termin nicht?
Dann vielleicht ein **SharePoint Camp**.
<http://www.sharepointcamp.de/sharepointserver2013/>



Die nächsten Termine 2016:
07. - 11. November in Berlin
12. - 16. Dezember in München

CARTOON



DISPLAY TEMPLATES

Eigene SharePoint-Anzeigevorlagen erstellen

Autoren: Dr. Eckhard Hauenherm, Walter Saumweber

SharePoint stellt standardmäßig eine begrenzte Anzahl von Anzeigevorlagen zur Verfügung, die den eigenen Bedarf jedoch nicht immer abdecken. Wir zeigen Ihnen im Weiteren, wie Sie eigene Anzeigevorlagen erstellen und anpassen können.

Eigene Anzeigevorlagen zu definieren bietet sich z. B. dann an, wenn man weitere Eigenschaften von Elementen anzeigen lassen oder die Darstellung auf der Webseite anpassen möchte. Außerdem ist es möglich, benutzerdefinierte Eigenschaften von Elementen über die Suche der Anzeige hinzuzufügen. Mit der vollständigen Integration der FAST-Search in SharePoint 2013 besteht die Möglichkeit, Veröffentlichungs-Websites in SharePoint einzurichten,

deren Inhalte mittels einer gezielten Suchabfrage auf einer Seite zusammengestellt und über CSS-gesteuerte Webparts angezeigt werden. Mit der Funktion des Cross-Site-Publishings (Website-übergreifende Veröffentlichung) können die Inhalte in SharePoint-Standardlisten erstellt und verwaltet werden und über die Suchfunktion in anderen Websites, Websitesammlungen, Webanwendungen oder sogar SharePoint-Farmen auf Webseiten veröffentlicht werden.

Für die Anzeige der Inhalte wird das Webpart der Inhaltssuche (Content Search Webpart, CSWP) genutzt, das mit der Aktivierung der Publishing-features in einer Websitesammlung zur Verfügung gestellt wird. In dem Webpart wird einerseits die Abfrage definiert, die beim Aufruf der Seite an den Index gesendet wird, und zweitens die Anzeigevorlage (Display Template) ausgewählt, die für die Darstellung der Inhalte genutzt wird.

SO NUTZEN SIE DIE FUNKTION DES CONTENT SEARCH WEBPART

Gehen Sie folgendermaßen vor, um das CSWP auf einer Webseite einzurichten: Fügen Sie zunächst auf einer Seite innerhalb einer Veröffentlichungswebsite das Webpart für die Inhaltssuche in einen Webpartbereich ein. Das Webpart finden Sie unter der Kategorie Inhaltsrollup. [s. Abb. 1]

In den Webparteeinstellungen befinden sich drei zentrale Optionen, die das Verhalten und die Darstellung des Webparts steuern. Ganz oben finden Sie die Einstellungen für die Suche selbst. Über die Option *Abfrage ändern* kann hier definiert werden, in welcher Inhaltsquelle nach welchen Elementen anhand welcher

Eigenschaften gesucht wird. Dabei stehen ein Schnellmodus und ein erweiterter Modus des Abfrage-Editors (Query Builder) zur Verfügung. [s. Abb. 2]

Neben der eigentlichen Abfrage nach bestimmten Elementtypen können im Abfrage-Editor auch Einschränkungen, Sortierkriterien und Einstellungen für die Umschreibung von URLs für die Elemente definiert werden. [s. Abb. 3]

Die zweite Option in den Webparteeinstellungen bezieht sich auf die Auswahl der Anzeigevorlagen. Basierend auf der Auswahl des Steuerelements (Liste, Bildschirmpräsentation

oder Liste mit Seitennavigation) stehen verschiedene Standardanzeigevorlagen zur Verfügung. [s. Abb. 4]

Im dritten Abschnitt des Webparts besteht die Möglichkeit, die Eigenschaftenzuordnungen der Felder entsprechend der ausgewählten Anzeigevorlage zu ändern. [s. Abb. 5]

Über benutzerdefinierte verwaltete Eigenschaften (Managed Proper-

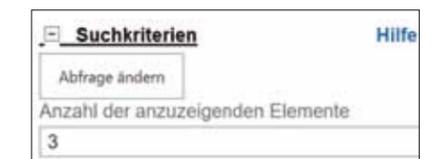


Abbildung 2: Aufruf des Abfrage-Editors

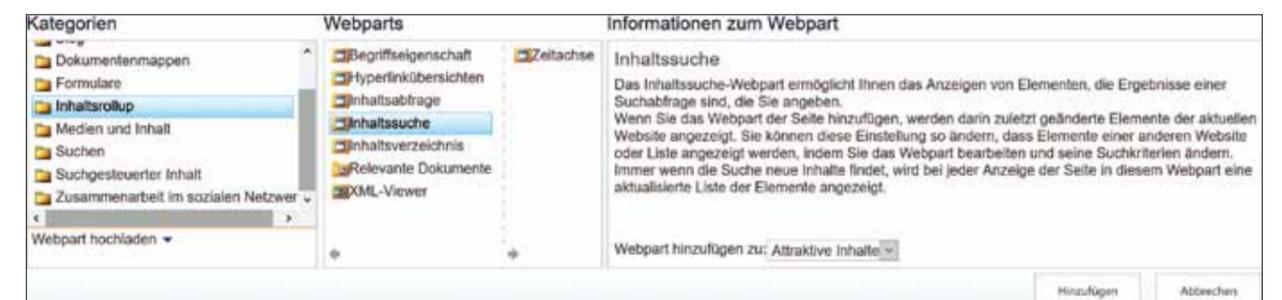


Abbildung 1: Inhaltssuche Webpart in der Webpart-Auswahl

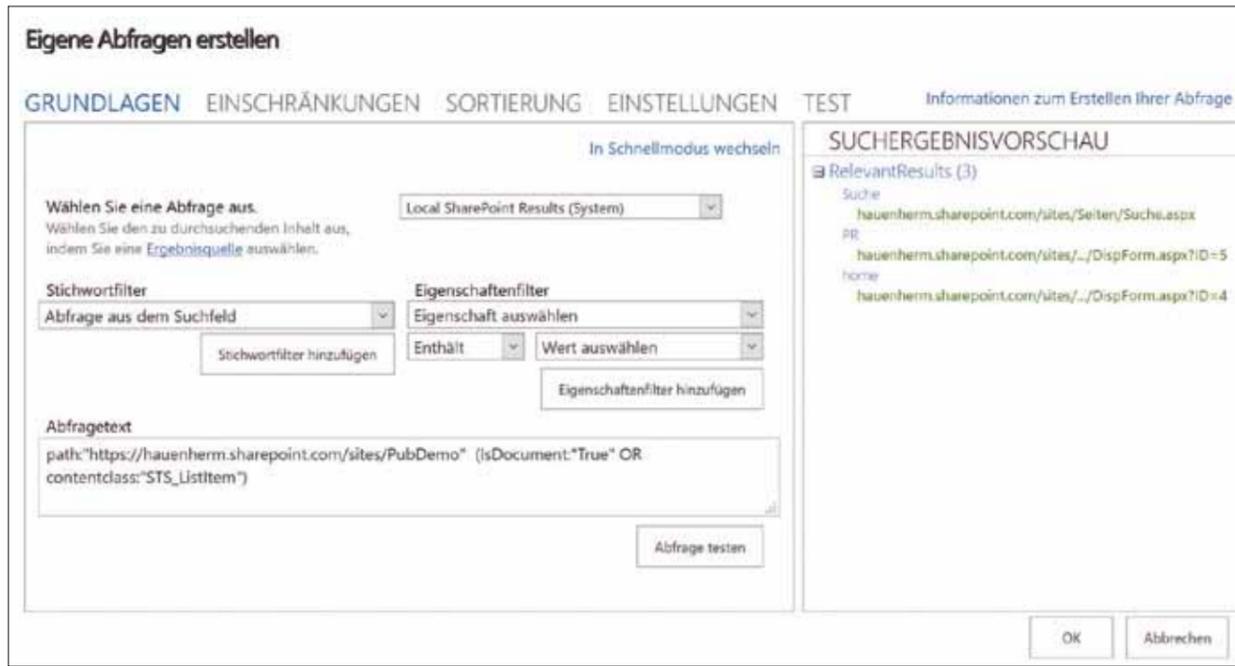


Abbildung 3: Erweiterter Modus des Abfrage-Editors

ties) im Suchschema der Suchfunktionen können auch selbstdefinierte Elementeigenschaften aus den Listenspalten der Elemente der Anzeige hinzugefügt werden.

DARSTELLUNG DER SUCH- ERGEBNISSE IM CSWP

Um eine eigene Anzeigevorlage zu erstellen, öffnet man am besten die Veröffentlichungswebsite mit dem SharePoint Designer 2013 (dieser lässt sich auch für SharePoint 2016 verwenden). Im SharePoint Designer navigiert man über den Dateieexplorer (ganz unten in der Navigation) in den Ordner `_catalogs/masterpage/Display Templates/Content Web Parts`. Dort werden alle zur Auswahl stehenden Anzeigevorlagen aufgeführt. [s. Abb. 6]

Für jede Anzeigevorlage existiert eine HTML-Datei (.html) und eine JavaScript-Datei (.js). Die JavaScript-Datei wird von SharePoint selbstständig erstellt und aktualisiert. Bearbeitet werden sollte immer nur die

HTML-Datei. Für die eigene Anzeigevorlage kopiert man am besten eine von den vorhandenen HTML-Dateien. Anschließend benennt man die kopierte Datei um und wartet, bis SharePoint die dazugehörige JavaScript-Datei erstellt hat. [s. Abb. 7]

Nun kann man die HTML-Datei über das Kontextmenü im erweiterten Modus öffnen, um den Code zu bearbeiten. Der Code der HTML-Datei stellt nicht direkt den Code der Anzeige dar, sondern steuert das Erstellen der Skriptdatei. Daher sind nur einige Teile der Datei zu bearbeiten. Innerhalb des Head-Tags sind mehrere Tags anzupassen. Zuerst wäre hier der Name der Anzeigevorlage zu nennen. Dieser wird im Title-Tag angegeben:

```
<title>Bild auf der linken Seite, 3 Zeilen rechts</title>
```

Im Tag `mso:MasterPageDescription` kann eine Beschreibung für die Vorlage angegeben werden:

```
<mso:MasterPageDescription msdt:dt="string">
```

Diese Elementanzeigevorlage zeigt ein Bild des Elements

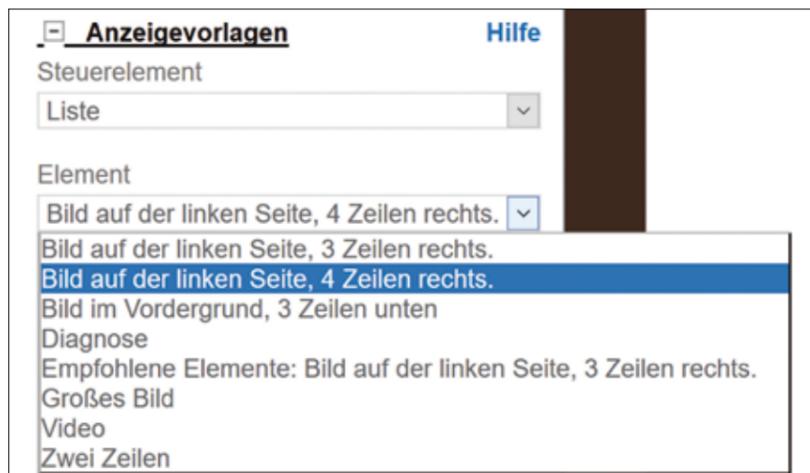


Abbildung 4: Auswahl der Anzeigevorlage



Abbildung 5: Ändern der Eigenschaftenzuordnung von Feldern

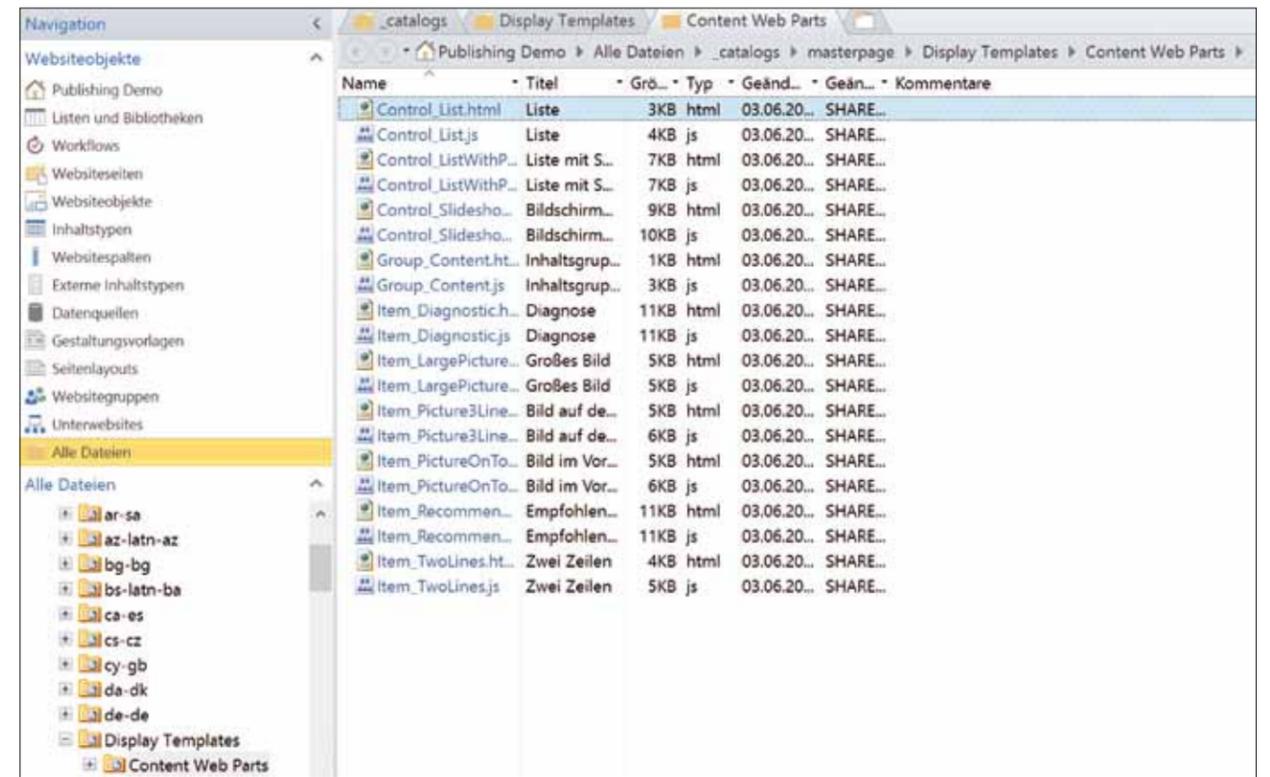


Abbildung 6: Auflistung der Anzeigevorlagen im SharePoint Designer

mit der Auflösung 100 x 100 auf der linken Seite an. Der Titel und die Standardelementbeschreibung werden rechts von dem Bild angezeigt mit einer zusätzlichen Zeile, die für eine benutzerdefinierte, verwaltete Eigenschaft zur Verfügung steht.

```
</mso:MasterPageDescription>
```

Die wichtigste Einstellung im Head ist die Definition der Standardfeldzuordnungen im Tag `mso:ManagedPropertyMapping`. Hier können Sie festlegen, welche Eigenschaften standardmäßig in die definierten Felder der Anzeigevorlage eingelesen werden. Änderungen an den Zuordnungen können Sie, wie oben beschrieben, hinterher in den Webparteeinstellungen vornehmen. Wenn Sie eine Anzeigevorlage mit zusätzlichen Feldern definieren, sollten Sie in diesem Tag aber die neuen Felder mit angeben.

```
<mso:ManagedPropertyMapping msdt:dt="string">PictureURL{Bild-URL}:'PublishingImage;PictureURL;PictureThumbnailURL','Link URL'{Link-URL}:'Path','Line 1'{Zeile 1}:'Title','Line 2'{Zeile 2}:'Description','Line 3'{Zeile 3}:'','SecondaryFileExtension','ContentTypeId'</mso:ManagedPropertyMapping>
```

Im Code klar zu erkennen ist, dass der englische Feldname in Hochkommata angegeben wird, die Feldbezeichnung für die Anzeige steht in geschweiften Klammern dahinter. Nach dem Doppelpunkt folgt eine – eventuell mehrwertige Liste – verwalteter Eigenschaften, getrennt durch Semikolon und in Hochkommata. Die anderen Bereiche des Head-Tags müssen in der Regel nicht bearbeitet werden. Innerhalb des Body können Sie nun die

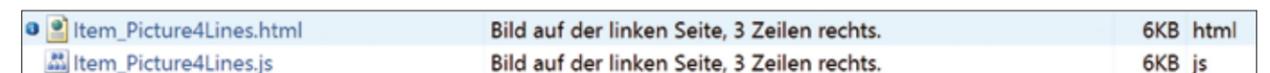


Abbildung 7: Die neue HTML-Datei mit der automatisch erstellten JavaScript-Datei

Felddefinition für Ihre Anzeigevorlage vornehmen. Dabei sind nur die Teile zu ändern, die innerhalb des `div`-Tags der Anzeigevorlage eingebunden sind. Darauf weist auch der Kommentar im Body hin:

```
Only the contents of the first <div> inside the <body> tag will be used while executing Display Template code.
```

Im Code werden die Anzeigezeilen über eine Reihe von Variablen definiert. Um eine zusätzliche Zeile zur Vorlage hinzuzufügen, müssen Sie einfach an den entsprechenden Stellen zusätzliche Variablen definieren:

```
var line4 = $getItemValue(ctx, „Line 4“);
line4.overrideValueRenderer($contentLineText);
var line4Id = encodedId + „,line4“;
```

Als Nächstes wird noch der Titel der Vorlage als Variable definiert. Diesen sollten Sie ebenfalls anpassen:

```
var dataDisplayTemplateTitle = „ItemPicture4Lines“;
```

Die Felddarstellung wird dann innerhalb des `Datacontainer`-Tags in Kommentaren angegeben. Der Code dient hinterher dazu, bei der Ausführung des Skripts den Daten CSS-Eigenschaften zuzuweisen und den Inhalt den Feldern zuzuordnen. Auch hier können Sie durch einfaches Kopieren und Anpassen die Definition der zusätzlichen Felder vornehmen:

```
if(!line4.isEmpty)
{
_#-->
<div class="cbs-pictureLine4 ms-textSmall ms-noWrap"
id="_{line4Id}#{line4Id}#{line4Id} title="_{line4Id}#{line4Id}#{line4Id}
defaultValueRenderer(line4) =#{line4Id}#{line4Id}#{line4Id}">
<!--#_
}
```

BEISPIELE FÜR EINFACHE ANPASSUNGEN

Im ersten Beispiel (Code_Zeile_2_mehrzeilig) werden basierend auf dem Standardtemplate Picture3Lines die CSS-Verweise für Zeile 2 gelöscht. Dadurch wird der Text aus dem eingelesenen Kommentarfeld nicht gekürzt und umbrochen.

```
<div title="_{#} $htmlEncode(line2.
defaultValueRenderer(line2)) =#{_} id="_{#} line2Id =#{_}
> _#{#} line2 =#{_}</div>
```

Beachten Sie, dass in den Codebeispielen der Domainname www.betcom.de gegebenenfalls zu ersetzen ist. [s. Abb. 8]

Das nächste Beispiel (Code_vierzeilig) passt das Standardtemplate so an, dass eine vierte Zeile wie oben beschrieben hinzugefügt wird. Die Bildzeile wird gelöscht, einige Formatierungseinstellungen im Code werden angepasst und es werden Feldbezeichner als statischer Text hinzugefügt:

```
<div class="cbs-Item" id="_{#} containerId =#{_} data-disp
laytemplate="Item4LinesWoP">
<div class="cbs-Detail" id="_{#} dataContainerId =#{_}
Seminarcode: <a class="cbs-Line1Link ms-noWrap ,,
href="_{#} linkURL =#{_} title="_{#} $htmlEncode(line1.
defaultValueRenderer(line1)) =#{_} id="_{#} line1LinkId
=#{_}"><b>_{#} line1 =#{_}</b></a>
<!--_{#}
if(!line2.isEmpty)
{
_{#-->
<div class="cbs-Line2 ms-noWrap" title="_{#}
$htmlEncode(line2.defaultValueRenderer(line2)) =#{_}
id="_{#} line2Id =#{_}">Ort: _#{#} line2 =#{_}</div>
<!--_{#}
}
if(!line3.isEmpty)
{
_{#-->
<div class="cbs-Line2 ms-noWrap" title="_{#}
$htmlEncode(line2.defaultValueRenderer(line2)) =#{_}
id="_{#} line3Id =#{_}">Beginn: _#{#} line3 =#{_}</div>
<!--_{#}
}
if(!line4.isEmpty)
{
_{#-->
<div class="cbs-Line2 ms-noWrap" title="_{#}
$htmlEncode(line2.defaultValueRenderer(line2)) =#{_}
id="_{#} line4Id =#{_}">Ende: _#{#} line4 =#{_}</div>
<!--_{#}
}
_{#-->
</div></div>
```

Das dritte Beispiel (Code_Standardicon) erstellt ein zweizeiliges Template, das anstelle eines dynamisch eingelesenen Dokument-Icons einen statischen Verweis als eine Art Aufzählungszeichen verwendet. Dazu wird die Variable für den Bildpfad entsprechend angepasst:

```
var iconURL = „https://www.betcom.de/SiteCollection
Images/Sonne.png“;
```



Exchange 2016 Administration
 Sie lernen alle Ebenen der Administration des Exchangeservers kennen, von der Installation über die Empfängerverwaltung bis hin zur Implementierung der Ausfallsicherheit und der Wiederherstellung. Auch Themen wie die Compliance-Sicherung und die Steuerung des Nachrichtenflusses werden detailliert behandelt

Abbildung 8:
Anzeigevorlage
für mehrzeilige
Textfelder

VERWALTETE EIGENSCHAFTEN ANPASSEN

Abschließend sei noch kurz das Verfahren zum Konfigurieren benutzerdefinierter verwalteter Eigenschaften angeführt. Die Definition eigener verwalteter Eigenschaften ist immer dann erforderlich, wenn in den Ergebnisfeldern des CSWP Daten aus Spaltenfeldern angezeigt werden sollen, die von SharePoint nicht standardmäßig einer verwalteten Eigenschaft zugeordnet sind. Verwaltete Eigenschaften dienen dazu, definierte Inhalte des Suchindex bestimmten Funktionen der Abfrage zur Verfügung zu stellen. Im ersten Schritt sind die Spalten in den Listen zu erstellen. Mindestens an einem Element sollte die Spalte auch einen Wert enthalten, damit sie als durchforstete Eigenschaft während des nächsten Crawls in den Index aufgenommen wird. Der Crawl der entsprechenden Inhaltsquelle muss einmal durchgelaufen sein, bevor die Eigenschaft unter den durchforsteten Eigenschaften aufgeführt wird. Entweder wartet man also einige Zeit oder stößt über die Suchdienstverwaltung in der Zentraladministration einen manuellen Crawl an. Anschließend kann die Spalte entweder einer neuen verwalteten Eigenschaft zugeordnet oder als eigene verwaltete Eigenschaft definiert werden. Die verwalteten Eigenschaften werden im Suchschema definiert, das auf der Ebene der Website, Websitesammlung oder des Suchdienstes bearbeitet werden kann. Arbeitet man mit anwendungsübergreifenden Veröffentlichungen, muss das Schema im Suchdienst entsprechend angepasst werden. Am besten überprüft man zuerst, ob für die gewünschte Spalte eine durchforstete Eigenschaft angelegt wurde. Dafür sucht man unter den durchforsteten Eigenschaften im Suchschema nach dem Namen der Spalte. In den Codebeispielen wurde eine Spalte *Seminarbeschreibung* angelegt. Nach einer Durchforstung (Crawl) der Inhaltsquelle wird diese Spalte als *ows_Seminarbeschreibung* in den durchforsteten Eigenschaften aufgeführt.

Unter den verwalteten Eigenschaften kann man über den gleichen Weg nach der Eigenschaft suchen, über die der Spalteninhalt in der Abfrage bereitgestellt werden soll. Klickt man nun auf den Namen der Eigenschaft, öffnen sich deren Einstellungen. Im Abschnitt *Zuordnung zu durchforsteten Eigenschaften* kann man nun über die Schaltfläche *Zuordnung hinzufügen* nach der eben geprüften durchforsteten Eigenschaft suchen, und diese der verwalteten Eigenschaft als Ergebnisquelle hinzufügen. Sind einer verwalteten Eigenschaft mehrere durchforstete Eigenschaften zugeordnet, kann man auswählen, ob die Inhalte aller Eigenschaften als Ergebnis der Suche zurückgeliefert werden sollen oder nur der Inhalt der ersten nicht leeren Eigenschaft. Anschließend kann die Eigenschaft wie oben beschrieben in den Einstellungen des CSWP einem Feld zugeordnet werden.

Magdeburger Developer Days



Die Community Konferenz für Entwickler

Am 10. Mai 2017 ist es wieder so weit: Dann startet die 2. Entwicklerkonferenz im KONGRESS & KULTURWERK – fichte in Magdeburg! Wir freuen uns über die zahlreichen hilfsbereiten Unterstützer, die es uns ermöglichen ein erstklassiges Programm mit renommierten Speakern rund um die Themen .NET, JAVA, HTML und ALM zusammenzustellen. Es werden bis zu 40 Sessions, vollgestopft mit wertvollen Erfahrungen und Praxiswissen von Entwicklern für Entwickler, bereitgestellt.



www.md-devdays.de info@md-devdays.de @MiB_MD_DevDays

Zusätzlich wird durch Aussteller ein Einblick in aktuelle Tools und Komponenten gewährt. Hier bieten sich optimale Möglichkeiten für Erfahrungsaustausch, Weiterbildung und Networking. Die Magdeburger Developer Days bieten ein umfassendes Themenspektrum aktueller Tendenzen zu OR Mappern, Datenbanken, Webtechnologien und Softwareentwicklungsprozessen. Erleben Sie die Entwickler-Community in Magdeburg als letzte Veranstaltung in diesem einmalig urbanen Ambiente.

10.-11.Mai 2017 - KULTURWERK fichte Magdeburg

.NET User Group Friedrichshafen

MI. 30.11.2016 – FABIAN FRASCH: FUNCTIONAL PROGRAMMING IN .NET

Schon seit einigen Jahren sind funktionale Programmiersprachen und Konzepte auf dem Vormarsch – auch in der .NET Welt. Die meisten werden auch schon Lambdas, LINQ und Async verwenden. Was dies mit funktionaler Programmierung zu tun hat, und welche Vorteile funktionale Programmierweise hat, welche Anwendungsgebiete es gibt und wie das Zusammenspiel mit objektorientierter Programmierweise funktioniert, wollen wir an diesem Abend betrachten.

MI. 14.12.2016 – CHRISTIAN PFISTERER: EIN SPIEL IN UNITY 3D

Moderne Game-Engines eignen sich nicht nur zur Entwicklung von Spielen für verschiedenste Plattformen, sondern kommen auch in Virtual- bzw. Mixed-Reality Szenarien zum Einsatz. Durch diese aktuellen Technologietrends kommt die Game Engine Unity auch vermehrt im industriellen Umfeld zum Einsatz. So werden holografische Inhalte für Microsofts HoloLens z.B. mit Unity

entwickelt. In diesem Vortrag erstellen wir ein erstes 3D-Spiel und lernen ganz nebenbei die IDE von Unity kennen.

WO UND WANN?

Die Vorträge finden von 18:30 – 21:30 statt. Veranstaltungsort ist im PRISMA Competence Park Friedrichshafen, Otto-Lilienthal-Straße 2, 88046 Friedrichshafen.

Mehr Informationen und die Möglichkeit dich anzumelden findest Du auf unserer Homepage unter www.dotnet-fn.de – Wir freuen uns auf dich





Foto: Tony Hegewald / pixelio.de

DEM ERFOLG AUF DER SPUR

Erfolg und Benutzerakzeptanz von Apps messen

Autoren: Dr. VEIKKO KRYPCZYK und OLENA BOCHKOR

Die meisten Apps werden produziert, um damit Geld zu verdienen. Den Erfolg kann man am Ende auch messen. Neben den direkten Einnahmen gibt es eine Reihe von weiteren Kennzahlen, welche als Erfolgsmaßstab herangezogen werden können. Im Artikel werden die Möglichkeiten einer Erfolgsmessung für ein App-Marketing vorgestellt. Aus den Ergebnissen können dann im zweiten Schritt Verbesserungsoptionen für die Folgeversion abgeleitet werden. Diese können die eigentliche App oder ihre Strategie zur Vermarktung betreffen.

Erfolgsmessung ist ein wichtiger Bestandteil des Tagesgeschäfts, denn nur durch regelmäßiges Monitoring kann auf positive und negative Entwicklungen rechtzeitig reagiert werden. Anhand von ausgewählten Kennzahlen kann man die App für die Zielgruppen mit dem höchsten Umsatzpotenzial optimieren. Dieses Vorgehen ist wichtig, um nachhaltig für wirtschaftlichen Erfolg zu sorgen. Instrumente wie beispielsweise iTunes-Connect und Google Developer Console sind dabei eine große

Hilfe. Mit diesen Tools können wichtige Kennzahlen des App-Marketings vielseitig ausgewertet werden.

KENNZAHLEN IM ÜBERBLICK

Die Messung des Erfolgs basiert auf der Auswertung wichtiger Kennzahlen. Um sich einen umfassenden Überblick zu verschaffen, ist es notwendig sich mit mehreren Kennzahlen zu beschäftigen. Dabei muss man wissen, dass dieses Vorhaben

durchaus zeitaufwändig ist. Der Aufwand kann sich jedoch lohnen, um wichtige Entscheidungen über die weitere Entwicklung der App zu treffen. Gerade Anbieter von mehreren Apps können auf diese Weise ein differenziertes Bild über ihre gesamte Produktpalette erlangen. Nur dann gelingt es, die Entwicklungsressourcen in der Zukunft in die richtige App zu investieren. Den Erfolg nur an einer Kennzahl, zum Beispiel den Umsatz festzumachen, greift definitiv zu kurz. Eine App kann durchaus den

„Nerv“ der Zeit treffen, aber bisher wurde vielleicht nur nicht die richtige Marketingstrategie angewendet oder die Nutzer haben sich nach anfänglicher Begeisterung wieder abgewendet. Letztes würde wahrscheinlich auf ein Missverhältnis von Erwartung und tatsächlicher Leistung beruhen, was zum Beispiel an mangelnder Qualität liegen kann. Kommen wir zunächst zu einigen markanten Kennzahlen:

■ **Umsatz:** Diese Größe ist wahrscheinlich eine der wichtigsten Kennzahlen zur Definition des wirtschaftlichen Erfolgs im App-Business. Umsätze können dabei auf mehreren Wegen generiert werden. Zum einem durch den Verkauf von kostenpflichtigen Apps und zum anderem durch den Verkauf von bestimmten Inhalten innerhalb der App.

■ **Anzahl der Installationen:** Eine wichtige Kenngröße, welche darüber Auskunft gibt, wie viele Nutzer in einem bestimmten Zeitraum die App heruntergeladen haben. Im Fall von kostenpflichtigen Apps ist die Berechnung ganz einfach: $\text{Umsatz} = \text{Preis} \times \text{Anzahl der Downloads}$. Wie lange und wie intensiv die App genutzt wird, ist kurzfristig ohne Bedeutung. Längerfristig wird sich jedoch die Zufriedenheit bzw. Unzufriedenheit am Markt durchsetzen und auf das Kaufverhalten für künftige potentielle

Kunden auswirken. Bei werbefinanzierten Apps sieht die Kalkulation anderes aus. Der wirtschaftliche Erfolg ist davon abhängig, dass die App gefällt und langfristig genutzt wird. Denn nur bei einer ausreichend hohen Nutzung werden aus der App Käufe getätigt und damit Umsätze generiert. In diesem Fall folgt nicht aus jedem Download auch gleich ein bestimmter Umsatz. Dennoch: Höhere Downloadzahlen sind ein Schritt in die richtige Richtung. Zwischen dem Zeitpunkt des Downloads und dem kostenpflichtigen Geschäft kann durchaus ein längerer Zeitraum liegen. Dabei bestimmt die Qualität des Inhaltes maßgeblich den wirtschaftlichen Erfolg. Ein typisches Beispiel ist eine App für eine digitale Zeitschrift. Die App wird meist kostenfrei angeboten. Je nachdem wieviel potentielle Kunden am Inhalt der elektronischen Zeitschrift Interesse haben, werden sie Käufe innerhalb der App auslösen.

■ **Anzahl der aktuellen Installationen:** Wir erhalten Auskunft darüber, auf wie vielen Geräten die App installiert ist und von wie vielen Benutzern diese noch benutzt wird. Diese Kennzahl ist insbesondere für Apps mit integrierter Werbung wichtig.

■ **Anzahl von Deinstallationen:** Dieser Wert ist sozusagen der Gegenpool

zur eben genannten Kennzahl. Hohe Werte drücken aus, dass die Nutzer trotz Installation die App nicht (nachhaltig) nutzen. Die Gründe dafür können ganz verschieden sein. Zu nennen sind: technische Ursachen, Änderung der Interessen oder die App entspricht nicht mehr den Erwartungen der Nutzer. Eine steigende Zahl von Deinstallationen sollte als ein Alarmsignal wahrgenommen werden.

■ **App Store-Aufrufe:** Es handelt sich um die Anzahl, wie oft die detaillierte Beschreibung der App im Store aufgerufen wurde. Diese Kennzahl gibt Auskunft darüber, wie oft die App von potenziellen Usern gesehen wurde. Damit erhält der Anbieter Feedback über die Reichweite der App im App Store. Eine niedrige Zahl spricht dafür, dass entweder die Keywords bei der Suche falsch sind oder der App-Name und/oder das App-Icon nicht ansprechend genug ausgewählt sind. Ein möglicher Grund kann auch sein, dass die App in externen Bewertungslisten und Empfehlungen nicht oder mit zu wenig Präsenz auftaucht. In diesem Fall werden die Besucher erst gar nicht auf die App aufmerksam gemacht. Anpassungen sind notwendig. Diese sollten schrittweise erfolgen, um die Ursache des anfänglichen Desinteresses zu ermitteln

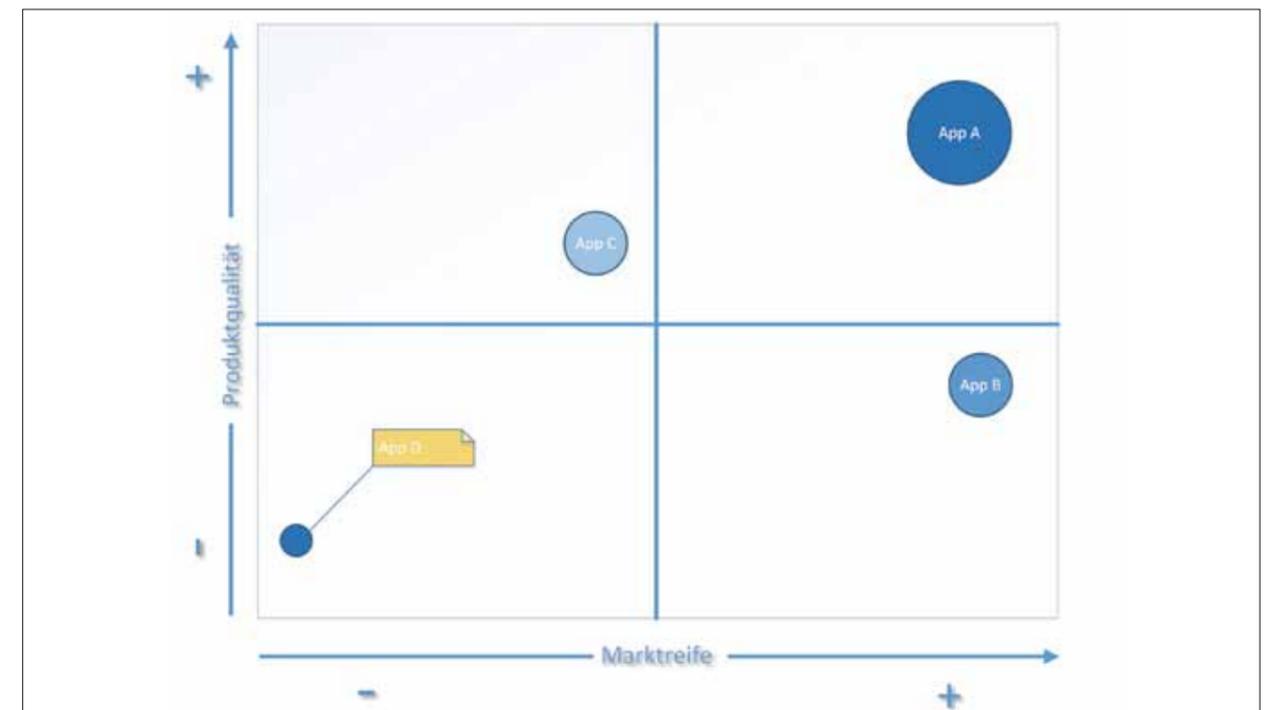


Abbildung 1: Portfolio-Methode zur Einschätzung der aktuellen und künftigen Erfolgsaussichten einer App.

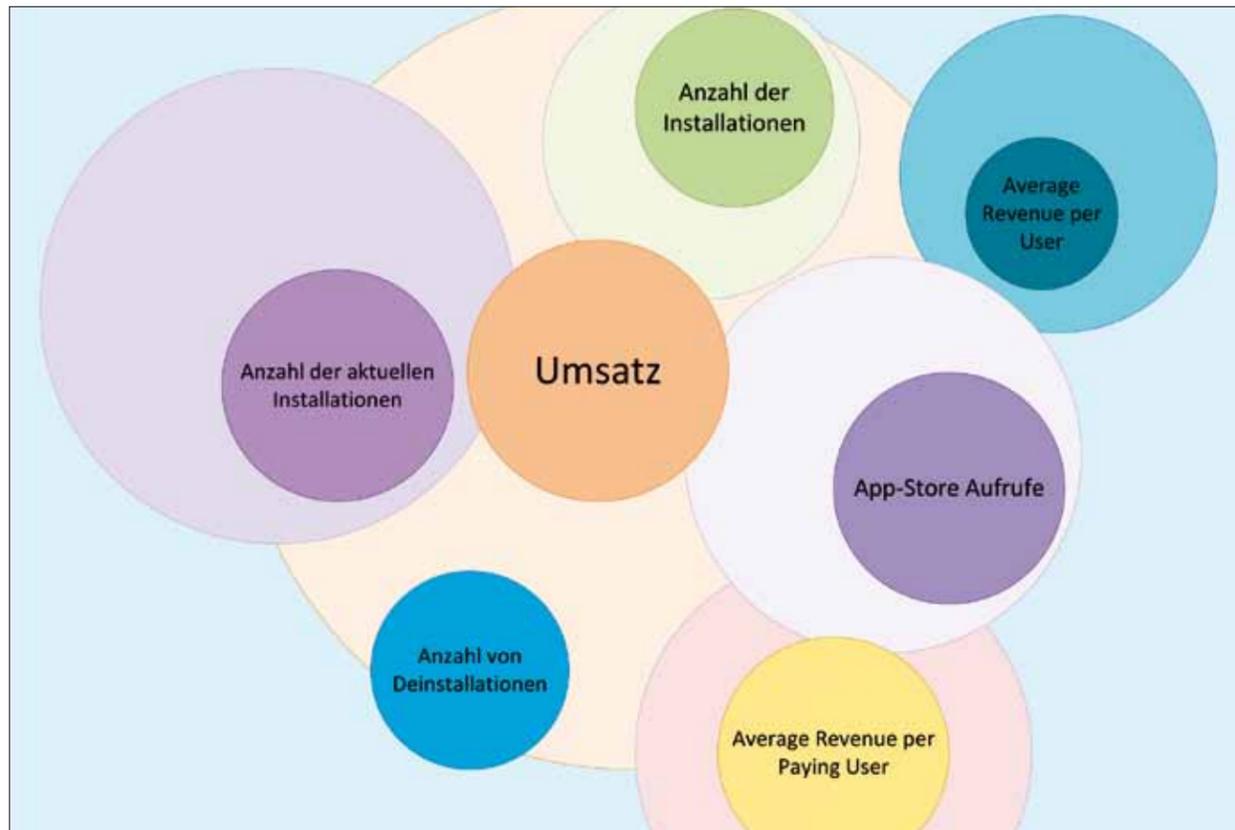


Abbildung 2: Diese Faktoren beeinflussen den Umsatz

und bei der nächsten App-Distribution diese Fehler zu vermeiden.

■ **Average Revenue per User (ARPU):** Eine Kennzahl für kostenlose Apps, in welchen mit virtuellen Gütern gehandelt wird, wichtig ist. Bei der Konzeption und der wirtschaftlichen Kalkulation der App nimmt man üblicherweise eine Prognose zu den *Einnahmen/User* vor. Ob diese Annahme der späteren Wirklichkeit entspricht, kann anhand dieser Kennzahl berechnet werden. Es gilt: $ARPU = \text{Anzahl der Installationen} / \text{erzielte Einnahmen}$. Falls der Wert niedriger als erwartet ausfällt, gibt es mehrere Möglichkeiten der Gegensteuerung. Ein erster Ansatz sollte es sein, die Anzahl der Installationen zu erhöhen bzw. die Deinstallationsrate zu reduzieren. Danach geht es um eine Optimierung der In-App-Käufe.

■ **Average Revenue per Paying User (ARPP):** Hiermit wird gemessen, welchen Betrag zahlende Kunden im Durchschnitt für die App ausgeben. Bei Apps mit virtuellen Gütern kann es schnell passieren, dass der Erfolg von einer sehr kleinen Benutzergruppe abhängt. ARPP wird wie folgt berechnet: $ARPP = \text{Umsatz} / \text{Anzahl der User}$ die mindestens einen Kauf durchgeführt haben.

Diese Kennzahlen sind alle numerischer Natur und können damit sehr gut ausgewertet werden. Neben einer solchen quantitativen Analyse, sollte man auch die verbalen Bewertungen der Nutzer auswerten. Da dieses nicht automatisiert geht, ist es aufwändig. Dennoch kann daraus eine qualitative Einschätzung zur Qualität der App und ggf. den damit verbundenen Content abgeleitet werden. Die Kommentare der Nutzer sind oft knapp, hart in der Formulierung, aber meist sehr nahe an der Wahrheit.

SCHLUSSFOLGERUNGEN

Grundsätzlich gibt es weitere Kennzahlen. Wichtig ist es jedoch sich auf die aussagekräftigsten Parameter zu stützen. Eine abschließende Beurteilung der App als einzelnes Produkt oder im Rahmen eines App-Portfolios kann nur dann vorgenommen werden, wenn man die erreichten Ergebnisse mit zuvor definierten Zielen abgleicht. Diese können völlig unterschiedlich aussehen. Beispielsweise kann das primäre Ziel der App die Kundenbindung für ein anders – durchaus nicht digitales – Produkt sein. Dazu wird die App kostenfrei angeboten. Innerhalb der App gibt es Möglichkeiten der Kontaktaufnahme mit dem Hersteller der Produkte. Diese Funktion wird intensiv durch die Nutzer angewendet. Eine andere Funktion ist das Auslösen von Bestellvorgängen. Ein kaum nachgefragtes Feature. Insgesamt kann die App mit guten Downloadzahlen überzeugen. Bei einer ausschließlichen betriebswirtschaftlichen Betrachtung der App stellt man fest, dass diese wirtschaftlich nicht erfolgreich ist. Der Grund ist die geringe Anzahl von In-App-Käufe. Das eigentliche Ziel der Kundenbindung wird jedoch sehr gut erreicht. In einer Folgeversion kann man sich ggf. auf diese Kernfunktion beschränken. Dabei tritt ein wichtiger Grundsatz zu Tage: Apps sind kleine Softwarelösungen mit einem stark eingeschränkten und fokussierten Funktionsumfang. Diese Kernfunktion muss die App jedoch in hervorragender Weise beherrschen.

Ein Hilfsmittel zur vergleichenden Beurteilung mehrerer Apps ist die Erstellung eines Portfolios. Dieses Instrument kennt man aus der Betriebswirtschaftslehre. Die betreffenden Produkte werden nach den Kriterien

relativer Marktanteil und Wachstum klassifiziert. Anhand der aufgeführten Kennzahlen erscheint beispielsweise eine folgende Bewertung sinnvoll:

■ **Marktreife:** Eine App wird nur Erfolg haben, wenn sie die Wünsche der Nutzer erfüllt. Mit anderen Worten: Die potentiellen Nutzer der App müssen ein hohes Interesse am Thema bzw. am Lösungsansatz haben. Das bedeutet noch nicht, dass die vorliegende App diese Wünsche tatsächlich erfüllt. Ein grundsätzliches Interesse äußert sich zum Beispiel darin, dass die App hohe Installationszahlen aufweist, die Detailseite sehr häufig besucht wird und unmittelbare Konkurrenzprodukte erfolgreich sind. Dieses Kriterium gibt an, dass ein Markt für das Thema der App vorhanden ist. Einschränkend muss man hinzufügen, dass für vollständig neuartige Ideen, sich erst ein Markt entwickeln bzw. durch Marketingmaßnahmen geschaffen werden muss.

■ **Produktqualität:** Trotz des grundsätzlichen Bedarfs kann eine App ohne Erfolg bleiben, wenn sie die Nutzer enttäuscht. Kennzahlen zur Messung der Produktqualität haben wir ebenso vorgestellt, zum Beispiel die Rate der Deinstallationen.

Wir verdichten also viele Einzelkennzahlen zu diesen zwei markanten Größen und stellen diese in Form einer Portfoliomatrix dar. Ein fiktives Beispiel ist in Abbildung 1 zu sehen. Dieses Portfolio gibt dann Anregungen, wie die weitere Strategie aussehen kann. Zwei Beispiele:

1. **Qualitätserhöhung:** Eine App deren Marktpotential als gut eingeschätzt wird, deren Qualität die Nutzer aber bisher mit einer konsequenten Deinstallation bestraft haben, muss als Projekt nicht verworfen werden. Um die App zum Erfolg zu führen ist jedoch die Qualität dringend zu erhöhen. Dieses kann zum Beispiel in einer deutlichen Überarbeitung der Benutzeroberfläche und der Usability bestehen.
2. **Strategische Entscheidung:** Apps deren Marktpotential als eher gering einzuschätzen ist, verbrauchen auf Dauer viele Ressourcen für die weitere Entwicklung. Eine Verbesserung der Qualität und des Funktionsumfangs werden kaum helfen. Die grundsätzliche Konzeption der App muss überprüft werden. Kann mit einer Neuausrichtung kein neuer Markt erschlossen werden, macht es wirtschaftlich keinen Sinn diese App weiter zu entwickeln. Frühzeitig sollte man aussteigen oder es als Hobby-Projekt für eine kleine Zielgruppe fortführen.

In einem Portfolio kann man eine dritte Dimension darstellen. Die Größe der Kreisfläche korreliert mit dem Umsatz der App.

ERFOLG BEEINFLUSSEN

Es gibt eine Reihe von Metriken, welche den Erfolg einer App beeinflussen. Wichtige Einflussfaktoren sind:

■ **Sprache und Land:** Darüber muss bereits bei der Konzeption einer App entschieden werden. Es ist immer empfehlenswert die Sprache zu wählen, welche ein Großteil der erwarteten Zielgruppe spricht. Anhand von Reports und Statistiken kann man sehen, welche Sprache am häufigsten auftaucht. Somit kann man darüber nachdenken, die App auch in weiteren Sprache anzubieten, d.h. eine

Lokalisierung durchzuführen. Einerseits kann es positives Feedback im Form von Rezensionen und Bewertungen auslösen, andererseits können durch die Lokalisierung auch vollständig neue Zielgruppe erreicht werden. Leider wird diese Metrik nur von Google (Developer Console) angeboten. Ebenso wie die Sprache, ist der Nutzungsort von großer Bedeutung. Sehr oft konzentrieren sich die Entwickler auf ihrem Heimatmarkt. Dabei wird leider zu oft verkannt, dass es durchaus möglich ist, dass in anderen Regionen die Nachfrage nach der App deutlich höher sein kann.

■ **Plattform und Geräte:** Die Verteilung auf Geräteklassen oder Geräte ist wichtig. Bei künftigen Marketingmaßnahmen können diese Erkenntnisse berücksichtigt werden. Auch mögliche technische Probleme können daraus abgeleitet werden, wenn die App auf einen bestimmten Gerätetyp nicht genutzt wird.

■ **Version und Absturzrate:** Es werden regelmäßig neue Versionen veröffentlicht. Es ist kaum noch möglich, die eigene App auf allen genutzten Versionen zu testen. Bei steigender Anzahl von Abstürzen oder Deinstallationen soll man die Lauffähigkeit der App auf den betreffenden Betriebssystemversionen untersuchen. Für die Weiterentwicklung ist auch wichtig zu wissen, auf welchen Versionen des Betriebssystems die Installation erfolgt. Bestimmte technische Funktionalitäten werden erst ab einer bestimmten Version angeboten. Somit ergibt sich folgendes Dilemma: Einsatz neuer Technologie oder möglicher Wegfall einer Nutzergruppe. Für eine bessere Abschätzung des Risikos können Versionsstatistiken helfen. Abstürze sind generell sehr negativ behaftet und werden durch die Nutzer im App-Business kaum toleriert. Es ist empfehlenswert die notwendige Zeit in das Testing und Debugging zu investieren, um die App auf möglichst vielen Geräten zu probieren.

FAZIT UND AUSBLICK

Mit der Bereitstellung der App in den Store ist das Projekt nicht abgeschlossen. Ebenso professionell sollten die Ergebnisse der Nutzung ermittelt werden. Daraus können wichtige Schlussfolgerungen abgeleitet werden. Aus Kritik und dem Nutzerverhalten ist zu festzustellen, welche Stellen der Folgeversion am meisten einer Verbesserung bedürfen. Nicht selten gehen dabei die Einschätzungen des Entwicklungsteams und der Nutzer auseinander. Wir Softwareentwickler neigen dazu – eine Art Berufskrankheit – mit jeder neuen Version auch neue Funktionalität bereitzustellen. Nutzer haben dagegen oft ganz andere Ansprüche, beispielsweise eine Überarbeitung der Usability. Ökonomische Auswertungen zielen darauf ab, die Ressourcen des Entwicklungsteams in die richtigen Apps zu investieren. Eine Fokussierung bedeutet eben auch, dass man sich von weniger erfolgreichen Projekten trennen muss. Nur auf diese Weise hat man die Chance ein Produkt dauerhaft am Markt mit wirtschaftlichen Erfolg zu platzieren.

LITERATUR UND LINKS

- [1] Mroz, R.: App-Marketing für iPhone und Android, mitp, 2016



Bild: David C. Thömmes (privat)

HOLOLENS GETESTET

Ein Blick in die Zukunft Review

Autor: DAVID C. THÖMMES

Auf der letzten Developer Week gab es die Möglichkeit am Stand der AIT die erste Generation der Microsoft HoloLens zu testen. Die Microsoft HoloLens ist eine Augmented Reality Brille, welche hochauflösende 3D-Projektionen vor dem Auge des Betrachters darstellt. Als Nutzer schaut man durch ein transparentes Visier und hat die Möglichkeit per Gesten-, Sprach-, Kopf- sowie Augenbewegung mit den Projektionen zu interagieren. Dabei verschwindet nicht die Realität, sondern diese wird lediglich um die projizierten Objekte erweitert.

LIVE

Mit wenigen Handgriffen sitzt die HoloLens sehr komfortabel auf dem Kopf. Mittels Einstellungsradchen,

ähnlich wie bei einem Fahrradhelm, kann man die Größe verändern. Die gesamte HoloLens fühlte sich wertig und stabil an. Die erste Orientierung ist meist etwas ungewohnt, doch wenige Momente später kann man die ersten Projektionen in einer atemberaubenden Qualität erkennen. Und mit jeder Kopf- oder Körperbewegung mehr werden die Überlagerung der Realität immer angenehmer und „fühlbare“. Selbst wenn hin und wieder ein Messebesucher durch eine Projektion läuft, welche man gerade betrachtete, stört dies nicht sonderlich beziehungsweise schränkt es das Erlebnis nicht ein.

Speziell für die Messebesucher waren in der Umgebung einige Projektionen versteckt, welche erst durch das Explorieren des Areal

sichtbar wurden. Dies führte unter den Testpersonen zu regelrechten Schnitzeljagden.

Nach den ersten Erkundungen folgte ein Test der Interaktion mit Windows 10 beziehungsweise dem Startmenü. Über eine entsprechende Fingergeste ist man in der Lage das Startmenü zu öffnen. Wenn man diese Geste jedoch nicht kennt, ist man auf eine Hilfestellung angewiesen. Zur Interaktion und als Feedback für den Nutzer zeigt die HoloLens einen Cursor an, welcher anhand der eigenen Kopfbewegung gesteuert werden kann. Mittels weiterer Fingergesten kann eine Interaktion mit einem fokussierten Objekt (Cursor befand sich über dem Objekt) getätigt werden. Um das neu erlernte Wissen zu nutzen und zu testen, kann man die

App - Galaxy Explorer erforschen. Hier warten weitere beeindruckende Erfahrung auf den Träger der HoloLens.

AUSBLICK

Während des Tests fiel das eher eingeschränkte Blickfeld auf. Je nach Blickwinkel und Größe der Objekte, können die Objekte an den Rändern der Projektionsfläche abgeschnitten werden. Für die Zukunft ist es wünschenswert, wenn das Blickfeld deutlich größer ausfällt. Weiterhin scheint die Gestenerkennung zu diesem Zeitpunkt noch nicht immer optimal zu funktionieren. Es benötigt einige Anläufe, bis man die korrekten Distanzen zwischen Kopf, HoloLens und Hand zur Interaktion sowie Manipulation der Objekte gefunden hat. Mit etwas Übung ist ein Navigieren meistens möglich. Ein Hilfesystem oder On-Boarding wäre für unerfahrene Nutzer bei der Erstnutzung empfehlenswert. Zweifellos ist die HoloLens eine sehr interessante und vielversprechende Innovation. Für den aktuellen Status der Entwicklung sind die genannten Abstriche verkraftbar. Die HoloLens bietet einen faszinierenden



Bild: Microsoft

Blick in die Zukunft und weckt die Lust auf mehr. Gerade im Enterprise- sowie Industriekontext kann man das System als sinnvolles Assistenzsysteme z.B. für Maschinenbediener einsetzen. Prinzipiell eröffnet die HoloLens UX Designern die Möglichkeit auf bequeme Art und Weise die Realität mit sinnvollen Metainformationen zu erweitern. Darstellungen komplexer CAD-Modelle, Fehlerhinweise an Maschinen oder vielleicht herumfliegende Code-Schnipsel für Visual Studio – vieles ist in diesen Tagen denkbar! Designvisionen die manch einer bereits bei der Google Glass hatte, scheinen heute immer greifbarer zu werden.

We ♥ IT!
And you?



Trainer

The WOW
of further education

Begeisterung & Leidenschaft

Training & Consulting

Mediendesign & Event

Marketing, HR & Controlling



Marketing



Wir suchen Dich!

ppedv AG, Marktler Straße 15b, 84489 Burghausen, karriere@ppedv.de



Foto: Tim Reckmann / pixelio.de

TaxoRecommend:

Multi-Channel B2B Präferenzanalyse System auf Basis der MS Dynamics CRM Datenstruktur und multiplen Taxonomien

Erstautor: HEIKO ANGERMANN, Zweitautor: NAEEM RAMZAN

Der Online-Vertriebshandel setzt mittlerweile immer häufiger auf multiple *On-* und *OffSite* Kanäle, genannt *Multi-Channel* (auch *Cross-Channel* oder *Omni-Channel*, z.B. *Social Media*, *Web-Shop*, *E-Mail*, etc.). Dieser Ansatz verspricht einerseits höhere Absatzchancen und ermöglicht ein breiteres Spektrum an Kundenwissen. Andererseits werden die dafür benötigten Infrastrukturen stets komplexer und die Nutzung der Kundendaten ist für viele Firmen noch immer unklar. Das in dieser Arbeit vorgestellte Präferenzanalyse-System **TaxoRecommend** nutzt die Vorteile des neuartigen Online-Handels und überwindet dabei die genannten Nachteile. **TaxoRecommend** basiert auf der *Structured Query Language* (SQL) und nutzt die Datenstruktur von Microsoft Dynamics CRM (*Customer Relationship Management*) sowie die darin bereitgestellten taxonomischen Strukturen. Hierdurch kann

das System Kundenwissen über verschiedenste Kanäle aggregieren und die Präferenzen auf unterschiedlich skalierbaren Objekt- und Zielgruppenebenen vorhersagen.

1. EINFÜHRUNG

Heutzutage ist der (Online)-Vertrieb vielfältiger und fragmentierter als jemals zuvor [1]. Die Produkte werden nun nicht mehr nur über einen Kanal beworben, sondern der Vertrieb gestaltet sich mittels des Zusammenspiels von multiplen und heterogenen Vertriebs- und Marketingkanälen, genannt *Multi-Channel*. Die Klassifizierung der unterschiedlichen Kanäle erfolgt hierbei je nachdem ob die Maßnahme innerhalb (*OnSite*) oder außerhalb der direkten Verkaufsplattform (*OffSite*) stattfinden. Dementsprechend spielen die verschiedenen Kanäle zusammen, so dass Kunden den Versandhandel

aufgrund verschiedenster *On-* und *OffSite* Maßnahmen und mittels unterschiedlichster Kanäle und dadurch auch Geräte betreten. Diese Art des Vertriebes führte in den letzten Jahren dazu, dass der E-Commerce Handel stark an Bedeutung gewonnen hat und Händler hierdurch die Chance haben, ein noch größeres Wissen über Kunden zu sammeln und schließlich jenes akquirierte Wissen für neue Marketingmaßnahmen gewinnbringend nutzen zu können, oft bezeichnet als „*Big Data*“.

Für insbesondere kleine und mittlere Unternehmen stellt die Datenflut sowie eine effiziente Nutzung dieser, jedoch noch immer eine große Herausforderung dar. Für diese Unternehmensgruppe stellt der hier vorliegende Artikel ein Präferenzanalyse-System vor, genannt **TaxoRecommend**, welches die Aufgabe hat Wissen zu aggregieren, Wissen zu analysieren sowie das gewonnene

Wissen automatisiert bereitzustellen. Das System basiert hierbei auf der Datenstruktur von Microsoft Dynamics CRM (*Customer Relationship Management*) um Kundenwissen effektiv zusammenzufassen. Hierdurch ist das System in die bestehende Systemlandschaft einbindbar und kann die Präferenzanalyse nach verschiedenen Objekt- und Zielgruppenebenen skalieren.

Präferenzanalyse-Systeme (eng. *Recommender Systems*) nutzen verschiedene Datenarten um Wissen über den Kunden zu analysieren [3]. Dieses kann dann beispielsweise genutzt werden, um personalisierte Marketingmaßnahmen zu generieren, z.B. mittels Empfehlungssystemen (eng. *Expert Systems*) [1b]. Die in *Recommender Systems* genutzten Algorithmen greifen hierbei je nach Komplexitätsgrad auf strukturierte, semi strukturierte, und/oder nicht strukturierte Daten zurück. Vor der Implementierung eines *Recommender Systems* sollten drei wesentliche Zielaspekte geklärt werden:

a) Welchen Nutzen kann die Datenflut für unser Unternehmen bewirken?

Daten über Kunden erlauben mehr über unsere Käuferschicht zu erfahren. Insbesondere werden in Erfahrung gebracht, wer kauft was, wer kauft dies wann, warum kauft er dies und wird er dies wieder kaufen wollen.

b) Welche Mechanismen für die Analyse der Daten stehen bereit?

Die Mechanismen (Algorithmen) welche bereitstehen sind vielfältig und dem Bereich des Data Mining und Machine Learning zuzordnen. Im Regelfall basieren diese Mechanismen auf in Wissenschaft und Forschung etablierten Algorithmen (Beispiele und Übersicht in [5]). Diese bewerten die Ähnlichkeiten innerhalb einer Menge oder zwischen mehreren Mengen, z.B. von Warenkörben, Suchanfragen, oder Bewertungen. Ein performantes Zusammenspiel mehrerer verschiedener Algorithmenarten mit unterschiedlicher Gewichtung ist das Hauptkriterium für präzise *Recommender Systems*. Die Kombination bewirkt, dass bestimmte Merkmale nicht überbewertet werden (eng. *Over Fitting*). Zum einen verspricht diese Aggregation eine Steigerung der Vorhersagegenauigkeit.

Zum anderen verspricht sie, dass das *Recommender System* mit verschiedenen Anwendungsfällen ausgewogener umgehen kann.

c) Welche direkten Maßnahmen können aus dem neuen System resultieren?

Basierend auf den gewonnenen Präferenzeinsichten können personalisierte *OnSite* und *OffSite* Marketingmaßnahmen erfolgen und die Maßnahmen können wiederum erneut strukturierte, semi strukturierte und nicht strukturierte Dokumente betreffen. *OnSite* Maßnahmen wären beispielsweise das Ausblenden von Produktkategorien, welche von bestimmten Kundengruppen nicht benötigt werden, die Optimierung der Suchmaschine durch ein höheres Ranking von präferierten Produkten, oder das Überarbeiten von Produkttexten. *OffSite* Maßnahmen können personalisierte Publikationen betreffen, z.B. *Postings* oder *Tweets* aus sozialen Plattformen, oder personalisierte E-Mail Kampagnen.

2. TAXORECOMMEND

Die im obigen Abschnitt genannten Zielaspekte haben ergänzend einen Kernaspekt gemeinsam. Nämlich, auf welchem Datenbestand soll das Präferenzsystem agieren, bzw. in welcher Systemlandschaft soll sich das System einmünden. Für kleine Online-Händler ist in der Theorie

klar, das System soll am besten direkt im *Backend* (deu. Administrationsebene) des Shop-Systems agieren. In der Praxis und vor allem für exportorientierte Händler sieht dies jedoch anders aus. Oftmals wird eine Interaktion von verteilten Systemen genutzt. Beispielsweise ein *Entity Relationship Management System* (ERP) wird für die Verwaltung von Lagerbeständen, und ein *Product Information Management System* (PIM) für die Verwaltung der produktrelevanten Informationen, z.B. Kategoriezuordnung. Die Shop-Plattform selbst kann die genannten Funktionalitäten eben nur in rudimentärer Form abbilden oder gar nicht [6]. Neben allen weiteren Systemen reit sich in aller Regel zusätzlich ein CRM System. Dieses ermöglicht beispielsweise im Multi-Channel Kontext, die Kundenaktionen plattformübergreifend zu sammeln. Beispielsweise können aus dem Druckkatalog getätigte Bestellungen genauso im CRM abgebildet werden wie Einkäufe aus E-Mail Bestellungen oder Bestellungen, die direkt aus dem Shop-System eingehen.

2.1 TAXORECOMMEND SYSTEMLANDSCHAFT, DATENSTRUKTUR UND SKALIERBARKEIT

Laut den Gartner Magic Quadranten ist neben Salesforce, Microsoft Dynamics CRM eines der technisch führenden CRM Systeme, auch hinsichtlich „*Big Data*“ [7, 8]. Beispielsweise

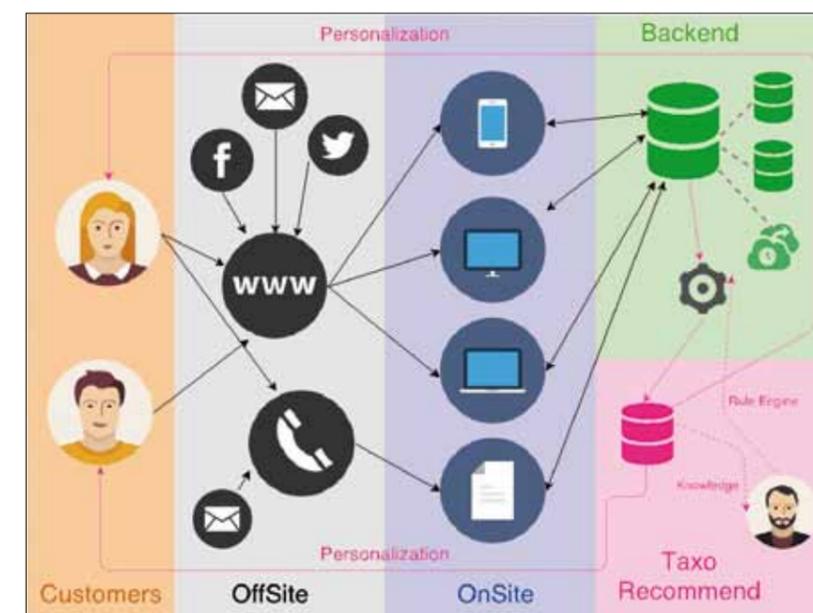


Abbildung 1: Traditionelle Architektur heutiger On-/Offline Händler (orange, grau, blau, grün) und Architektur mit dem Präferenzanalyse System TaxoRecommend (pink).

System	Tabelle	Spalte	TaxoRecommend
CRM	Firma	Firma.Name	CRM.Company
CRM	Firma	Firma.ID	CRM.Company
CRM	Firma	Branchencode1	CRM.Sector.Name
CRM	Firma	Branchencode1 Text	CRM.Sector.ID
CRM	Firma	Branchencode2	CRM.Subsector.Name
CRM	Firma	Branchencode2 Text	CRM.Subsector.ID
CRM	Ansprechpartner	Firma.ID	-
CRM	Ansprechpartner	Ansprechpartner.ID	CRM.Person.ID
CRM	Tools	Ansprechpartner.ID	-
CRM	Tools	Referenznummer.ID	CRM.Order.Header, CRM.Order.Detail
CRM	Tools	Grundgerätenummer	PIM.Product.ID
CRM	Tools	MTYP (Gerätname)	PIM.Product.Name
PIM	Klasse	Klasse.ID	PIM.Class.ID
PIM	Klasse	Klassenname	PIM.Class.Name
PIM	Gruppe	Gruppen.ID	PIM.Group.ID
PIM	Gruppe	Gruppen.Name	PIM.Group.Name
BKL	Abhängigkeit	Abhaengigkeit.ID	BKK.Dependency.ID
BKL	Abhängigkeit	Abhaengigkeit.Name	BKK.Dependency.Name

Tabelle 1: Microsoft Dynamics CRM Datenstruktur gematched mit der Datenstruktur von TaxoRecommend

durch die Integration von Social Engagement wird es nun möglich, innerhalb sozialer Plattformen die Stimmung zu Produkten und Marken basierend auf dem Zusammenspiel von nicht und semi strukturierten Daten erfahren zu können. Jene und bewährte Funktionalitäten in Microsoft Dynamics CRM haben alle das Ziel, die sogenannte Customer Experience (deu. Kundenerfahrung) zu steigern.

TaxoRecommend kann durch das Matching (deu. Abgleich), siehe Tabelle 1, mit der Microsoft Dynamics CRM Datenstruktur in die bestehende

Multi-Tier Systemlandschaft eingebunden werden, siehe Abbildung 1. Die vollständige Architektur umfasst fünf Komponenten mit unterschiedlichen Akteuren:

■ **Customers:** Diese Komponente umfasst den Akteur Kunde. In einem Business to Business (B2B) Use-Case werden 1-N Person einem Kunden (Firma) zugeordnet.

■ **OffSite:** Die Komponente *OffSite* umfasst alle Akteure (Kanäle), in welchen der Kunde nicht direkt eine Bestellung tätigen kann.

■ **OnSite:** Die Komponente und der Akteur *OnSite*, genauer der

(Web)-Shop selbst und dessen Benutzeroberfläche, erlauben es, direkt eine Bestellung auszulösen.

■ **Backend:** Die Komponente Backend enthält die Verwaltungsebene des Shops. Der *Backend*-Bereich ist unterschiedlich geprägt, je nach Systemlandschaft.

■ **TaxoRecommend:** Die Komponente *TaxoRecommend* zur Präferenzanalyse besteht aus einer Wissensbasis sowie einer Regeleinheit (eng. Rule Engine). Innerhalb der Regeleinheit kann der Administrator definieren, für welche Zielgruppenebene (z.B. einzelner Besteller, Firma, Branche, etc.) und auf welcher Objektebene (z.B. Produkte, Produktkategorien, etc.) die Analyse stattfinden soll. Das Ergebnis der Analyse wird dem Administrator ersichtlich, welcher hieraus für den Kunden personalisierte On- und OffSite Maßnahmen für Marketing und Vertrieb ansteuern kann.

Konkret werden für die Anbindung von *TaxoRecommend* an Microsoft Dynamics CRM drei Tabellen aus Microsoft Dynamics CRM benötigt (reduziert auf genutzte Spalten): Firma, Ansprechpartner und Tools.

■ Die Tabelle *Firma* beinhaltet alle Daten über den Kunden, bzw. die Firma.

■ Die Tabelle *Ansprechpartner* wird insbesondere für den B2B Anwendungsfall benötigt. Hier werden N

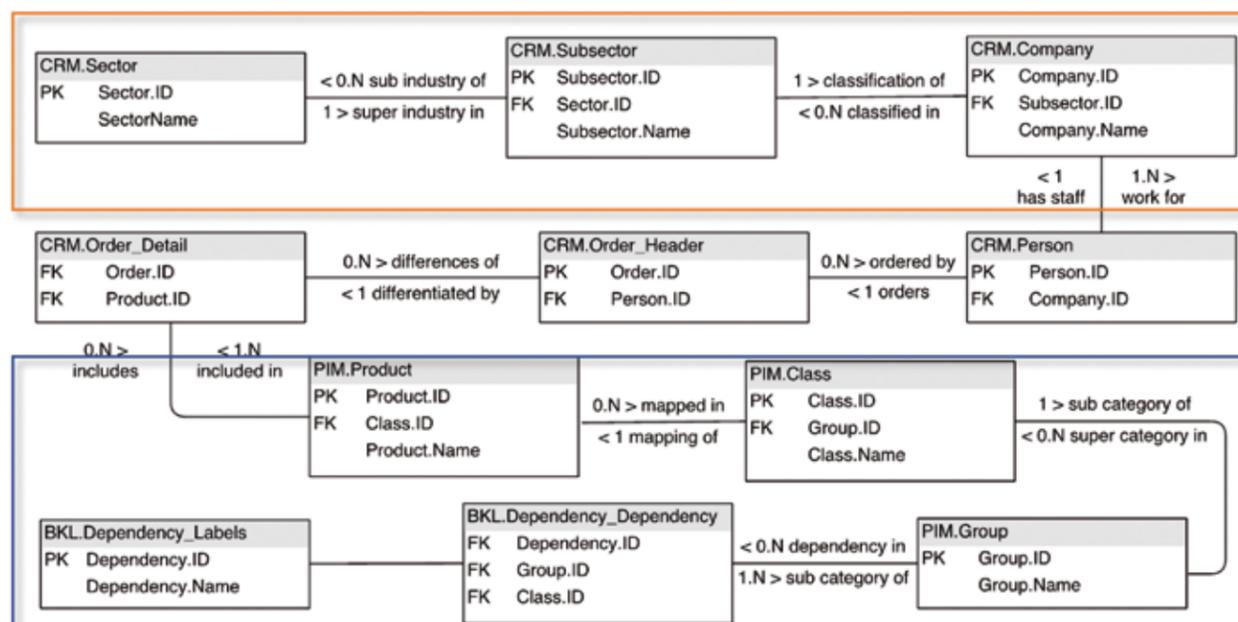


Abbildung 2: TaxoRecommend Entity Relationship Model und die darin enthaltenen Tabellen zur Abbildung mehrdimensionaler Taxonomien (orange = CRM Taxonomie, blau = PIM Taxonomie).

Personen einer einzelnen Firma zugeordnet.

■ Die Tabelle *Tools* enthält die Bestellungen von Produkten.

Für eine Präferenzanalyse auf Ebene von Produktkategorien wird ergänzend eine Produkttaxonomie benötigt, welche i. d. R. mittels des Shop-Systems selbst oder eines PIM Systems bereitgestellt werden kann. Soll die Analyse ähnliche Kategorien vereinen, kann dies mit Hilfe von Abhängigkeiten erfolgen. Das in der Tabelle dargestellte System BKL (*Background Knowledge*) stellt die beschriebenen Abhängigkeiten dar. Hier wird definiert, welche Klassen innerhalb welcher Gruppen ähnlich sind und demnach zu derselben Abhängigkeit gehören. Die finale Datenstruktur von TaxoRecommend wird mittels eines Entity-Relationship Models in Abbildung 2 skizziert.

Durch die Verwendung von zwei verschiedenen Taxonomien innerhalb von TaxoRecommend entsteht eine Skalierbarkeit hinsichtlich Objekt- und Zielgruppenebene, illustriert in Abbildung 3:

■ **CRM-Taxonomy:** Die CRM-Taxonomie klassifiziert Kunden hinsichtlich Ihrer Branche. Ein Subsector ist dabei die genaueste Ausprägung (z.B. Bäckerei), wobei ein Sector eine stärker allgemeine Ausprägung ist (z.B. Lebensmittelgewerbe). Jeder Kunde (Firma) wird bei der Migration innerhalb Microsoft Dynamics CRM einem Subsector zugewiesen, mit Hilfe dessen auf den Sector geschlossen werden kann. Hierdurch resultiert hinsichtlich der Präferenzanalyse eine Skalierbarkeit der zu analysierenden Zielgruppe. Somit kann die Analyse auf Ebene des einzelnen Kunden erfolgen, auf Ebene des Subsectors, oder auf Ebene des Sectors. Ferner

kann die Analyse auf den B2C Anwendungsfall umgeändert werden, respektive auf einzelne Besteller einer Firma, da das Mapping zu Personen eine weitere mögliche Skalierebene darstellt.

■ **PIM-Taxonomy:** Die PIM-Taxonomie klassifiziert Produkte anhand der Produktmerkmale. Hierbei ist eine Class die am meisten spezifische Ausprägung (z.B. Akkuschauber) und eine Group eine mehr generelle Ausprägung (z.B. Schrauber). Analog der CRM-Taxonomie entsteht hieraus eine erneute Skalierbarkeit. Die Präferenzen können auf Produktebene, Produktklassenebene und Produktgruppenebene durchgeführt werden. Ferner kann hierdurch die Präferenz für das jeweilige Objekt auch auf Abhängigkeitsebene durchgeführt werden. Jene berichtet dem Administrator dann, welche Zielgruppe für jene Klasse eine Präferenz hat und welche Klassen dieser Klasse ähnlich sind.

2.2 TAXORECOMMEND ZUSAMMENSPIEL VERSCHIEDENER ALGORITHMEN

Der in TaxoRecommend verwendete Mechanismus zur Präferenzanalyse basiert auf fünf Schritten:

- (1.) Analyse von Epochen,
- (2.) Analyse von Ähnlichkeiten,
- (3.) Analyse von Präferenzen,
- (4.) Analyse von Schwellwerten,
- (5.) Analyse von Präferenzstatus

Der Mechanismus ist wie im vorherigen Abschnitt geschildert, je nach Objekt und Zielgruppe skalierbar. Für die nun folgende Schilderung wird die Präferenzanalyse für Kunden beschrieben, welche N Personen vereinen die eine Bestellung tätigen können (B2B) und die Analyse findet auf Ebene von Abhängigkeiten statt.

2.2.1 ANALYSE VON EPOCHEN:

Die Analyse von Epochen bewirkt, dass für eine Firma A (Kunde), alle Bestellvorgänge gefiltert werden, welche von Personen getätigt wurden, welche der Firma A zugeordnet sind. Diese Bestellvorgänge werden anschließend auf Epochen reduziert. Das heißt dass eine Firma mit den chronologisch geordneten Bestellvorgängen [AB1, CA4, DE2] drei Epochen hat [AB1 = 1, CA4 = 2, DE2 = 3]. Dieses Prinzip folgt der Annahme, dass weiter zurückliegende Epochen einen schwächeren Einfluss auf die Präferenz haben, als kürzlich getätigte Epochen. So kann jeder Epoche ein dynamischer Epochenwert zugeordnet werden, welcher mit Zunahme der Epochennummer zunimmt. Für TaxoRecommend wurde die Formel präsentiert in [2] zusammengefasst, mit der Hinzunahme einer Variablen. Diese erlaubt, dass die Abnahme des Epochenwertes vom Betreiber des Multi-Channel Handels dynamisch variieren kann. Beispielsweise können hiermit Händler, welche Kunden mit vielen und sich oft ändernden Präferenzen haben, eine höhere Variable verwenden. Händler, welche Kunden mit geringen oder oft gleichbleibenden Präferenzen haben, sollten eine niedrigere Variable festlegen:

$$Epochenwert_{Epoche} = e^{-\left(\left(\frac{1}{N_{Epochen}} * Variable\right) * \left(\frac{1}{2}\right)\right) * (N_{Epochen} - Epoche)}$$

Für jede Epoche werden auf Basis der ursprünglichen Bestellnummer, die bestellten Abhängigkeiten ermittelt. Diese ergeben sich aus dem taxonomischen Pfad: Produkt -> Klasse -> Abhängigkeit. Abschließend wird die Menge von allen Abhängigkeiten mit den bestellten Abhängigkeiten subtrahiert, um eine

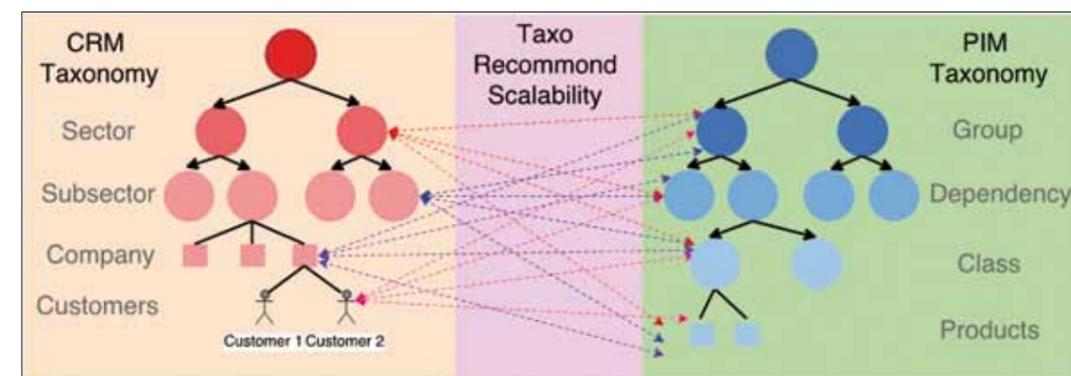


Abbildung 3: Skalierbarkeit innerhalb des Präferenzanalyse-systems TaxoRecommend, ergebend aus der Flexibilität multipler Taxonomien im E-Commerce Kontext.

Menge von Nicht-Präferenzen zu ermitteln. Diese Erkenntnisse werden final in einem Fünf-Tupel gespeichert: $Epochs = (Company.ID, Order.ID, Epoch, [Preferences], [Non - Preferences])$.

2.2.2 ANALYSE VON ÄHNLICHKEITEN:

Da Kunden eine unterschiedliche Loyalität gegenüber Ihrer Präferenzen haben, wird die Ähnlichkeit zwischen den Epochen berechnet. Die Ähnlichkeit ergibt sich hierbei aus dem arithmetischen Mittel von Jaccard und Cosine Ähnlichkeit basierend auf den Vergleich aller Epochen, respektive einem Vergleich der Mengen aus Nicht-Präferenzen. Im Gegensatz zu der Präferenz-Menge ist die Nicht-Präferenz-Menge in aller Regel harmonischer ausgeprägt und bewirkt demnach eine höhere Präzision der Vorhersagegenauigkeit. Dieses Resultat wird in einem Zwei-Tupel zwischengespeichert:

$Similarity = (Company.ID, Similarityvalue)$.

2.2.3 ANALYSE VON PRÄFERENZEN:

Der Präferenzwert schreibt jeder Abhängigkeit einen Wert zu (0 bis 1). Je höher dieser Wert, desto höher ist die Präferenz. Der Präferenzwert basiert hierbei auf einer Multiplikation von Epochenwert und Ähnlichkeitswert, gesichert in einem Drei-Tupel:

$Preference Value = (Company.ID, Dependency.ID, Preference.Value)$.

2.2.4 ANALYSE VON SCHWELLENWERTEN:

Um später entscheiden zu können, ob tatsächlich eine Präferenz vorliegt, werden sog. Schwellwerte benötigt (eng. Threshold). TaxoRecommend ist ausgelegt auf insgesamt drei natursprachliche Präferenzangaben: niedrig (keine Präferenz), mittel (interessiert) und hoch (präferiert). Hierdurch ergibt sich der Bedarf von insgesamt zwei Schwellwerten. Der erste Schwellwert prüft, ob eine mittlere Präferenz vorliegt. Der zweite Schwellwert prüft eine hohe Präferenz. Beide Schwellwerte beziehen sich auf das arithmetische Mittel aller Epochenwerte, wobei der Administrator auf beide Schwellwerte durch Variablen Einfluss nehmen kann. So kann er die Schwellenwerte in beide Richtungen (höher, niedriger) skalieren. Beide Werte werden so für jede Firma dynamisch berechnet und abschließend in einem Drei-Tupel gesichert:

$Thresholds = (Company.ID, Threshold1, Threshold2)$

Objekt	Northwind	Adventureworks	Premium-Tool
CRM.Company	93	700	500
CRM.Sector	5	4	21
CRM.Subsector	18	15	59
CRM.Person	93	700	500
CRM.Order_Header	829	31464	1400
CRM.Order_Detail	2155	121317	1218
CRM.Product	77	320	118
PIM.Class	22	37	43
PIM.Group	8	4	9
BKL.Dependency	16	14	21

Tabelle 2: Charakteristik der für die Bewertung verwendeten Datenbanken.

Datenbank	Precision	Recall	F-Measure	Accuracy
Northwind	0.89	0.60	0.71	0.59
Adventureworks	0.88	0.72	0.80	0.74
Real-World Tool	0.97	0.92	0.94	0.89

Tabelle 3: TaxoRecommend Resultate Evaluierung.

2.2.5 ANALYSE VON PREREFERENZSTATUS:

Für jede Firma und jede Abhängigkeit wird abschließend ein Vergleich des Präferenzwertes und der beiden Schwellenwerte durchgeführt. Dies bewirkt danach den finalen Status von einer Firma für eine Abhängigkeit. Ist der Präferenzwert kleiner oder gleich des hohen Schwellenwertes liegt eine hohe Präferenz vor. Ist dieser kleiner des hohen, aber größer oder gleich des mittleren Schwellenwertes, liegt eine mittlere Präferenz vor, ansonsten liegt keine Präferenz vor, ausgedrückt durch ein Drei-Tupel:

$Preference = (Company.ID, Dependency.ID, Preference.Status)$

3. BEWERTUNG

TaxoRecommend wird bewertet mit Hilfe von zwei öffentlichen Datenbanken (Adventureworks, und Northwind), welche durch die Microsoft Projekt Hosting Seite Codeplex [9] bereitgestellt werden und einer Datenbank bereitgestellt durch einen deutschen Hersteller von Premium-Werkzeugen, im Folgenden genannt Premium-Tool. Die Charakteristiken der Datenbanken sind in Tabelle 2 zusammengefasst. Die Präzision von TaxoRecommend wird hierbei evaluiert anhand der vier Standardmetrik im Bereich des Information Retrieval [4]: Precision, Recall, F-Measure, und Accuracy.

Precision sagt aus, wie oft eine Abhängigkeit korrekt als irrelevant klassifiziert wurde:

$$Precision = \frac{\sum TP}{\sum TP + \sum FP}$$

Recall sagt aus, wie viele irrelevante Abhängigkeiten gefunden wurden:

$$Recall = \frac{\sum TP}{\sum TP + \sum FN}$$

Der F-Measure Score (auch F1) vergleicht das harmonische Mittel der beiden zuvor genannten:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Und schließlich, Accuracy bestätigt wie akkurat die Vorhersage verläuft:

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum FP + \sum FN + \sum TN}$$

In allen vier genannten Formeln gilt: TP ist eine wahr positive Analyse, TN eine wahr negative Analyse, FP eine falsch positive Analyse und TN eine wahr negative Analyse.

Die Resultate zeigen, dass TaxoRecommend mit einer Präzision von im Durchschnitt 91,33 % agiert, siehe Tabelle 3. Dies entspricht einer nahezu ähnlichen Präzision, als wäre die Analyse von einem Marketingexperten selbst durchgeführt worden. Insbesondere ist anzumerken, dass das vorgestellte Präferenzanalyse-System besonders präzise mit realen Daten agiert, nämlich mit 97 %.

4. FAZIT

Dieser Artikel behandelte das Präferenzanalyse-System TaxoRecommend, welches in seiner Datenstruktur an das Datenmodell von Microsoft Dynamics CRM angelehnt ist. Durch die Nutzung der CRM Daten kann das System für den heute oft verwendeten Multi-Channel Vertriebsansatz verwendet werden, ist skalierbar hinsichtlich zu analysierender Objekt- und Zielgruppenebene und kann

problemlos in die bestehende Systemlandschaft eingebunden werden.

TaxoRecommend wurde anhand von drei Datenbanken evaluiert. Die Evaluierung zeigt, dass TaxoRecommend mit einer Präzision ähnlich des Marketingexperten agiert.

REFERENZEN

- [1a] Angermann, H. und Ramzan, N.; „E-Commerce mit Smartstore.NET – Bewertung des durch Microsoft vertriebenen Shop-Systems“; Visual Studio One, März 2016; S. 39-42.
- [1b] Angermann, H. and Ramzan, N.; TaxoPublish: „Towards a solution to automatically personalize taxonomies in e-catalogs“; Elsevier, Expert Systems with applications; Dezember 2016; S. 76-94.
- [2] Ding, Y. and Li, X.; „Time weight collaborative filtering. Association for Computing Machinery, In Proceedings of the 14th ACM International conference on information and knowledge management; New York, NY, USA; 2005; 4; S. 85-92.
- [3] He, Ch. Und Parra, D. und Verbert, K.; „Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities“; Elsevier, Expert Systems with applications; September 2016; S. 9-27.
- [4] Powers, D.; „Evaluation: From precision, recall and F-factor to ROC, informedness, markedness & correlation“; Technical Report; Flinders University; 2007.
- [5] Wu, X. and Kumar, V. and Ross Quinlan, J. and Ghosh, J. and Yang, Q. and others; „Top 10 algorithms in data mining“; Springer, Knowledge and Information Systems, January 2008, S. 1-37.
- [6] <http://bit.ly/18T1gLL>
- [7] <http://gtnr.it/2dg8Zvf>
- [8] <http://gtnr.it/2cUWSke>
- [9] <http://bit.ly/UB16Eg>

SCHULUNGS-TERMINE

MICROSOFT SQL SERVER

Power-Woche: SQL Server Abfragen, 5 Tage

Karlsruhe ab 07. Nov.
Leipzig ab 28. Nov.
Düsseldorf ab 05. Dez.

SQL Server - Abfragen optimieren, 2 Tage

Karlsruhe ab 10. Nov.
Berlin ab 17. Nov.
Düsseldorf ab 8. Dez.

SQL Server - Programmierung, 3 Tage

München ab 16. Nov.
Wien ab 21. Nov.
Stuttgart 5. Dez.

Power-Woche: SQL Server Data Warehouse, 5 Tage

Düsseldorf ab 21. Nov.
Berlin ab 12. Dez.

SQL Server Integration Services (SSIS), 2 Tage

Leipzig ab 14. Nov.
Düsseldorf ab 21. Nov.
Berlin ab 12. Dez.

SQL Server Analysis Services (SSAS), 3 Tage

Düsseldorf ab 23. Nov.

SQL Server Reporting Services (SSRS), 2 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

PPEDV.DE/SQL

Info und Anmeldung
[ppedv AG](mailto:ppedv@ppedv.de) · +49-8677-988 90
schulung@ppedv.de
[facebook.com/ppedvAG](https://www.facebook.com/ppedvAG)

ARCHITEKTUR, .NET UND VISUAL STUDIO

.NET - Architektur und Designprinzipien, 3 Tage

Berlin ab 23. Nov.
Frankfurt ab 7. Dez.
München ab 18. Jan.

.NET Core, 3 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

Xamarin - Cross Plattform-Apps, 4 Tage

Düsseldorf ab 08. Nov.
Burghausen ab 28. Nov.
Köln ab 13. Dez.

Universal Windows Platform (UWP), 4 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

C# Programmierung, 4 Tage

Düsseldorf ab 08. Nov.
Stuttgart ab 22. Nov.
Burghausen ab 29. Nov.
München ab 6. Dez.

C++ Programmierung, 4 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

Windows Presentation Foundation (WPF), 4 Tage

Berlin ab 22. Nov.

Team Foundation Server, 4 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

WEB-DEVELOPMENT

Architektur von modernen Web-Anwendungen, 3 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

ASP.NET MVC – Model View Controller, 3 Tage

Nürnberg ab 21. Nov.

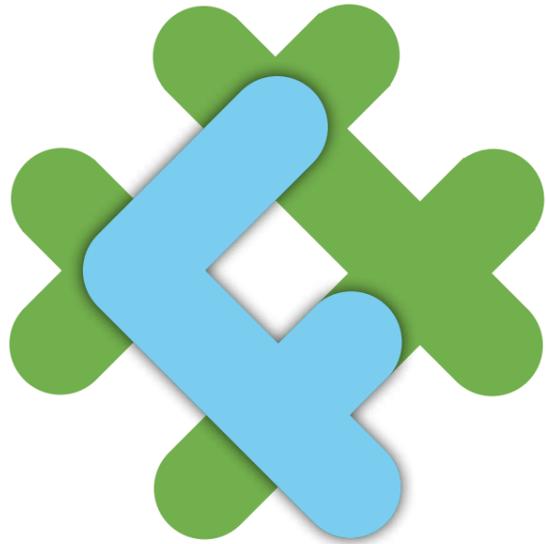
ASP.NET Web API, 2 Tage

Bundesweite Durchführung
ab 1 Teilnehmer.

PPEDV.DE/DOTNET

Info und Anmeldung
[ppedv AG](mailto:ppedv@ppedv.de)
+49-8677-988 90
schulung@ppedv.de
[facebook.com/ppedvAG](https://www.facebook.com/ppedvAG)
twitter.com/ppedv





F# Das unbekannte Wesen

Funktionale Programmiersprachen als Alternative zu objektorientierten

Autor: FABIAN DEITELHOFF

Die Sprache F# ist schon lange nicht mehr nur in wissenschaftlichen Kontexten anzutreffen. Auch die verbreitete Meinung, F# wäre nur etwas für die Finanzwelt oder allgemein für mathematische Probleme geeignet, könnte falscher nicht liegen. Über die Jahre hat sich F# aus seiner Nische hervorgearbeitet und wird aktuell deutlich aktiver eingesetzt. Der Artikel wirft einen Blick auf die Besonderheiten, die Anforderungen und die Unterschiede zur objektorientierten Sprache C#.

F# ist nur etwas für wissenschaftliche oder mathematische Anwendungen. Diese beiden Vorurteile halten sich bis heute sehr hartnäckig. Die Gründe dafür sind verschiedener Natur. Zum einen hat F# tatsächlich als Forschungsprojekt bei Microsoft Research das Licht der Welt erblickt. Und das schon vor genauer Zeit im Jahr 2002. Nicht viel später als C# beziehungsweise Visual Basic .NET für die .NET Plattform.

In den vergangenen Jahren hat F# einen großen Sprung in Bezug auf die Akzeptanz bei Entwicklern gemacht. So langsam scheint es als komme die Programmiersprache aus seiner Nische hervor. Ein größeres Problem sind die vielen Vorurteile, mit denen Entwickler der Sprache F# begegnen. Nicht nur auf technischer Basis, sondern auch Vorurteile die Community betreffend.

WAS IST F#?

Heruntergebrochen ist F# eine Programmiersprache für das .NET Framework. Was bedeutet, dass F#-Code kompiliert wird und der Zwischencode aus der Common Intermediate Language (CIL) besteht. Identisch zu C# und Visual Basic .NET. F# ist multiparadigmatisch, was

übersetzt heißt, dass verschiedene Programmierparadigmen unterstützt werden. Dazu gehören funktionale, imperative und objektorientierte Ansätze und Konzepte.

F# wird auch gerne als „functional-first“ Programmiersprache bezeichnet, weil die funktionalen Ansätze im Vordergrund stehen. Die Sprache ist zudem Open Source, steht unter der Apache-Lizenz 2 und ist Cross-Plattform tauglich. Letzteres bedeutet, dass F# unter Linux, Mac OS X, Android, iOS, Windows, GPUs und im Browser läuft. Des Weiteren gibt es zahlreiche zusätzliche Projekte, die F# als Ausgangssprache nutzen, aber nicht in die CIL übersetzen.

Hinter vielen Entwicklungen rund um F# steckt zum einen eine äußerst aktive Community. Viele Entwicklungen, von Tools, über Anleitungen, Sessions auf Events bis hin zu Verbesserungen des F# Compilers an sich sind dieser Community zu verdanken. Zusätzlich gibt es noch die F# Software Foundation [1]. Die Foundation ist ein Zusammenschluss verschiedener F# Programmierer und hat sich die Ziele gesetzt, die Programmiersprache F# zu bewerben, zu beschützen, sowie in der Entwicklung voran zu bringen.

DIE ENTWICKLUNG-UMGEBUNG

Dem unermüdlichen Einsatz der Community ist es zu verdanken, dass F# mittlerweile in einer Vielzahl von Entwicklungsumgebungen zu Hause ist. Visual Studio gehört unter anderem dazu, allerdings offiziell erst ab der Version 2010, in der die Sprache erstmals offiziell unterstützt wurde. Für Visual Studio 2008 steht eine Erweiterung zur Verfügung.

Unter Windows, also einem Betriebssystem z. B., gibt es zudem viele weitere Möglichkeiten. Soll Visual Studio, in den Versionen 2012, 2013 oder 2015 zum Einsatz kommen, ist F# bereits mit an Bord. Es wird automatisch installiert, wenn das erste F# Projekt erstellt oder geöffnet werden soll. Alternativ lassen sich die Visual F# Tools [2] von Microsoft manuell installieren. Abbildung 1 zeigt die Projekttemplates in Visual Studio 2015, die zur Verfügung stehen, wenn F# Unterstützung installiert ist.

Zusätzlich lohnt sich ein Blick auf die Visual F# Power Tools [3], die von der Community bereitgestellt werden. Diese Erweiterung unterstützt Visual Studio 2013 und 2015. Seit der Version 2.0.0 gibt es keinen offiziellen Support mehr für Visual Studio 2012. Die Power Tools bieten

eine Reihe von Features und Komfortfunktionen an, die in der originalen F# Unterstützung für Visual Studio nicht mit von der Partie sind. Zum Beispiel eine bessere Syntaxhervorhebung, Formatierung, Refaktorisierungen, eine Code-Navigation, Tastaturkürzel und vieles mehr.

Eine weitere Möglichkeit ist der freie Code-Editor Visual Studio Code. Er ist kostenfrei, Open Source, Cross-Plattform fähig und unterstützt eine große Menge weiterer Sprachen. Durch das Ionide [4] Projekt, das sich hervorragend in Visual Studio Code integriert, steht F# unter anderem auch in Visual Studio Code zur Verfügung. Abbildung 2 zeigt Visual Studio Code in Version 1.5.3 mit installierter F# Ionide Erweiterung. Diese fügt nicht nur IntelliSense und die Syntaxhervorhebung für F# hinzu, sondern auch die mächtige F# Interactive Konsole, mit der Code-Snippets schnell und unkompliziert getestet werden können.

Neben den ganzen Entwicklungsumgebungen und Code-Editoren gibt es zu guter Letzt noch die Möglichkeit, den F# Compiler und einige Tools ohne jeglichen Schnickschnack zu installieren. Die Anforderungen dafür sind das .NET Framework in Version 4.5, ein aktuelles Windows SDK, die Microsoft Build

Tools 2015 und die Visual F# Tools in Version 4.0. Zumindest zum aktuellen Zeitpunkt. Im weiteren Verlauf der Entwicklung werden sich diese Anforderungen sicherlich ändern. Eine genaue Auflistung dieser Anforderungen verrät die spezifische Windows-Rubrik auf der F#-Webseite [5]. Dort sind auch Informationen vorhanden, falls der F# Compiler selbst kompiliert werden soll. Und selbstverständlich auch diejenigen Informationen, um F# auf anderen Plattformen einzusetzen.

WARUM F# NUTZEN?

Bevor es um die konkrete Syntax und um zahlreiche Beispiele der Sprache F# geht, versucht der Artikel zunächst eine deutlich grundlegendere Frage zu beantworten: Warum soll ich F# einsetzen? Beziehungsweise warum sollte ich mich damit beschäftigen, wenn ich noch gar nicht genau weiß, was dahintersteckt und ob ich es überhaupt für meine Anwendungsfälle einsetzen könnte?

Diese allgemeine Frage ist nicht leicht zu beantworten. Es lassen sich zwar allgemeingültige Vorteile, Eigenschaften beziehungsweise Unterschiede von F# nennen, insbesondere im Vergleich zu anderen Programmiersprachen. Ob diese

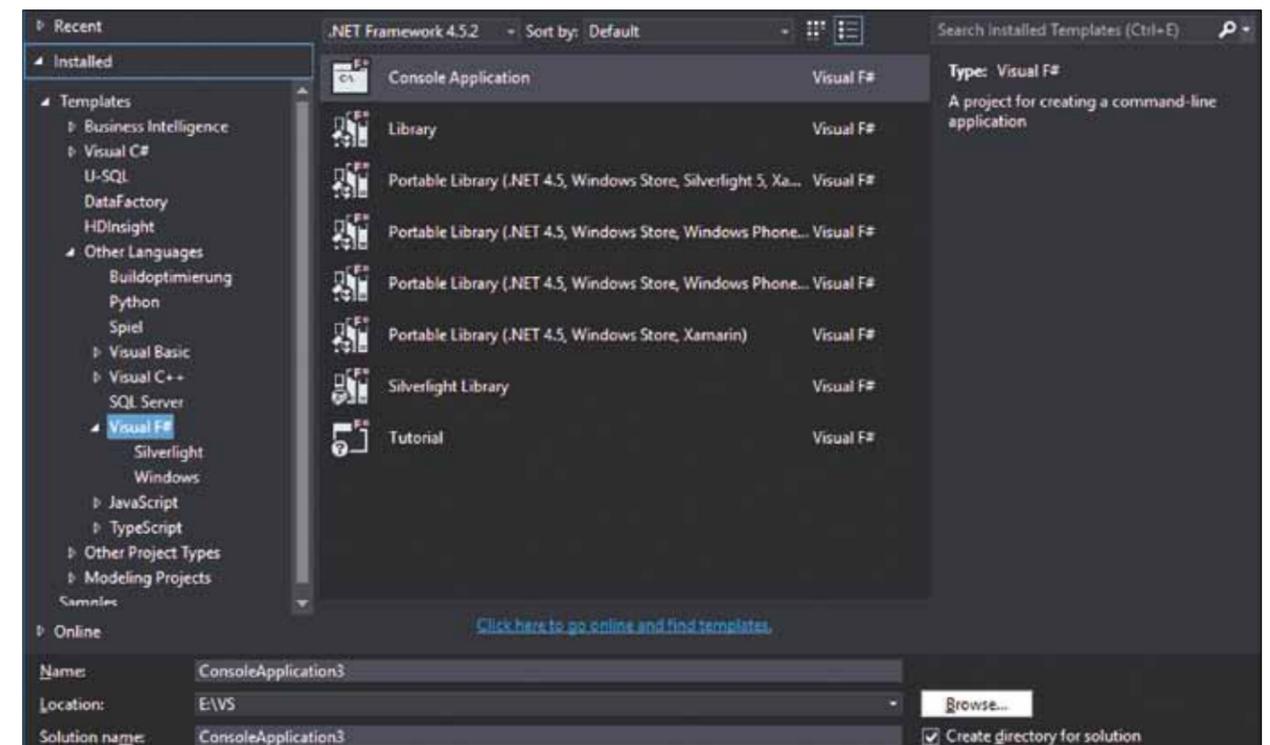


Abbildung 1: Projekttemplates in Visual Studio 2015 für F#.

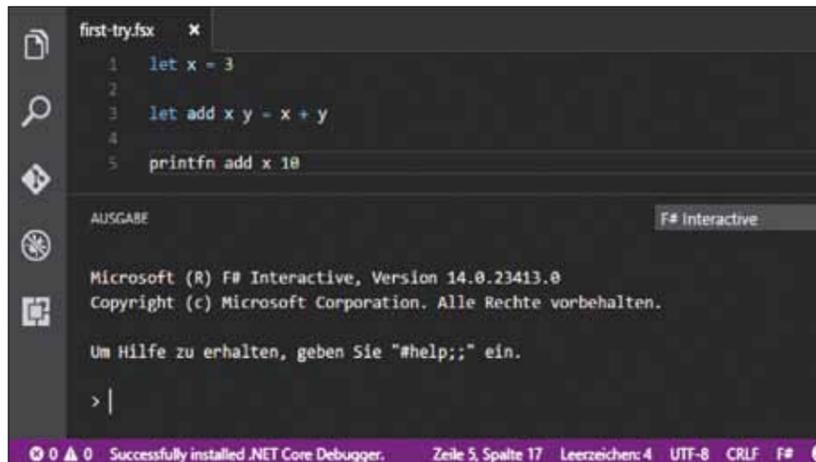


Abbildung 2: Visual Studio Code mit installierter F# Unterstützung

aber überzeugend sind oder nicht, lässt sich im Grunde genommen nur durch eigene Tests bestätigen oder widerlegen.

Immer wenn darauf herumgeritten wird, dass sich F# nur für wissenschaftliche oder mathematische Anwendungen eignet, ist das, in dieser Verallgemeinerung, sicherlich falsch. Nicht ganz so falsch ist es allerdings, dass F# schon einige Eigenschaften besitzt, die es für diese Anwendungsfälle ganz besonders geeignet macht. Aber eben nicht ausschließlich, denn es gibt auch gute Gründe für F# im Enterprise Umfeld.

Eine in diesem Zusammenhang häufig genannte Eigenschaft von F# ist die Prägnanz, mit der sich Code ausdrücken lässt. Im Übrigen eine Eigenschaft, die ganz häufig im Zusammenhang mit funktionalen Programmiersprachen genannt wird. Prägnanz bedeutet in diesem Zusammenhang, dass F# nicht mit dem üblichen Rauschen zu kämpfen hat, das in anderen Programmiersprachen anzutreffen ist. Zum Beispiel ausgelöst durch Klammern, Semikolons und so weiter und so fort. Auch das Typsystem mit seiner Typinferenz trägt dazu bei, da nur in ganz seltenen Fällen der Typ eines Objekts überhaupt angegeben werden muss. In den meisten Fällen kann F# das selbstständig

ableiten. Diese Vorteile führen oftmals zu weniger Code, verglichen mit anderen Programmiersprachen.

F# gilt allgemein hin als annehmbare beziehungsweise bequeme Sprache. Nicht im Sinne von faul zu verstehen, sondern in der Hinsicht, dass viele Aufgaben in F# schneller gelöst werden können. Wieder im Vergleich mit anderen Programmiersprachen. Dazu gehören zum Beispiel Aufgaben wie das Erzeugen von komplexen Typen, die Verarbeitung von Listen, Vergleiche und vieles dergleichen mehr.

Die Korrektheit ist ein weiteres Merkmal, das im Zusammenhang mit F# häufig zur Sprache kommt. Durch das Typsystem, das auf Typinferenz setzt, werden viele der üblichen Fehler, wie der Zugriff auf null-Referenzen, vermieden. Werte sind zum Beispiel standardmäßig unveränderlich (Immutable), was eine große Fehlerquelle beseitigt.

Ein weiterer Vorteil, und gleichzeitig der letzte Punkt in dieser Auflistung, ist die Vollständigkeit von F#. Obwohl es sich um eine funktionale Programmiersprache handelt, ist F# als hybride Sprache designt. Konkret bedeutet das, dass sowohl funktionale wie auch objektorientierte Konzepte vertreten sind. Das macht es einfacher, mit nicht funktionalen Dingen

zu interagieren. Zum Beispiel Datenbanken, andere Anwendungen und so weiter.

Die Liste der Unterschiede zu anderen Programmiersprachen ist lang. Insbesondere, wenn der Vergleich mit nicht funktionalen Sprachen wie C# stattfindet. Und dennoch bieten die oben genannten Punkte nur einen kleinen Einblick in F# selbst. Ein möglichst vollständiges Bild ergibt sich erst, wenn selbst Hand angelegt und F# Code produziert wird.

ZWEI HAUPTUNTERSCHIEDE IM SCHNELLDURCHLAUF

Funktionale Programmiersprachen können sehr mysteriös, unverständlich und durchaus abschreckend auf Entwickler wirken. Umso mehr, je weniger vorher schon mit funktionalen Sprachen gearbeitet wurde. Daher ist es wichtig, kurz einen Blick auf grundlegende Syntaxelemente und Herangehensweise und F# zu schauen, bevor tatsächlich versucht wird, die ersten Zeilen Code zu schreiben.

Die zwei wohl größten Unterschiede zwischen der F# Syntax und Programmiersprache der C Sprachfamilie sind, dass keine geschweiften Klammern für die Definition von Code-Blöcken zum Einsatz kommt und das Leerzeichen Parameter von Funktionen trennen. Anstelle der geschweiften Klammern wird die Einrückungstiefe des Codes genutzt. Die Sprache Python ist ein guter Vergleich, weil es dort, in Bezug auf die Code-Blöcke, sehr ähnlich funktioniert. Diese beiden Hauptunterschiede sind es auch, die funktionalen Code so anders aussehen lassen. Mit Sprachen der C Sprachfamilie ist im Folgenden hauptsächlich C# gemeint, mit der die Eigenschaften von F# verglichen werden. Weitere Vergleiche mit anderen Programmiersprachen würde den Rahmen eines einzelnen Artikels sprengen.

Nach einiger Eingewöhnungszeit empfinden viele, nach eigener Aussage, die Syntax von F# als sehr klar und ausdrucksstark. Ein oft gehörtes Argument ist, dass es besser ist eine Sprache durch eine klare Syntax für das wiederholte Lesen zu optimieren, anstatt die Sprache so zu konzipieren, dass der Einstieg leichter ist.

Ob das auch für einen selbst zutrifft, lässt sich nur durch praktische Tests mit der Sprache herausfinden.

ALLGEMEINE SYNTAXELEMENTE

Wie diese beschriebenen Unterschiede, sowie die schon mehrfach erwähnte klare Syntax, aussehen, zeigen die folgenden Beispiele. Zunächst die ganz simplen Dinge: Doppelte Schrägstriche leiten Zeilenkommentare ein. Mehrzeilige Kommentare sind auch möglich:

```
(*
    Ich bin ein mehrzeiliger
    Kommentar...
*)
```

Variablen gibt es im Grunde in F# nicht. Allerdings lassen sich Werte definieren. Wie schon beschrieben sind diese unveränderlich (Immutable), nach der Definition also nicht änderbar. Das let Schlüsselwort leitet solche Werte ein:

```
let value = 5
let text = „Ich bin eine
Zeichenkette.“
```

Durch die Typinferenz sind keine Typangaben notwendig. Die leitet sich der F# Compiler selbstständig ab. Zuweisungen im klassischen Sinne, wie sie aus C# und artverwandten Sprachen bekannt sind, gibt es daher selten in F# zu sehen. Um einen veränderbaren Wert zu definieren, ist in F# explizit das Schlüsselwort mutable erforderlich. Dann lässt sich der Wert über die folgende Syntax anpassen:

```
let mutable value = 5
value <- value + 1
```

Das Komma niemals als Trennzeichen zwischen Werten genutzt werden, beweisen Listen in F#. Wie weiter oben schon erwähnt, werden Parameter von Funktionsaufrufen durch Leerzeichen getrennt. Bei Listen kommt das Semikolon zum Einsatz. Durch das let Schlüsselwort lassen sich somit Listen definieren, die wiederum unveränderlich sind. Die folgenden drei Anweisungen definieren daher verschiedene Listen, basierend auf der vorherigen Liste:

```
let list = [1;2;3;4;5]
let list2 = 0 :: list
let list3 = list2 @ [6;7]
```

Das kleine Snippet zeigt zudem weitere Konzepte. Über die beiden Doppelpunkte wird eine neue Liste mit einem neuen Element vorne erzeugt.

Das @-Zeichen dient zur Konkatenation von zwei Listen zu einer neuen. Die Elemente der neuen Liste sind in der gleichen Reihenfolge wie die Elemente der beiden Quelllisten.

Das let Schlüsselwort dient ebenfalls dazu benannte Funktionen zu definieren. Ein klassischer und gleichzeitig simpler Fall zeigt die folgende Definition einer add Funktion: `let add x y = x + y` Die Parameter stehen direkt nach dem Funktionsnamen. Nach dem Gleichheitszeichen erfolgt die Definition des eigentlichen Funktionsinhalts, also was beim Aufruf ausgeführt werden soll. Aufgerufen wird die Funktion ohne Klammern oder sonstige zusätzliche Zeichen:

```
let result = add 3 4
```

Was auffällt ist, dass keine return Anweisung notwendig ist. Die F# Rückgaben sind implizit. Eine Funktion gibt immer den Wert des letzten Ausdrucks zurück. Folgendes Beispiel macht von einer mehrzeiligen Funktion Gebrauch, um diese Tatsache noch einmal zu verdeutlichen.

```
let addTen x =
    let y = 10
    x + y
let result2 = addTen 5
```

Es handelt sich zwar zugegeben um ein konstruiertes Beispiel, trotzdem zeigt es, wie innerhalb der addTen Funktion zwei weitere Anweisungen definiert sind. Einmal mit dem schlichten Wert, gespeichert in y und die zweite, die diesen Wert in der Addition nutzt. Die Rückgabe der Funktion ist somit implizit das Ergebnis der Addition.

Ein weiteres, wichtiges Feature ist das sogenannte Pattern Matching. Daher verbirgt sich, vereinfacht gesagt, ein Switch-Case-Statement. Allerdings ein aufgemotztes, da nicht nur auf Zeichenketten verglichen werden kann. Vergleiche sind auf Typen wie Integer, Float und Objekten möglich. Zudem können verschiedene Vergleiche verschachtelt werden, was ein weiterer großer Vorteil ist. Des Weiteren sind Filter in den einzelnen Matching-Regeln mit dem when Schlüsselwort möglich. Die Liste der Features und Vorteile ließe sich endlos weiter vorsetzen. Obwohl das Pattern Matching in F# ein so großer Vorteil ist, fällt es schwer, Außenstehenden die Vorteile tatsächlich schmackhaft zu machen. Pattern

Matching ist daher ein gutes Beispiel für ein Feature, das sich schwer beschreiben lässt und dessen Vorteile erst so wirklich greifbar sind, wenn sie selber ausprobiert werden. Trotzdem darf in einem Einführungs-Artikel zu F# kein Beispiel zum Pattern Matching fehlen. Folgendes Snippet zeigt die Kombination aus einem direkten Vergleich und dem Einsatz eines Filters für die „Variable“ value, die in einem Beispiel weiter oben bereits definiert wurde:

```
let patternMatching =
    match value with
    | 5 -> printfn „value is 5“
    | value when value < 5 ->
        printfn „value is smaller“
    | _ -> printfn „Something
else!“
```

Die erste Regel, eingeleitet mit dem Schrägstrich, prüft, ob der Wert genau fünf ist. Die zweite Regel überprüft, ob der Wert kleiner fünf ist. Die letzte Regel dient sozusagen als Standardregel, wenn keine der zuvor definierten Regeln greifen. Fehlt diese, warnt der F# Compiler, dass es Werte gibt, die von dem Pattern Matching nicht erfasst wurden. Ein sehr gutes Feature, um direkt bei der Implementierung Fehlerquellen ausfindig zu machen.

An dieser Stelle gäbe es noch viele Beispiele, die einen näheren Blick wert wären. Zum Beispiel Tuple, Records Types, Union Types und so weiter und so fort. Allerdings ist selbst dieser Artikel schon umfangreich, obwohl er nur einen kleinen Einblick gewährt. Die Thematik an sich ist eine komplexere, wie das bei jeder Vorstellung beziehungsweise Einführung in eine neue Programmiersprache der Fall wäre.

FUNKTIONALES DENKEN

Sehr wichtig ist es, zu verstehen, dass funktionale Programmiersprachen wie F# nicht nur andere Stilmittel im Sinne von Syntax und Programmaufbau benutzen. Vielmehr handelt es sich um ein ganz anderes Konzept. Funktionales Denken ist daher essentiell, um alles aus F# herauszuholen. Grundsätzlich erlaubt F# es zwar, auch ohne dieses funktionale Denken und ohne den Einsatz von Funktionen weit zu kommen, allerdings bleiben dann viele Vorteile auf der Strecke.

WAS IST TYPINFERENZ?

Typinferenz, auch als Typableitung bezeichnet, ist ein Verfahren zur Ableitung von konkreten Typen durch die im Quelltext angegebenen Informationen bzw. Daten. In einer Programmiersprache, die Typinferenz nutzt, ist es nicht erforderlich, Typangaben zu machen. Diese Informationen können aus den restlichen Angaben und den Typisierungsregeln rekonstruiert werden. Für die Ableitung der Typinformationen werden dieselben Regeln bemüht, die auch zur Typprüfung dienen.

Zum Beispiel ist es bei F# wichtig zu verstehen, dass das Gleichheitszeichen nicht mit einer Zuweisung gleichzusetzen ist. Vielmehr handelt es sich um ein Binding. Beim Beispiel der Funktion `addTen`, definiert weiter oben im Artikel, ist `x` keine Variable oder ähnliches, die einen Wert zugewiesen bekommt. Es ist ein Name, eine Assoziation für einen Wert. Weiter gedacht ist der Name `addTen` selbst wiederum nur ein Binding auf die Funktion, die wir definiert haben. Immer wenn dieser Name irgendwo im Code auftaucht, wird er gegen die definierte Funktion ausgetauscht. Der Name `addTen` wird daher auch als `function value` bezeichnet.

Funktionales Denken in einen einzelnen Artikel zu komprimieren, ist nahezu ausgeschlossen. Es gibt eine hervorragende Blogserie zu dem Thema, die über viele verschiedene Wege und mit Hilfe vieler verschiedener Beispiele versucht, das Thema herunter zu brechen und dem interessierten Leser näherzubringen. Gemeint ist die *Thinking functionally* Serie [6] auf dem Blog *F# for Fun and Profit* [7]. Ein Blog, der in Gänze zu empfehlen ist, wenn es um das Erlernen von F# beziehungsweise der funktionalen Denkweise im Allgemeinen geht.

HÄUFIG GEMACHTE FEHLER

Einen Einblick auf die häufigsten Fehler, die beim Einstieg in F# gemacht werden, erleichtern den Einstieg noch einmal enorm. Von den Fehlern anderer zu lernen ist selten eine schlechte Idee. Die folgenden Punkte sind als lose Sammlung zu verstehen. Auch die Reihenfolge ist beliebig, da es sicherlich noch weitere nennenswerte Fehler und deren Lösung gäbe.

Runde Klammern beim Aufruf von Funktionen sind Tabu. Wie schon zuvor kurz erwähnt, werden Klammern in F# sehr selten benötigt.

Parameter werden auch nicht durch Kommata voneinander getrennt. Insbesondere in Kombination mit runden Klammern ergibt das ein Tupel mit `n` Elementen. Was dazu führen kann, dass der F# Compiler moniert, dass zu wenige Parameter für die aufzurufende Funktion angegeben wurden.

Die Elemente einer Liste werden nicht durch Kommata voneinander getrennt. Kommata geben wieder ein Tupel an. Im Zweifel wird eine Liste erzeugt, die nur ein Element, nämlich das `n`-Element Tupel enthält.

Den Operator `!=` gibt es nicht in F#. Eine Negation wird mit dem Schlüsselwort `not` angegeben. Ungleich bedeutet zudem der Operator `<>` wie beispielsweise in SQL. Zuweisungen werden durch den Operator `<-` angegeben, nicht durch das Gleichheitszeichen.

Der bereits genannte Blog *F# for Fun and Profit* bietet eine erweiterte Liste von häufig gemachten Fehlern an. Inklusiv einer Übersicht häufiger Fehlermeldungen des F# Compilers. Der Blogpost [8] ist definitiv einen Blick wert.

WEITERE PROJEKTE

In der F# Community sind eine ganze Reihe von Projekten entstanden, die das Ökosystem um die Sprache herum bilden. Sie sind durchaus ein weiterer Grund dafür, sich mit der Programmiersprache näher zu befassen. Folgende Projekte sind in diesem Zusammenhang erwähnenswert. Die Liste ist allerdings alles andere als vollständig und

nur als interessanter Ausschnitt zu verstehen:

- **FsReveal:** Erzeugt aus Markdown und F# Skriptdateien Slides für `reveal.js`.
- **FSharp.Formatting:** F# Tools zur Generierung von Dokumentationen (Markdown und F# Code-Formatierung).
- **FAKE:** F# Build-Automatisierung.
- **Paket:** Dependency Manager für .NET - Unterstützt NuGet-Pakete und GitHub Repositories.
- **Type Provider:** Sammlung von Datenschnittstellen für verschiedene Formate beziehungsweise Quellen (CSV, HTML, JSON, XML, Amazon S3).

FAZIT

F# ist eine mächtige Programmiersprache. Diese Flexibilität ist eine große Stärke, da eben nicht nur ein einziges Programmierparadigma unterstützt wird, sondern gleich mehrere. Allerdings besteht die Gefahr, dass dadurch der Fokus verloren geht. In F# gibt es immer mehrere Wege, etwas zu tun.

Eine große Hürde für Einsteiger oder Umsteiger ist, dass F# uns Entwickler nicht immer gut an die Hand nimmt. Etwas, was Sprachen wie C# auf viele Arten machen. Hinzu kommt, dass F# seine volle Stärke erst dann ausspielt, wenn die funktionale Denkweise beim Entwickler angekommen ist. Das erfordert Zeit und Mühe, da diese Erkenntnis nicht einfach so vom Himmel fällt.

Diese Punkte, die sich vermutlich deutlich negativer anhören, als das tatsächlich beabsichtigt ist, sollen aber nicht abschrecken. F# ist als „General Purpose Language“ definitiv einen Blick wert. Die Community ist in der Vergangenheit stark gewachsen und mit ihr auch die verfügbaren Projekte. Die Integrationen in verschiedene Entwicklungsumgebungen und Code-Editoren ist sehr gut und auch die Dokumentation hat einen großen Sprung gemacht.

Trotzdem ist es wichtig, und das gilt ohne Einschränkung für jede Programmiersprache, vor dem Einsatz abzuwägen, ob F# die richtige Wahl ist. Individuell auf die eigenen Anwendungsfälle abzuwägen ist niemals verkehrt und gerade dann wichtig, wenn neue Technologieentscheidungen anstehen. Mindestens genauso wichtig sind aber auch erste praktische Tests, da sich Technologien, gerade Programmiersprachen, niemals von der Ferne aus wirklich einschätzen lassen.

LINKS UND QUELLEN

- [1]: Webseite der F# Software Foundation: <http://foundation.fsharp.org/>
- [2]: Download der Visual F# Tools: <http://bit.ly/2dyMeAG>
- [3]: Projektseite der Visual F# Power Tools: <http://bit.ly/2dqWJ94>
- [4]: Webseite des Ionide Projekts: <http://ionide.io/>
- [5]: F# unter Windows einsetzen: <http://fsharp.org/use/windows/>
- [6]: Blogserie zum funktionalen Denken: <http://bit.ly/2d3ortY>
- [7]: Der Blog *F# for Fun and Profit*: <http://fsharpforfunandprofit.com/>
- [8]: Wichtige Tipps zur Fehlersuche in F#: <http://bit.ly/2cBMmSS>

MARKTANTEILE VON WINDOWS 10 SINKEN

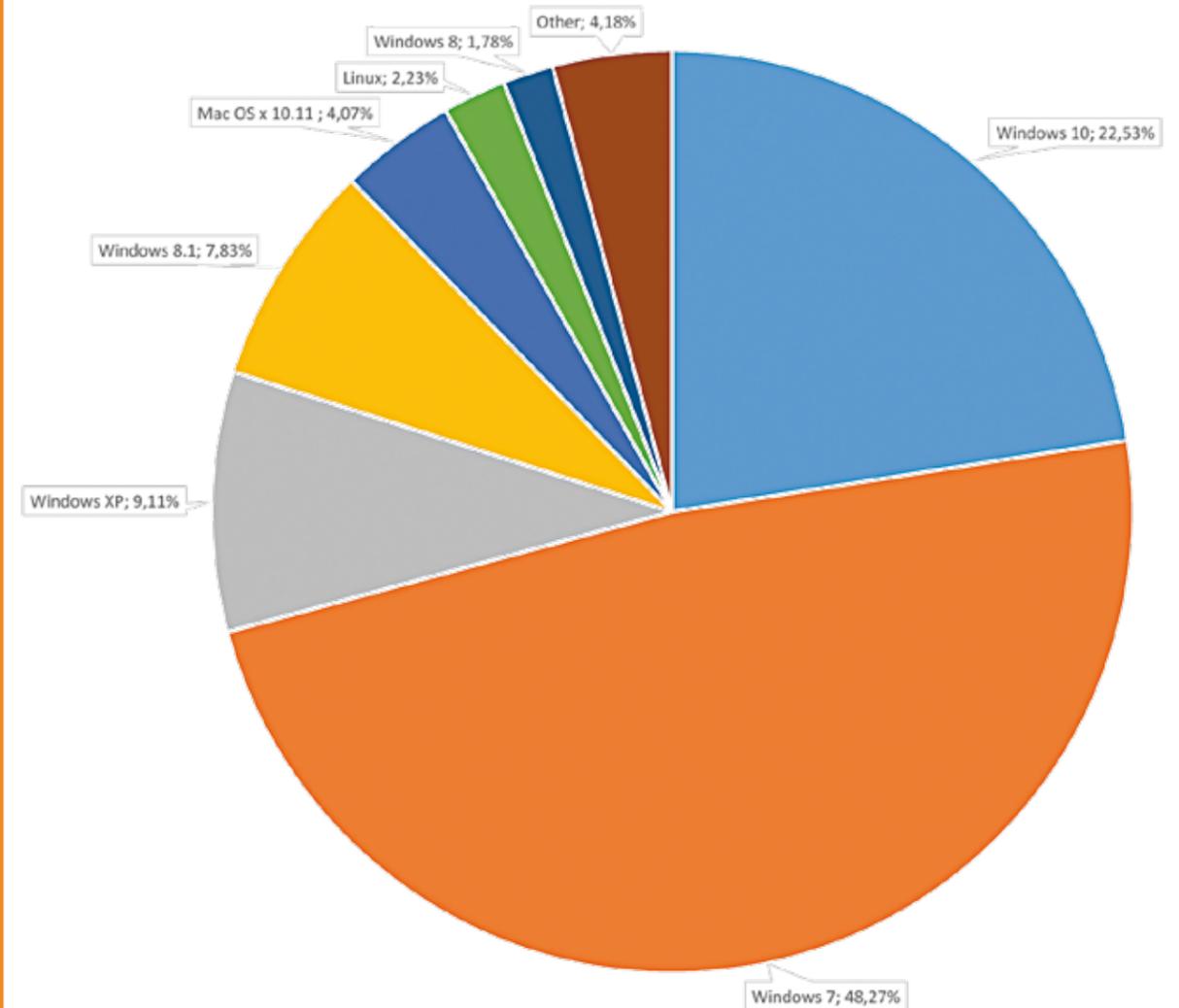
Im August erzielte Windows 10 mit einem Wachstum von beinahe 1,9 Prozent Marktanteilgewinne. Im September sank jedoch der Marktanteil von Windows 10 um 0,46 Prozent auf 22,53 Prozent, wobei die frühere Windows 7 Version eine Steigerung von ca. einem Prozent erreichte und somit 48,27 Prozent aller PC Marktanteile einnimmt. Die Begründung für diesen Marktanteilanstieg könnte darin liegen, dass mit Ende Oktober das Herstellungs- und Verkaufsende von Windows 7 Computern gesetzt ist. Ausnahme hierbei sind bereits im Handel vorrätige Systeme.

Anzumerken ist, dass sich bei früheren Windows-Versionen eine Marktanteilsenkung bemerkbar machte. Im Gegenzug steigerte sich dafür das macOS 10.12 Sierra von Apple um 0,21 Prozent. Betrachtet man jedoch alle macOS-Versionen, erkennt man, dass diese auf den niedrigsten Stand seit 2015 geschrumpft sind, was einen Marktanteil von 6,92

Prozent bedeutet. Linux erreichte eine Steigerung von 0,12 Prozent auf einen Anteil von insgesamt 2,23 Prozent.

MARKTANTEIL BEI BROWSERN

Im September konnte der Google-Browser Chrome seine Führung mit einer Zunahme von 0,45 Prozent auf 54,41 Prozent ausbauen. Im Gegenzug dazu verzeichnete der Internet Explorer von Microsoft einen Rückgang um 1,9 Prozent und erlangte somit nur noch 25,48 Prozent. Durch die Marktanteilsenkung von Windows 10 blieb Edge bei 5,16 Prozent. Mozilla Firefox konnte seinen Anteil um 1,5 Prozent auf insgesamt 9,19 Prozent erhöhen und erreichte seinen höchsten Marktanteil seit dem Mai 2016. Dieser Zuwachs gelang Mozilla Firefox vermutlich durch den Wechsel vieler IE-Nutzer zu diesem Browser.



FORCE FEEDBACK PROGRAMMING

Clean Code leicht gemacht

Jetzt wird es leichter, Clean-Code zu schreiben! Endlich! Denn diese Idee hat Ralf Westphal schon lange mit sich herumgetragen: Force Feedback Programming.

Autor: RALF WESTPHAL

DER HINTERGRUND

Wenn wir etwas verändern wollen, wenn wir etwas Neues lernen wollen, dann brauchen wir Feedback. Wir müssen feststellen, ob wir auf dem richtigen Weg sind.

Wenn wir lernen, Fahrrad zu fahren, entsteht solches Feedback sehr schnell. Die Schwerkraft zerrt konstant an uns, sodass wir unmittelbar merken, ob wir beim Fahrradfahren noch etwas falsch machen.

Etwas länger dauert es, wenn wir lernen, unsere Schuhe zu binden. Die Schleife mag ordentlich aussehen – doch, wenn wir dann eine Stunde gegangen sind, geht unser Schuh vielleicht auf. Das macht es etwas schwerer, eine effektive Schuhbindung zu lernen.

Je schneller Feedback also kommt, desto leichter lernen wir. Deshalb können auch so viele Menschen „irgendwie“ programmieren. Compiler und sofort gestartetes Programm geben so schnell Feedback, dass man zügig Kompetenz aufbauen kann.

Anders ist es jedoch mit dem sauberen Programmieren, mit Clean Code. Das ist viel, viel schwieriger zu lernen. Eben weil das Feedback zur Qualität der Wandelbarkeit des Codes sehr spät eintrifft – womöglich ereilt es sogar den Codeautor nicht einmal selbst, sondern seinen Kollegen.

In stark verzögertem Feedback sieht Ralf Westphal den Hauptgrund dafür, dass Clean Code zwar im Bewusstsein vieler Entwickler angekommen sein mag, allerdings nicht in deren Code. Zwischen Kopf und Hand, zwischen Denken und Tun liegt hier eine lange Leitung.

```
internal class Flatten
{
    public Flatten() : this(false) {}
    public Flatten(bool treatAllIEnumerableAlike)
    {
        // handleNull
        var handleNull = new Flatten_null();
        var scalar = new Flatten_scalar();
        var cache = new Cache();
        var ienum = new Flatten_IEnumerableAlike();
        var array = new Flatten_array();
        var nonScalar = new Flatten_non_scalar_field_by_field();
        var collect = new Collect();

        // _process
        _process = handleNull.Process;

        handleNull.Null += collect.Process;
        handleNull.PassOn += scalar.Process;

        scalar.Scalar += collect.Process;
        scalar.PassOn += cache.Process;

        cache.CacheHit += collect.Process;
        cache.CacheMiss += ienum.Process;

        ienum.MetaObject += collect.Process;
        ienum.Element += handleNull.Process; // hier würde der Compiler sagen
        ienum.PassOn += array.Process;

        array.MetaObject += collect.Process;
        array.Element += handleNull.Process; // hier würde der Compiler sagen
        array.PassOn += nonScalar.Process;

        nonScalar.MetaObject += collect.Process;
        nonScalar.FieldValue += _process; // verändert sich bei jedem Aufruf

        collect.Result += _ => this.Result(_);

        // Fertig
        ienum.Configure(treatAllIEnumerableAlike.ToString());
    }

    private readonly Action<object> _process;
    public void Process(object objectToFlatten)
    {
        _process(objectToFlatten);
    }
}
```

Abbildung 1: Anhand der braunen Hinterlegung sieht man deutlich das dieser Code nicht clean ist

Deshalb suchte er nach einem Weg, über das hinauszugehen, was er mit Stefan Lieser zusammen im Clean Code Developer Wiki zusammengetragen hat. Und auch das, was sie in Trainings der CCD School vermitteln, ist noch zu wenig. Denn selbst mit der Methodenidee vom Green Coding im Kopf braucht es in der Praxis Feedback, ob man sich auf dem Weg zu guter Anwendung befindet.

Tools wie StyleCop oder TeamScale liefern zwar Feedback, doch auch wieder sehr spät. Wenn Feedback zur Sauberkeit erst bei einem späteren Build oder gar erst in einem Review entsteht, ist das kraftlos.

Pair Programming kann Feedback viel unmittelbarer herstellen. Doch erstens programmiert nicht jeder im Pair. Zweitens ist der Feedbackgeber auch nur ein Mensch. Das bedeutet: Er kann mit seinem Feedback nachlässig sein, er kann einen geringeren/fluktuierenden Anspruch an Sauberkeit haben, er kann auch Vermeiden, durch wiederholte Hinweise auf Unsauberkeit in Konflikt mit dem Partner zu geraten. Aber wie denn dann?

UNMITTELBARES FEEDBACK

IDEs haben inzwischen einen Testrunner eingebaut. Der führt auf Wunsch die Unit Tests im Code aus. Das ist ein schnelles Feedback zur Verhaltenskorrektheit des Codes. Es ist ähnlich schnell, wie das Compilerfeedback zur syntaktischen Korrektheit.

Doch es geht schneller. Intellisense und Hintergrundcompilation geben bereits beim Tippen Feedback zur Verwendung der Programmiersprache. Und Tools wie NCrunch haben diese Schnelligkeit inzwischen auch für Unit Tests hergestellt.

Die Codesauberkeit hingegen hinkt bis jetzt immer noch weit hinterher. Denn nun hat Ralf Westphal mit Robin Sedlaczek eine Visual Studio Extension entwickelt, die die Codesauberkeit unmittelbar spürbar macht, während Sie tippen.

Mit der Force Feedback Programming (FFP) Extension bekommen Sie Feedback auf zwei Sinneskanälen:

Visuelles Feedback: Die Extension

```
using System;
using System.Collections.Generic;
using equalidator.datamodel;

namespace equalidator.flowmodel
{
    public class Flatten_array
    {
        public void Process(object objectToFlatten)
        {
            if (objectToFlatten.GetType().IsArray)
            {
                Issue_descriptive_opening_fragment(objectToFlatten);
                Iterate_array_elements_in_all_dimensions((Array)objectToFlatten, new List<>());
                Issue_closing_fragment();
            }
            else
            {
                this.PassOn(objectToFlatten);
            }
        }

        internal void Issue_descriptive_opening_fragment(object objectToFlatten)
        {
            var t = objectToFlatten.GetType();
            var dimensionLengths = "";
            for (var d = 0; d < t.GetArrayRank(); d++)
            {
                if (d > 0) dimensionLengths += ",";
                dimensionLengths += ((Array)objectToFlatten).GetLength(d);
            }

            this.MetaObject(new OpeningFragment(string.Format("{0};{1};{2}", t, t.GetArrayRank(), dimensionLengths));
        }

        internal void Iterate_array_elements_in_all_dimensions(Array objectToFlatten, List<int> indexes)
        {
            var t = objectToFlatten.GetType();
            if (t.GetArrayRank() > indexes.Count)
            {
                for (var i = 0; i < objectToFlatten.GetLength(indexes.Count); i++)
                {
                    indexes.Add(i);
                    Iterate_array_elements_in_all_dimensions(objectToFlatten, indexes);
                    indexes.RemoveAt(indexes.Count - 1);
                }
            }
            else
            {
                this.Element(objectToFlatten.GetValue(indexes.ToArray()));
            }
        }

        private void Issue_closing_fragment()
        {
            this.MetaObject(new ClosingFragment());
        }
    }
}
```

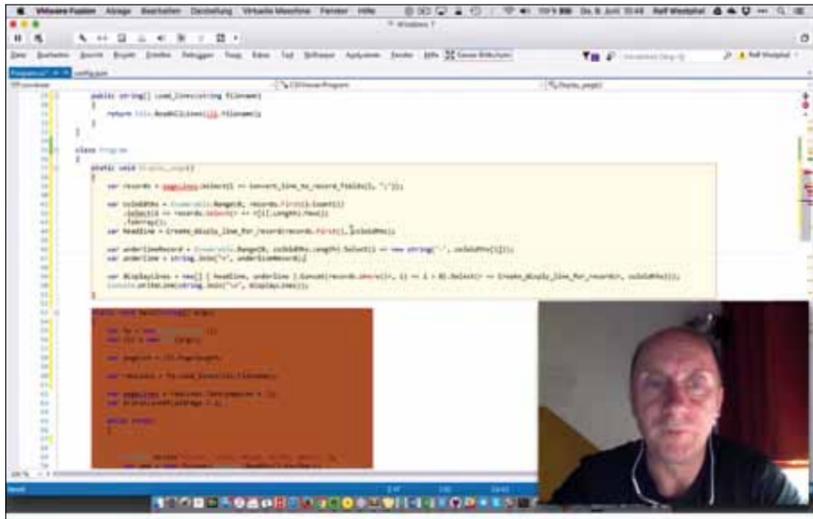
Abbildung 2: Die hellgelbe Hinterlegung deutet auf deutlich saubereren Code hin

färbt den Rumpf von Methoden nach dem Grad ihrer Sauberkeit ein. Sie sehen auf einen Blick, wie tief das Brownfield ist, durch das Sie waten. Nicht kleine Icons am Editorrand oder dezente Striche kennzeichnen Unsauberkeit, sondern der ganze Hintergrund einer Methode.

Taktiler Feedback: Die Extension macht die Sauberkeit einer Methode fühlbar „at your fingertips“, indem sie Ihre Tastatureingabe während des Schreibens von Code stört. Je unsauberer Ihr Code, desto häufiger die Störungen. Es fällt Ihnen dadurch immer schwerer, unsauberem Code weitere Unsauberkeit hinzuzufügen. Derzeit ist die Metrik, die die FFP

Extension zur Sauberkeitsbeurteilung heranzieht, sehr, sehr simpel: Sie zählt einfach die Zahl der Zeilen in einer Methode. Angesichts des Ergebnisses eines Experiments von Prof. Dag Sjøberg meint Ralf Westphal jedoch, dass die Zeilenzahl ein (zunächst) guter Startpunkt ist. Unsauberkeit lässt sich ohnehin nicht exakt bestimmen. Es geht um eine Näherung und dann um eine Tendenz. Viel wichtiger als eine genaue Messung von Unsauberkeit ist das Feedback. Wie kann ein Unsauberkeitswert spürbar gemacht werden?

Wenn unsauberer Code die Verständlichkeit bzw. Lesbarkeit beeinflusst, warum dann nicht die



VIDEO: https://youtu.be/v01jCip7_ro

Lesbarkeit tatsächlich reduzieren? Deshalb die Einfärbung des Methodenhintergrunds. Der Code wird – je nach Farbschema – immer schwerer zu entziffern. Dazu muss man kein Programmierer oder Domänenexperte sein. Man muss nur hinschauen. Das versteht sogar ein Laie.

Wenn unsauberer Code die Veränderbarkeit beeinflusst, warum dann nicht die unmittelbar reduzieren? Code inhaltlich nur schwer verändern zu können, beim Bug Fixing oder für neue Features, ist eine Sache. Diese Schwierigkeit bemerkt nur ein Programmierer. Sie entsteht zunächst auch nur in seinem Kopf. Was aber, wenn es tatsächlich physisch schwieriger wird, den Code zu verändern? Dann muss man nicht erst ein mentales Modell des Codes aufbauen, sondern man spürt beim Tippen, dass es schwieriger wird. Über Farbe kann man hinwegsehen. Doch wenn das

Tippen behindert wird, dann nervt Brownfield-Code im Hier und Jetzt.

Das ist die Theorie hinter Force Feedback Programming. Wir brauchen schnelleres Feedback, wenn wir unsere Strukturierungs- und Codierungspraxis ändern wollen. Visuelles und taktiles Feedback sollen helfen, Codesauberkeit unmittelbar spüren zu können.

DAS TOOL

Kooperationspartner, die bei der Umsetzung behilflich sein sollten, sprangen, wenn es an die Programmierung ging, schnell ab.

Nach Lektüre eines Artikels von Robin Sedlaczek über die VS Extension Programmierung glaubte Ralf Westphal wieder einen geeigneten Kandidaten für eine gute Zusammenarbeit gefunden zu haben und schrieb ihn an. – Gemeinsam sind beide nun über die Theorie hinausgekommen.

Robin Sedlaczek ist der VS Extension Experte und Codierer, Ralf Westphal ist der Product Owner und Kommunikator. Gemeinsam gehen sie in kleinen Inkrementen stetig vorwärts. Es gibt ein Backlog und noch mehr Ideen. Es gibt ein Github-Repository und einen automatischen Buildprozess.

Noch wichtiger aber für Sie: Es gibt inzwischen eine funktionierende FFP Extension und Sie können die Releases aus dem Repo herunterladen. Ausprobieren ausdrücklich erwünscht (Achtung: Damit eine Installation auch ein Ergebnis zeigt, muss eine Konfigurationsdatei existieren. Lesen Sie hier, was dafür noch zu tun ist. Es gibt auch eine Beispieldatei.)

Damit Sie aber nicht die Katze im Sack installieren, finden Sie unter dem folgenden Link eine Demonstration von Force Feedback Programming mit Stand Juni 2016:

https://youtu.be/v01jCip7_ro

Im Video refaktoriert Ralf Westphal eine kleine Brownfield-Anwendung. Hier und da kommentiert er das, aber die eigentliche Refaktorisierung läuft in Zeitraffer ab. Es geht nicht darum, wie er refaktoriert, sondern dass er es macht und dass Sie das insbesondere anhand der Entwicklung der Codefarbe verfolgen können.

Beim Refaktorisieren leitet vor allem das visuelle Feedback. Wenn es um den Einbau von Veränderungen geht, wird auch taktiles Feedback wichtig. Davon ein andermal mehr.

Was halten Sie von der Idee? Probieren Sie die Extension doch einmal aus und geben Sie uns Feedback.

WINDOWS REMOTE ARDUINO



Autor: TAM HANNA

Microsoft's Versuche in der Welt der Prozessrechner verliefen bisher im Großen und Ganzen erfolglos: Managed Code ist für Echtzeitaufgaben nun mal ungeeignet. Mit Windows Remote Arduino geht man in Redmond einen neuen Weg.

Die dahinterstehende Idee ist Elektronikern unter dem Begriff des kombinatorischen Prozessrechners bekannt. Die zu lösende MSR-Aufgabe wird in mehrere Teile zerlegt: Zeitkritische Jobs laufen auf einem dafür geeigneten Prozessrechner, während Steuerung und Anzeige anderswo erledigt werden.

Im Fall von Microsofts API für den Arduino wird dabei das Firmata-Protokoll genutzt, was zur Abbildung der gezeigten Architektur führt.

BESCHAFFUNG DER MODULE

Die per NuGet angebotenen Pakete sind nicht sonderlich aktuell und enthalten einige ärgerliche Fehler. Der Autor empfiehlt aus praktischer Erfahrung, die beiden Git-Repositories <https://github.com/ms-iot/remote-wiring.git> und <https://github.com/ms-iot/serial-wiring.git> herunterzuladen. Nach dem Beschaffen des Codes müssen die folgenden drei Projekte in die Solution aufgenommen werden:

```
Microsoft.Maker.win10/Microsoft.Maker.RemoteWiring/
Microsoft.Maker.RemoteWiring.vcxproj
Microsoft.Maker.win10/Microsoft.Maker.Firmata/Microsoft.
Maker.Firmata.vcxproj
Microsoft.Maker.Serial.win10/Microsoft.Maker.Serial.
vcxproj
```

Wir wollen für die folgenden Schritte einen per USB angeschlossenen Arduino verwenden. Als Entwicklungssystem dient eine unter Windows 8.1 laufende Workstation, die ein Windows 10-Tablet der Bauart TecLast X98 Pro fernsteuert. Die anfangs angebotene Unterstützung von Windows 8.1 wurde von Microsoft mittlerweile aufgegeben: Windows Remote Arduino lässt sich nur noch mit Windows 10 benutzen. Der Lohn dieser Mühen ist

allerdings, dass die API - bis zu einem gewissen Grad - auch am Windows Phone und theoretisch sogar auf der Xbox One funktioniert.

Zur Kommunikation mit dem Prozessrechner sind zwei Klassen erforderlich, die wir als Membervariable des Formulars anlegen:

```
public sealed partial class MainPage : Page
{
    UsbSerial myUSB;
    Microsoft.Maker.RemoteWiring.RemoteDevice myArduino;
```

Im Konstruktor wird die Verbindung aufgebaut. Hier sind zweierlei Sachen relevant: erstens nimmt die RemoteDevice-Klasse eine Zweitklasse entgegen, die die Art der Verbindung beschreibt. Zweitens sind im Fall von USB Informationen über das Zielgerät erforderlich.

Die von uns hier abgedruckten Werte sind für einen Arduino Uno vorgesehen - wer einen anderen Prozessrechner verwenden möchte, findet weitere Informationen in der Systemsteuerung:

```
public MainPage()
{
    this.InitializeComponent();
    myUSB = new UsbSerial(„VID_2341“, „PID_0043“);
    myArduino = new Microsoft.Maker.RemoteWiring.
RemoteDevice(myUSB);
    myUSB.ConnectionEstablished +=
MyUSB_ConnectionEstablished;
    myUSB.ConnectionFailed +=
MyUSB_ConnectionFailed; ;
    myUSB.begin(115200, Microsoft.Maker.Serial.
SerialConfig.SERIAL_8N1);
}
```

Windows Remote Arduino funktioniert immer dann am besten, wenn man die Eventgetriebenheit des Produkts berücksichtigt. Aus diesem Grund warten wir den erfolgreichen Verbindungsaufbau per USB ab, um dort einen weiteren Eventhandler einzuschreiben, der nach dem erfolgreichen Verbindungsaufbau zum Prozessrechner aktiviert wird:

ppedv
BUSINESS INTELLIGENCE TRAININGS
 INTEGRATION SERVICES & ANALYSIS SERVICES REPORTING SERVICES POWER BI
 PPEDV.DE/SQL

ppedv AG Marktstr. 13b, 84469 Bogen, Vorstand: Hannes Proshuber, Handelsregister Traunstein, HRG 27033, St.Nr. 131/4542

```
private void MyUSB_ConnectionEstablished()
{
    myArduino.DeviceReady += MyArduino_DeviceReady;
}
}
```

Kenner klassischer Prozessrechner müssen an dieser Stelle umdenken. Ob der Realisierung durch Firmata erfolgt das Lesen der Eingangspins auf eine neuartige Art und Weise: der Arduino überwacht seine Pins permanent, und sendet eine Meldung in Richtung der Workstations, wenn ein relevantes Ereignis aufgetreten ist.

Die auf der Workstation befindlichen Lesefunktionen retournieren - analog zum Bluetooth-Stack von Windows 10 - sofort, und nutzen dabei Informationen aus dem vorgehaltenen Cache.

```
private void MyArduino_DeviceReady()
{
    myArduino.DigitalPinUpdated +=
MyArduino_DigitalPinUpdated;
}
private void MyArduino_DigitalPinUpdated(byte pin,
PinState state){ }
```

Auf Seiten des Arduinos nutzen wir den normalen StandardFirmata-Sketch, der unter Beispiele -> Firmata -> StandardFirmata in der Arduino-IDE bereitsteht. Achten Sie darauf, die Bandbreite bzw. Kommunikationsgeschwindigkeit anzupassen:

```
void setup()
{
    Firmata.setFirmwareVersion(FIRMATA_FIRMWARE_MAJOR_VERSION,
FIRMATA_FIRMWARE_MINOR_VERSION);
    Firmata.begin(115200);
}
}
```

Der Zugriff auf externe Hardware setzt Berechtigungen voraus. Aufgrund eines Fehlers im Manifesteditor müssen Sie den folgenden Block von Hand einfügen - ersetzen Sie die schon vorhandene Version durch den hier abgedruckten Code:

```
<Capabilities>
  <Capability Name="internetClient" />
  <DeviceCapability Name="serialcommunication">
    <Device Id="any">
      <Function Type="name:serialPort"/>
    </Device>
  </DeviceCapability>
</Capabilities>
```

An dieser Stelle noch eine kleine Anmerkung: Der hier gezeigte Aufbau lässt sich im Großen und Ganzen für Bluetooth recyclieren. Der einzige nennenswerte Unterschied ist, dass statt USBSerial eine Instanz der Klasse BluetoothSerial eingesetzt wird. Diese braucht einen Zeiger auf ein Element einer DeviceInformationCollection: dessen Beschaffung erfolgt wie bei der normalen Bluetooth-Programmierung.

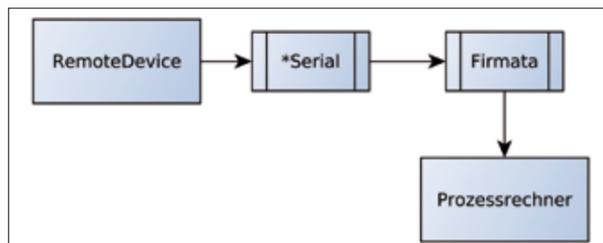


Abbildung 1: Der Aufbau ist medienagnostisch

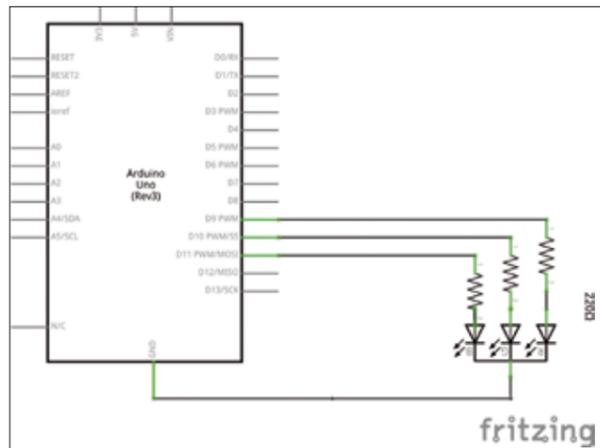


Abbildung 2: Die RGB-LED besteht aus drei Unterdioden

ABER NUN LOS

Nach diesen Vorbereitungsmaßnahmen ist es an der Zeit, den Prozessrechner unter Feuer zu setzen. Als erstes Beispiel wollen wir eine RGB-LED ansteuern. Verbinden Sie diese wie in Abbildung zwei gezeigt über Widerstände mit den Ausgangspins des Arduinos. Die Widerstandswerte sind dabei nicht kritisch: sofern sie in einem vernünftigen Rahmen bleiben (Datenblatt), verändert sich nur die Helligkeit der Diode.

Die meisten Arduinos bringen eine PWM-Engine mit, die für das Ansteuern von RGB-LEDs geradezu prädestiniert ist. In unserem Fall sieht der notwendige Code so aus:

```
private void MyArduino_DeviceReady()
{
    myArduino.DigitalPinUpdated +=
MyArduino_DigitalPinUpdated;
    myArduino.pinMode(9, PinMode.PWM);
    myArduino.pinMode(10, PinMode.PWM);
    myArduino.pinMode(11, PinMode.PWM);
    myArduino.analogWrite(11, 240); //R
    myArduino.analogWrite(10, 0); //G
    myArduino.analogWrite(9, 240); //B
}
}
```

Kenner der Arduino-Programmierungsumgebung reiben sich an dieser Stelle wahrscheinlich verwundert die Augen. Das RemoteDevice-Objekt ist eine mehr oder weniger

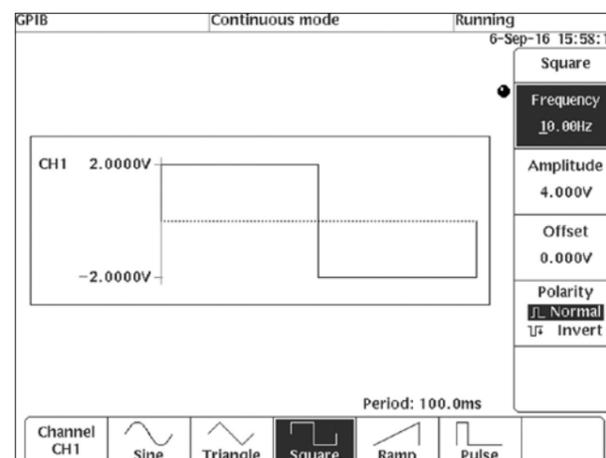


Abbildung 3: Achten Sie darauf, die Signalspannung an ihren Arduino anzupassen: Nicht alle vertragen 4 bzw. 5 V

vollständige Reimplementierung der Wiring-API, die in der normalen Arduino-Programmierung zum Einsatz kommt. Der einzige Unterschied besteht darin, dass die Ausführung nun - wie in Abbildung 1 gezeigt - über die Firmata-Verbindung erfolgt.

Die Erfassung dieser Latenz ist insofern schwierig, als der Arduino Leseanfragen aus dem Cache bedient. Als erstes Beispiel wollen wir eine von einem Funktionsgenerator (Tektronix AWG2021) erzeugte Rechteckquelle analysieren. Dazu verbinden Sie den Prozessrechner mit der Signalquelle. Statt dem hier verwendeten Tektronix AWG2021 können Sie auch andere Funktionsgeneratoren nutzen. Die Wellenform ist eine normale Rechteckwelle: Abbildung 3 zeigt die im folgenden verwendeten Grundeinstellungen.

Zur präzisen Zeitmessung wird in der Universal Windows Plattform gerne die Stopwatch-Klasse eingesetzt. Wir beschaffen den Startzeitpunkt im Rahmen von DeviceReady, und setzen den Ausgangsmodus des Pins Nummer zwei ebenda auf Eingang.

Im Interesse des Funktionsgenerators sollten Sie zumindest diesen Teil des Codes deployen, bevor Sie den Ausgang mit dem Prozessrechner verbinden: Ausgleichsströme aufgrund eines Ausgangspins sind sowohl für Prozessrechner als auch für Funktionsgenerator ungesund:

```
private void MyArduino_DeviceReady()
{
    startedAt = Stopwatch.GetTimestamp();
    ...
    myArduino.pinMode(2, PinMode.INPUT);
}
}
```

Als erste Frage müssen wir nun feststellen, wann die Methode MyArduino_DigitalPinUpdated aufgerufen wird: Meldet sich der Arduino nur bei Änderungen, oder setzt er immer wieder einen Ping ab. Dies lässt sich dadurch prüfen, dass man bei jedem Aufruf der Funktion einen String erweitert. Nach dem dreißigsten Aufruf findet sich eine tautologische Anweisung, die als Basis für einen Breakpoint dient:

```
private void MyArduino_DigitalPinUpdated(byte pin,
PinState state)
{
    if (myUpdateCounter < 30 && pin==2)
    {
        long temp = Stopwatch.GetTimestamp()
- startedAt;
        float outVal = (float)temp / (float)Stopwatch.
Frequency;
        startedAt = Stopwatch.GetTimestamp();
        myOutStore += „Neuer Zustand:“ +
state.ToString() + „ , Delta T „ + outVal + „\n“;
        myUpdateCounter++;
    }
    else {
        myUpdateCounter = myUpdateCounter;
    }
}
}
```

Bei einer Betriebsfrequenz von 10 Hz findet sich im Debugger sodann folgender String:

```
Neuer Zustand:HIGH , Delta T 0.0468891
Neuer Zustand:LOW , Delta T 0.05620804
... (nur bei Toggle)
```

Damit ist bewiesen, dass der Eventhandler nur bei Änderungen des Pins aufgerufen wird. Diese Information können wir im nächsten Schritt zur Ermittlung der Latenz nutzen. Wie im Fall des vorigen Programms unterteilen wir den Arbeitsprozess auch diesmal in zwei Schritte. Im ersten Durchlauf ermitteln wir die zwischen den einzelnen Zustandsänderungen verstrichene Zeit unter Nutzung der GetTimestamp-Methode:

```
private void MyArduino_DigitalPinUpdated(byte pin,
PinState state)
{
    if (myUpdateCounter < 30 && pin==2)
    {
        myOutputs[myUpdateCounter]=
Stopwatch.GetTimestamp() - startedAt;
        startedAt = Stopwatch.GetTimestamp();
        myUpdateCounter++;
    }
}
}
```

Die eigentliche Auswertung der Daten erfolgt - im Großen und Ganzen - unter Nutzung elementarer Statistikfunktionen. Die Frequenz ist dabei definiert als ein Hertz, das laut SI-Einheitensystem als Eins durch Sekunde definiert ist. Die Multiplikation mit 0,5, also eine Division durch den Faktor zwei, ist erforderlich, weil unser Prozessrechner den High- und den Low-Teil der Wellenform separat ausweist.

```
else {
    long max=0; long min=9999999;
    long accu = 0;
    for (int i = 0; i < 30; i++)
    {
        if (myOutputs[i] > max) max = myOutputs[i];
        if (myOutputs[i] < min) min = myOutputs[i];
        accu += myOutputs[i];
    }
    accu /= 30;
    double tAvg = (double)accu / Stopwatch.Frequency;
    double fAvg = 1 / tAvg *0.5;
}
myUpdateCounter = myUpdateCounter;
}
}
```

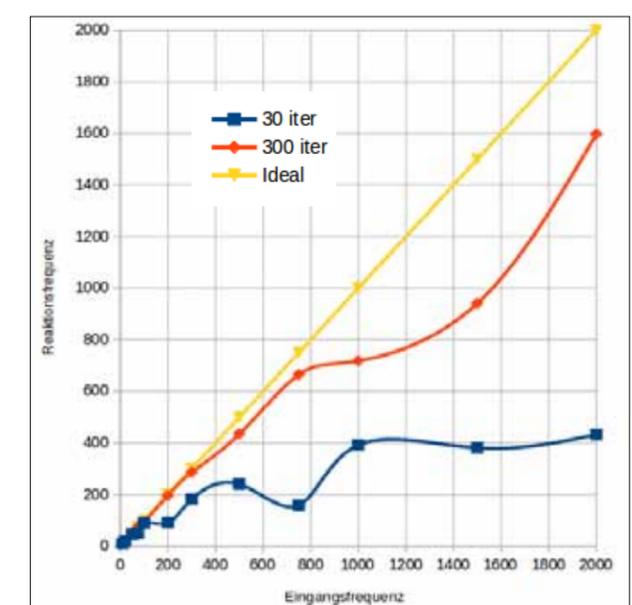


Abb. 4: Bei längerer Sampling-Dauer wird die Genauigkeit höher

Wir ließen den folgenden Test sowohl mit 30 als auch mit 300 Wiederholungen durchlaufen, um so Informationen über das Devianzverhalten zu erreichen. Abbildung 4 ist ein zusammengefasstes Diagramm, das die gemessene und die eingegebene Frequenz gegenüberstellt.

Anhand der Nichtlinearität beider Kurven lässt sich feststellen, dass das Erfassungsverhalten nicht sonderlich genau ist. Wenn es auf harte Echtzeit ankommt, ist schon ab rund 50 Hz Schluss: Geht es um langfristige Erfassung, so verhält sich das System besser. Klar ist jedenfalls, dass diese Vorgehensweise nicht zur Realisierung von schnellen Systemen taugt.

MIT DAMPF UND OHNE AUFENTHALT!

Die in der Einleitung erwähnten kombinatorischen Prozessrechnersysteme sind dadurch definiert, dass die Aufgaben aufgeteilt sind: Im Fall des vorhergehenden Beispiels war dies insofern nicht gegeben, weil jede Wellenformänderung eine Interaktion auf beiden Teilen des Prozessrechners hervorgerufen hat.

Wie bei der Nutzung von Beschreibungssprachen wie QML gilt auch beim Prozessrechnerdesign, dass das Spielen von Ping Pong in höchstem Maße ungesund ist: Man sollte als Entwickler davon ausgehen, dass jeder Datenaustausch zwischen den beiden Teilen Schmerzen verursacht.

Das Firmata-Protokoll bietet mit den SysEx-Befehlen ein Interface an, mit dem man beliebige eigene Kommandos realisieren kann. Dazu ist allerdings eine Modifikation der StandardFirmata-Firmware erforderlich. Die für die Abarbeitung von SysEx-Befehlen zuständige Funktion sieht folgendermaßen aus:

```
void sysexCallback (byte command, byte argc, byte *argv)
{
    byte mode;
    . . .

    switch (command) {
        case I2C_REQUEST:
            mode = argv[1] & I2C_READ_WRITE_MODE_MASK;
            . . .
    }
}
```

Es hat sich in der Praxis als Standard etabliert, eigene Befehle einfach über dem ersten I2C-Befehl zu platzieren. Damit stellt sich allerdings die Frage nach den Kommandobytes. In Firmata.h findet sich folgende Passage, die den für den Benutzer zur Verfügung stehenden Bereich hinreichend beschreibt:

```
// extended command set using sysex (0-127/0x00-0x7F)
/* 0x00-0x0F reserved for user-defined commands */
#define SERIAL_MESSAGE      0x60
#define ENCODER_DATA        0x61
#define SERVO_CONFIG        0x70
```

Als kleines Programmierbeispiel wollen wir hier einen Pattern-Generator erzeugen, der an den Ausgangspins des Arduino eine vorher definierte Bytesequenz ausgibt. Diese früher in dedizierter Hardware verfügbaren Geräte sind immens nützlich - wer viel im Labor verbringt, sollte vielleicht etwas Zeit in die Perfektionierung des Produkts stecken.

Firmata zeigt sich an dieser Stelle insofern haarig, als es auf dem MIDI-Protokoll basiert. Dort ist festgelegt,

dass die Nutzdaten nur sieben Bits in Anspruch nehmen dürfen. Die in Firmata.h befindliche Deklaration ist deshalb irreführend:

```
#define MAX_DATA_BYTES 64 // max number of data bytes in
incoming messages
Immerhin findet sich unter https://github.com/firmata/
protocol/blob/master/protocol.md#string auch eine Erklärung,
wie man mit Firmata Strings überträgt. Wir drucken
den relevanten Abschnitt des Headers hier einfach ab:
0 START_SYSEX          (0xF0)
1 STRING_DATA          (0x71)
2 first char LSB
3 first char MSB
. . .
... additional bytes up to half the buffer size - 3
(START_SYSEX, STRING_DATA, END_SYSEX)
N END_SYSEX (0xF7)
```

Im nächsten Schritt muss ein globaler Speicherbereich für die auszugebenden Signalinformationen angelegt werden. Wir nutzen hier ein 16 Byte langes Array, das unter dem schon vorhandenen Banner angelegt wird:

```
/*=====
 * GLOBAL VARIABLES
 *=====*/
byte myStore[16];
. . .
```

Eingehende SysEx-Kommandos werden - im Großen und Ganzen - nach der von C-Parametern bekannten Konvention verarbeitet. argc legt die Länge der angelieferten Informationen fest, während argv die eigentlichen Daten enthält.

Unser Produkt hat hier die Aufgabe, die von der Windows-Workstation angelieferten Elemente in den Speicherbereich zu schreiben. Zudem wird eine Flag gesetzt, die für das Aktivieren der Wellenformausgabe zuständig ist.

```
#define PATTERN_MESSAGE  0x00
void sysexCallback(byte command, byte argc, byte *argv)
{
    . . .
    switch (command) {
        case PATTERN_MESSAGE:
            for(int i=0;i<16;i++){
                myStore[i]=argv[i];
            }
            Firmata.sendString(„Nachricht erhalten“);
            break;
    }
    . . .
}
```

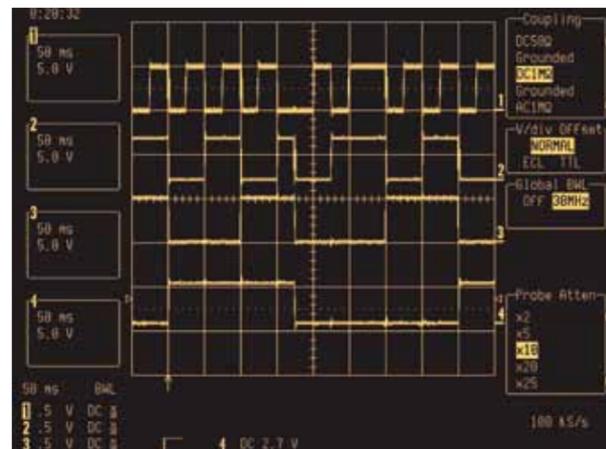


Abbildung 5: Alles ist da, wo es sein soll



Abbildung 6: Der Jitter ist mit freiem Auge erkennbar

Auf Codeseite setzen wir hier - schon aus Gründen der Einfachheit - auf ein zweites Projekt, dessen Aufbau dem vorhergehenden Beispiel ähnelt. Im Konstruktor wird nun die Firmata-Instanz exponiert, um direkten Zugriff zu ermöglichen:

```
public MainPage()
{
    this.InitializeComponent();
    myFirmata = new UwpFirmata();
    myFirmata.StringMessageReceived +=
MyFirmata_StringMessageReceived;
    myUSB = new UsbSerial(„VID_2341“, „PID_0043“);
    myArduino = new Microsoft.Maker.RemoteWiring.
RemoteDevice(myFirmata);
    myFirmata.begin(myUSB);
    myUSB.ConnectionEstablished +=
MyUSB_ConnectionEstablished;
    myUSB.ConnectionFailed += MyUSB_ConnectionFailed; ;
    myUSB.begin(115200, Microsoft.Maker.Serial.Serial-
Config.SERIAL_8N1);
}
```

Beim Entwickeln von Prozessrechnerschaltungen ist es sinnvoll, wenn das Zielsystem den Eingang von Nachrichten quittiert. Aus diesem Grund schreiben wir den StringMessageReceived-Eventhandler ein, der vom weiter oben abgedruckten Code mit einer Message versorgt wird.

Die Tautologieanweisung dient - wie immer - als Basis für einen Breakpoint.

```
private void MyFirmata_StringMessageReceived(Uw
pFirmata caller, StringCallbackEventArgs argv)
{
    var content = argv.GetString();
    content = content;
}
```

Der zweite Unterschied ist, dass die Pins nun als Ausgänge deklariert werden:

```
private void MyArduino_DeviceReady()
{
    myArduino.pinMode(2, PinMode.OUTPUT);
    myArduino.pinMode(3, PinMode.OUTPUT);
    myArduino.pinMode(4, PinMode.OUTPUT);
    myArduino.pinMode(5, PinMode.OUTPUT);

    myArduino.digitalWrite(2, PinState.HIGH);
    myFirmata.sendSysex((byte)0x00, new byte[] {
0, 32, 16, 48, 48, 8, 40, 24, 56, 4, 36, 20, 52, 12, 44,
28, 60 }.AsBuffer());
}
```

Die Zuweisung der Pins ist nicht zufällig: Bei der seriellen Kommunikation sollte man die Pins Null und Eins

MODU-WAS?

Viele (rüstungs-)elektronische Probleme lassen sich in der Modulationsdomäne einfacher darstellen. Der Autor bietet in seinem YouTube-Kanal unter www.youtube.com/watch?v=IBLEfVUVGyU ein Video an, das die Grundlagen dieser zugegebenermaßen seltsamen Darstellungsart erklärt.

freihalten, da sie mit dem UART verbunden sind. Das gilt im Übrigen auch für die zu sendenden Bits: Sie realisieren eine charakteristische Stufenfolge, die man am Bildschirm des Oszillographen leicht erkennt.

LANGE LAUFEND!

Damit stellt sich nur noch eine Frage: Unser Pattern-Generator kann keine Endlosschleife benutzen, da er sonst das Funktionieren des restlichen Firmata-Stacks bedrohen würde. Die Lösung für das Problem besteht darin, den eigenen Code in den von Firmata vorgegebenen Eventhandler einzubinden:

```
void loop()
{
    byte pin, analogPin;

    if(receivedPattern && myNextMilli < millis())
    {
        myCounter++;
        if(myCounter==16)myCounter=0;
        digitalWrite(2, myStore[myCounter]&4);
        digitalWrite(3, myStore[myCounter]&8);
        digitalWrite(4, myStore[myCounter]&16);
        digitalWrite(5, myStore[myCounter]&32);
        myNextMilli+=25;
    }
}
```

An dieser Stelle sind zwei Elemente interessant: Erstens nutzen wir den von Millis zurückgegebenen Wert, um die Schleife nicht allzu oft abzuarbeiten. Dies ist vor allem deshalb wichtig, weil wir sonst zu viel Ressourcen verbrennen würden.

Das eigentliche Ermitteln der auszugebenden Bytes erfolgt über klassische Bitoperationen. Die Zahlen 4, 8, 16 und 32 entsprechen jeweils einem gesetzten Bit: wer sich viel mit Mikrocontrollern beschäftigt, sollte sich die die Potenzen der Zahl Zwei einprägen.

Unser kleiner Symbolgenerator ist einsatzbereit. Abbildung sechs zeigt das auf einem Digitalspeicheroszillographen teledyne'scher Bauart entstehende Schirmbild - Besitzer andere Oszillographen können ein im Großen und Ganzen ähnliches Schirmbild erwarten.

In der Modulationsdomäne zeigt sich derweil, dass die Struktur mit Firmata trotz allem nicht unproblematisch ist - das in Abbildung 6 abgedruckte Schirmbild stammt übrigens von einem HP 53310A.

FAZIT

Auch wenn erfahrene Elektroniker auf dieses Feature sicher nicht gewartet haben: Freude macht seine Nutzung trotzdem. Wer die Aufgaben richtig verteilt, erspart die Implementierung eines Kommunikationsprotokolls - selbst trivialste Netzwerkprotokolle bergen in der Praxis mehr Tücken, als man auf den ersten Blick erwartet.

WEB-APIs, DIE SIE BISHER VERMUTLICH NICHT KANNTEN

Autor: MAX SCHWEIGERDT

Es gibt Hunderte von APIs, welche die unterschiedlichsten Aufgaben erledigen oder vereinfachen und so manche API ist auf jeden Fall sinnvoller als eine andere. Dieser Artikel gewährt einen Einblick in weniger bekannte APIs, die aber für den einen oder anderen Sinn machen.

PAGE VISIBILITY

Die erste API, die vorgestellt wird, definiert ein Mittel, um die Sichtbarkeit einer Seite zu bestimmen. Ohne den Sichtbarkeit-Zustand haben Web-Developer die Seiten so entwickelt, als ob sie immer sichtbar wären. Klar muss man sich nun über Runtime-Entscheidungen Gedanken machen, aber dadurch sind Entwickler nun in der Lage Maschinenressourcen und Energieeffizienz zu optimieren.

Das `document.visibilityState` Attribut kann folgende Zustände haben:

- **visible**, wenn der User-Agent nicht minimiert und doch die Vordergrund Registerkarte ist
- **hidden**, wenn der User-Agent minimiert, in einen Hintergrund Tab oder der Bildschirm gesperrt ist
- **prerender**, wenn das Dokument im prerender-Modus und gerade nicht sichtbar ist

Das dazugehörige `Event visibilitychange` wird ausgelöst, wenn sich der Wert von `visibilityState` ändert.

Anwendung:

```
document.addEventListener('visibilitychange',  
handleVisibilityChangeEvent, false);
```

```
function handleVisibilityChangeEvent() {  
  switch (document.visibilityState) {  
    case 'prerender':  
      document.title = 'pre-rendering';  
      break;  
    case 'hidden':  
      document.title = 'hidden';  
      break;  
    case 'visible':  
      document.title = 'visible';  
      break;  
  }  
}
```

Aus historischen Gründen wird die Unterstützung für das `document.hidden` Attribut beibehalten, Entwickler sollten wenn möglich `document.visibilityState` einsetzen.

CLIPBOARD.JS

Clipboard.js nutzt die Web-API Selection und `execCommand`, denn um einen Text in die Zwischenablage zu kopieren sollte es nicht hunderte von Schritten erfordern. Deswegen gibt es das schlanke Clipboard.js (10KB) welches ganz ohne Flash auskommt.

Nachdem man das Script lokal eingebunden oder aus einem CDN-Anbieter geladen hat, muss man lediglich das Clipboard durch einen DOM-Selektor, HTML-Element oder eine Liste von Elementen instanziiieren.

```
var clipboard = new Clipboard(„.clipBtn“);
```

Intern muss für jedes Auslöser-Element, das mit dem Selektor übereinstimmt, ein Ereignis-Listener befestigt werden. Wenn Sie sehr viele Elemente haben, kann dieser Vorgang enorm viel Speicherplatz verbrauchen. Aus diesem Grund verwendet man besser die Library Ereignis-Delegation, die mehrere Event-Listener durch einen einzigen ersetzt.

Ein häufiger Anwendungsfall ist, den Inhalt von einem anderen Element zu kopieren.

Dazu muss man nur ein `data-clipboard-target` Attribut zu unserem Auslöser-Element hinzufügen.

Der Wert des data-Attributes muss auf ein gültiges Ziel referenzieren.

Beim Kopieren können Sie das `data-clipboard-action` Attribut weglassen, da "copy" der Standardwert ist.

```
<!-- Ziel -->  
<p id="copy1">  
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
</p>  
  
<!-- Auslöser -->  
<button class="clipBtn"  
  data-clipboard-target="#copy1">Kopieren</button>
```

Sie benötigen nicht einmal ein anderes Element um deren Inhalt zu kopieren. Sie können auch nur ein `data-clipboard-text` Attribut Ihrem Auslöser-Element zuweisen

und der String dieses Attributes wird in die Zwischenablage kopiert.

```
<button class="clipBtn" data-clipboard-text="Zu kopierender Text">Kopieren</button>
```

Möchten man stattdessen etwas ausschneiden, so muss `data-clipboard-action` der Wert "cut" übergeben werden. Das Ausschneiden funktioniert nur bei den Elementen `<input>` oder `<textarea>`!

```
<!-- Ziel -->  
<textarea id="cut1" cols="40" rows="10" >  
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
</textarea>  
  
<!-- Auslöser -->  
<button class="clipBtn" data-clipboard-target="#cut1"  
  data-clipboard-action="cut">  
  Ausschneiden  
</button>
```

Benutzerdefinierte Events wie `success` und `error` erlauben es Benutzern Feedback zu geben oder erfassen was beim Kopieren bzw. Ausschneiden ausgewählt worden ist. Das erlaubt eine eigene Logik zu implementieren.

```
clipboard.on('success', function (e) {  
  
  console.log(„Trigger: „, e.trigger);  
  console.log(„Action: „, e.action);  
  console.log(„Text: „, e.text);  
  
  e.clearSelection();  
});  
  
clipboard.on('error', function (e) {
```

```
  console.error(„Action:“, e.action);  
  console.error(„Trigger:“, e.trigger);  
});
```

Safari ist der einzige Browser der die Funktionalität von `clipboard.js` nur teilweise unterstützt. Dafür wird der Nutzer in Safari, durch eine Info-Box aufgefordert CMD-C zu drücken, um den markierten Text zu Kopieren.

ONLINE STATE

Diese API gibt Verfügbarkeitsinformationen an. Zum Beispiel ob sich der Browser mit einem Netzwerk oder dem Internet verbinden konnte.

`window.navigator.onLine`;

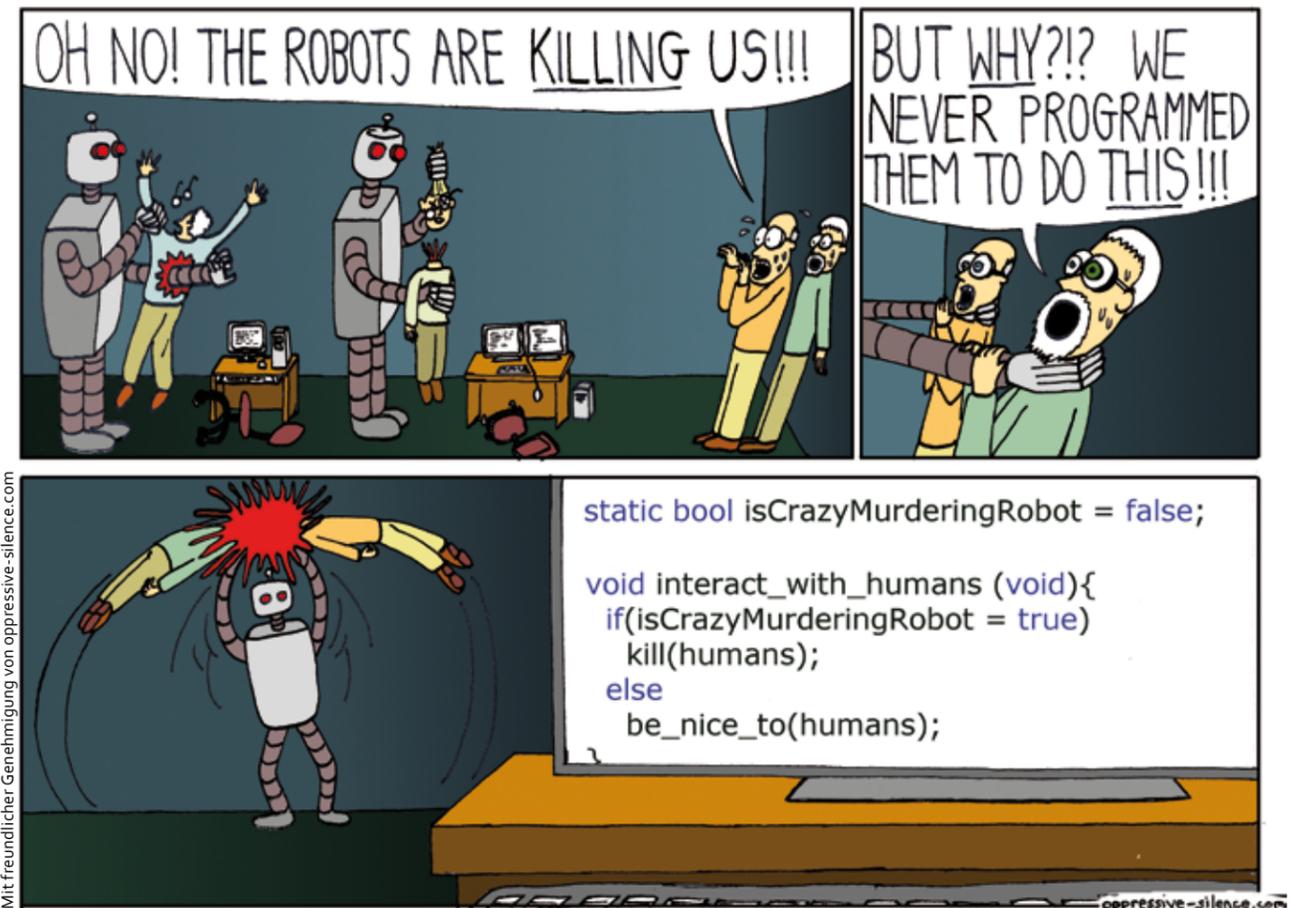
Gibt `false` zurück, wenn der User-Agent *definitiv offline* ist (vom Netz getrennt).
Gibt `true` zurück, wenn der User-Agent *online sein kann*.

Die dazugehörigen `Events online` und `offline` werden ausgelöst, wenn sich der **Wert** dieses Attributs **ändert**.

```
window.addEventListener(„offline“, function(e) {  
  console.info(e.type);  
}, false);  
  
window.addEventListener(„online“, function(e) {  
  console.info(e.type);  
}, false);
```

Dieses Attribut ist unzuverlässig. Ein Gerät kann ohne Internetzugang mit einem Netzwerk verbunden werden. In solchen Fällen würde `navigator.onLine` den Wert `true` zurückliefern.

CARTOON





VON DER NATUR LERNEN

Es gibt Probleme, die sich nur durch wiederholtes Ausprobieren lösen lassen. Doch was, wenn die Anzahl an möglichen Lösungen dermaßen hoch wird, dass das zu lange dauert? Genetische Algorithmen ermöglichen das effiziente Annähern an die optimale Lösung.

Autor: GOLO RODEN

Viele Probleme lassen sich leicht lösen. Das gilt immer dann, wenn sich die Lösung direkt und effizient berechnen lässt. Doch für einige Probleme gilt das nicht: Es gibt schlichtweg keinen effizienten Algorithmus, um eine Lösung zu berechnen - oder zumindest ist keiner bekannt. Was dann bleibt, ist lediglich das Ausprobieren von potenziellen Lösungen. Je nachdem, wie viele Möglichkeiten es dabei allerdings gibt, kann es sehr lange dauern, bis man die tatsächliche Lösung findet.

Ein Beispiel dafür ist das Problem *Travelling Sales Person (TSP)* aus der theoretischen Informatik, das aber beispielsweise in der Logistik tatsächlich einen Bezug zur realen Welt aufweist. Bei TSP geht es darum, eine Reihe von Städten mit einer möglichst kurzen beziehungsweise günstigen Reiseroute zu besuchen.

Sollen beispielsweise die Städte Frankfurt, Köln, München, Stuttgart und Freiburg bereist werden, ergeben sich verschiedene mögliche Reiserouten (siehe Abbildungen 1 und 2):

-
- Frankfurt - Köln - München - Stuttgart - Freiburg
- Frankfurt - Köln - München - Freiburg - Stuttgart
- Frankfurt - Köln - Stuttgart - München - Freiburg
- Frankfurt - Köln - Stuttgart - Freiburg - München
- [...]
-

Die Komplexität des Problems steigt rasch, da man prinzipiell von jeder Stadt zu jeder anderen aufbrechen könnte. Anders gesagt: Bei n Städten gibt es in der ersten Stadt $n-1$ mögliche Reiseziele. In der zweiten Stadt sind es noch $n-2$, in der dritten $n-3$, und so weiter. Da die Anzahl der Optionen miteinander zu multiplizieren ist, ergeben sich insgesamt

$$n! = n * (n-1) * (n-2) * \dots * 3 * 2 * 1$$

Routen, die zu berechnen sind, wobei die Funktion $n!$ als Fakultät bezeichnet wird. Rechnet man die Anzahl der möglichen Reiserouten auf dem Weg aus, stellt man fest,

dass die Fakultät bereits für kleine n sehr schnell sehr groß wird:

n	$n!$	4	24	9	362880
-- --	5	120	10	3628800	
1	1	6	720	11	39916800
2	2	7	5040		
3	6	8	40320		

Eine clevere Lösung für das Problem kann sein, die Route möglichst entlang der West-Ost- oder der Nord-Süd-Achse zu planen. Das funktioniert jedoch nur, wenn tatsächlich die Strecke minimiert werden soll. Sind statt der Entfernung jedoch die Reisekosten relevant, funktioniert das nicht, weil mehrere lange Strecken dann unter Umständen deutlich günstiger sein können als eine kurze.

Beispielsweise kann der Hin- und Rückflug von Köln nach München auf Grund eines Sonderangebots günstiger sein als die einfache Strecke von Stuttgart nach München mit der Bahn. Erschwerend kommt noch hinzu, dass Reisepreise nicht symmetrisch sind, der Preis der Strecke von Stuttgart nach München also gegebenenfalls ein anderer ist als der von München nach Stuttgart.

Bei gerade einmal 15 Städten gibt es bereits über eine Billion Möglichkeiten. Sie alle zu berechnen, um die kürzeste beziehungsweise günstigste herauszusuchen, ist unmöglich. Daher bleibt dann nichts anderes übrig als zu raten: Vermutlich wird man auf dem Weg nicht die beste Route erhalten, die schlechteste wird es aber auch nicht sein.

Die naheliegende Frage an der Stelle lautet, wie sich die geratene Route derart verbessern lässt, dass

man mit überschaubarem Aufwand zumindest zu einer halbwegs vernünftigen Lösung kommt.

DAS GRUNDLEGENDE VORGEHEN KLÄREN

Einen interessanten Ansatz dafür hat man sich von der Natur abgeschaut, der in der Informatik als *genetischer Algorithmus* bezeichnet wird. Die Idee ist, aus zwei geratene Lösungen eine neue zu entwickeln, indem man Teile der bestehenden Lösungen übernimmt. Dabei passiert prinzipiell das gleiche wie in der biologischen Fortpflanzung, wo von beiden Eltern jeweils ein Teil der Gene an das Kind weitergegeben werden.

Dazu muss zunächst eine Möglichkeit bestimmt werden, die Güte einer Lösung zu bewerten. Eine Funktion, die das vornimmt, wird in der künstlichen Intelligenz als Kosten- beziehungsweise Fitnessfunktion bezeichnet. Für die Bewertung einer Reiseroute bietet sich an, die Summe der Entfernungen beziehungsweise Reisekosten zu bilden.

Führt man das nicht nur für zwei Lösungen aus, sondern für viele, arbeitet man nicht nur mit einem Paar, sondern einer ganzen Population. Besonders praktisch an dem Vorgehen ist, dass sich das Finden neuer Lösungen auf dem Weg hervorragend parallelisieren lässt. Das ist vor allem deshalb sinnvoll, weil das Berechnen der Fitnessfunktion - im vorliegenden Beispiel also das Berechnen des Gesamtpreises der Reise - in der Regel der aufwändigste und teuerste

Schritt eines genetischen Algorithmus ist.

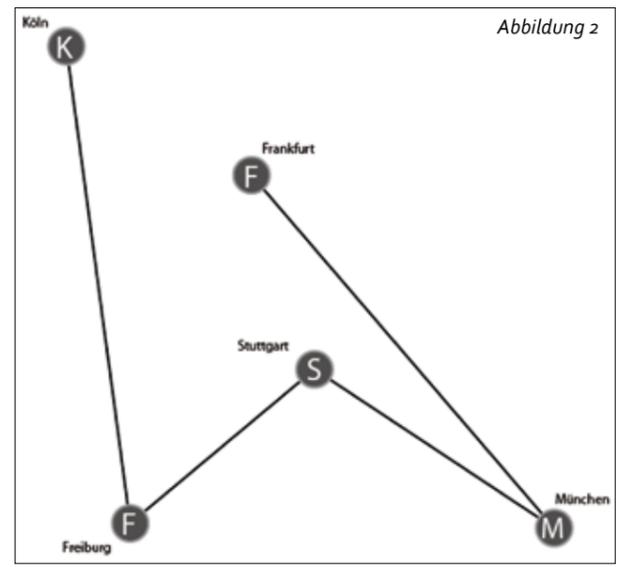
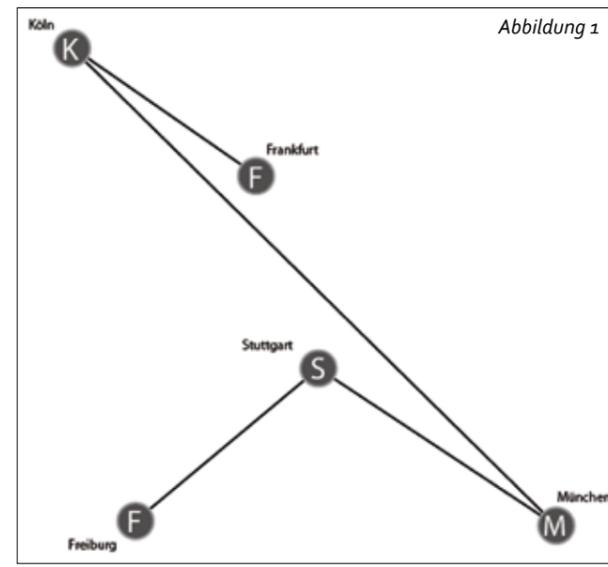
Nachdem die nächste Generation an Lösungen erzeugt wurde, werden die schlechtesten Mitglieder der Population ersetzt. So wird die Qualität der Lösungen in der Population nach und nach besser.

LÖSUNGEN ALS ELTERN AUSWÄHLEN

In der Regel will man das allerdings nicht für alle Lösungen ausführen, sondern nur für einige. Das wirft die Frage auf, wie man geeignete Lösungen als Eltern auswählt. Am nächstliegenden ist, die n besten Lösungen der Population auszuwählen. Das scheint logisch, bereitet in der Praxis jedoch zahlreiche Probleme: Um die n besten Lösungen auswählen zu können, müssen nämlich zunächst die Fitnessfunktion jeder einzelnen Lösung berechnet und die Ergebnisse anschließend sortiert werden.

Der Vorteil, das Verfahren parallelisieren zu können, wird damit ad Absurdum geführt, da das Sortieren letztlich einen unerwünschten Flaschenhals darstellt. Außerdem läuft man auf dem Weg Gefahr, sich in einer Sackgasse zu verrennen. Schließlich können die Eltern nur das weitergeben, was sie selbst besitzen. Neue Impulse von Außen fehlen.

Deshalb gibt es ein weiteres Verfahren, *Tournament Selection*, mit dem man Lösungen parallelisiert auswählen kann. Dazu werden zufällig zwei beliebige Lösungen ausgewählt und nur deren jeweilige



Fitnessfunktion berechnet und verglichen. Die bessere Lösung kommt eine Runde weiter und tritt erneut gegen eine andere zufällig ausgewählte Lösung an. Nach einer gewissen Anzahl an Runden bricht das Verfahren ab: Die Lösung, die den Vergleich für sich entschieden hat, wird als Elternteil ausgewählt.

Abgesehen davon, dass das Vorgehen Rechenzeit spart und sich parallelisieren lässt, ahmt es außerdem auch die Natur nach: Schließlich muss ein Lebewesen in der Natur auch nicht jeden Tag gegen das global gesehene „beste“ bestehen, sondern nur lokal gegen jene, die es zufällig antrifft. Außerdem verhindert das Vorgehen, dass stets die besten Lösungen als Eltern ausgewählt werden, was eine gewisse Varianz einbringt.

Es bleibt die Frage, wie groß eine Population sein, wie viele Lösungen als Eltern ausgewählt werden, und wie viele Runden für den Tournament-Selection-Algorithmus verwendet werden sollten. Hierfür gibt es keine festgelegte Formel, stattdessen muss man mit verschiedenen Werten experimentieren und sich an geeignete Werte herantasten. Jeff Heaton empfiehlt in seinem Buch [Artificial Intelligence for Humans, Volume 2: Nature-Inspired Algorithms] (<https://www.amazon.de/dp/1499720572>), mit einer Populationsgröße von 1000 und einer Rundenzahl von 5 zu starten.

NACHWUCHS DURCH REKOMBINATION

Sobald zwei Lösungen als Eltern ausgewählt wurden, lässt sich durch deren Rekombination eine neue Lösung als Nachwuchs erzeugen. Am einfachsten ist dabei das Vorgehen des sogenannten Splice-Crossovers. Dabei werden zunächst die genetischen Informationen des einen Elternteils kopiert. Anschließend wird ein Abschnitt der genetischen Informationen des zweiten Elternteils kopiert, der den betreffenden Abschnitt in der Kopie überschreibt. Sind beispielsweise die beiden Routen

~~~~~  
 Köln - Frankfurt - München - Stuttgart - Freiburg  
 Frankfurt - Köln - Stuttgart - München - Freiburg  
 ~~~~~

als Eltern ausgewählt worden, ergibt sich der Nachwuchs auf folgendem Weg: Zunächst wird die genetische Information des ersten Elternteils dupliziert. Damit weist der Nachwuchs zunächst folgende genetische Information auf:

~~~~~  
 Köln - Frankfurt - München - Stuttgart - Freiburg  
 ~~~~~

Nun werden zufällig ein Start- und ein Endpunkt in der genetischen Information bestimmt, die den Abschnitt definieren, der vom zweiten Elternteil übernommen werden soll. Hierbei könnte beispielsweise entschieden werden, die Elemente 3 und 4 zu übernehmen:

~~~~~  
 Köln - Frankfurt - München - Stuttgart - Freiburg  
 Stuttgart - München  
 ~~~~~

Das Ergebnis ist dann die folgende, neu zusammengesetzte Route (siehe Abbildung 3):

~~~~~  
 Köln - Frankfurt - Stuttgart - München - Freiburg  
 ~~~~~

Gibt es in der bestehenden Population eine Lösung, deren Fitnesswert schlechter ist als jener der neuen Route, ersetzt die neue Lösung die alte.

Wichtig zu beachten ist, dass Splice-Crossover nur dann funktioniert, wenn Wiederholungen ausgeschlossen werden können. Angenommen, anstatt den Positionen 3 und 4 wären die Positionen 2 und 4 ausgewählt worden, dann wären die folgenden genetischen Informationen rekombiniert worden:

~~~~~  
 Köln - Frankfurt - München - Stuttgart - Freiburg  
 Köln - Stuttgart - München  
 ~~~~~

Im Ergebnis wäre Köln zwei Mal aufgetreten, Frankfurt hingegen hätte gefehlt:

~~~~~  
 Köln - Köln - Stuttgart - München - Freiburg  
 ~~~~~

Das gilt es aus naheliegenden Gründen zu verhindern. Am einfachsten ist das, indem man zunächst die Schnittmenge der beiden Unterabschnitte bildet: Nur die Elemente, die in beiden auftreten, dürfen ersetzt werden. Die übrigen müssen unangetastet bleiben. So entsteht in dem Fall dann die gleiche Lösung wie bei Auswahl der Positionen 3 und 4:

~~~~~  
 Köln - Frankfurt - Stuttgart - München - Freiburg  
 ~~~~~

MUTATION ZULASSEN

Wie auch in der Natur empfiehlt es sich, gelegentlich Schwankungen in den Algorithmus einzubauen, um zu verhindern, dass sich eine Population zu sehr in eine bestimmte Richtung entwickelt: Einflüsse von Außen sind wünschenswert und fördern die Varianz der betrachteten Lösungen. Solche zufälligen Schwankungen bezeichnet man im Rahmen von genetischen Algorithmen als *Mutation*.

Die einfachste Möglichkeit, eine Mutation zu erzeugen, besteht darin, die Reihenfolge der Elemente innerhalb einer Lösung zu ändern. Es bietet sich an, die Position zweier Elemente innerhalb der Lösung zufällig zu tauschen, so dass beispielsweise aus

~~~~~  
 Köln - Frankfurt - Stuttgart - München - Freiburg  
 ~~~~~

die Lösung

~~~~~  
 Köln - München - Stuttgart - Frankfurt - Freiburg  
 ~~~~~

wird (siehe Abbildung 4). Das Verfahren wird als *Shuffle Mutation* bezeichnet. Gelegentlich kann es auch gewünscht sein, andere Arten der Mutation durchzuführen, beispielsweise Zahlen sich leicht ändern zu lassen. Im vorliegenden Beispiel macht das jedoch nicht viel Sinn: Aus *Frankfurt* beispielsweise das Wort *Frankfurt* zu machen, ist zum Finden einer optimalen Reiseroute nicht zuträglich.

Anders verhält es sich, wenn die genetischen Informationen einer Lösung aus Zahlen bestehen, die sich leicht nach oben oder unten verzerren lassen. Diese Art der Änderung nennt man *Perturb Mutation*.

Jegliche Mutation lässt sich außerdem auch mit anderen Vorgehen aus der künstlichen Intelligenz verbinden,

beispielsweise mit *Simulated Annealing*. Dabei wird das Abkühlen von glühendem Metall simuliert, was zunächst schneller, nachher langsamer erfolgt. Auf den Einfluss von Mutationen bei genetischen Algorithmen übertragen bedeutet das, dass die Schwankung und die Varianz der Mutationen im Lauf der Zeit abnehmen, um nach und nach stabilere Ergebnisse zu erhalten. Alternativ lässt sich ein solches Vorgehen auch auf die Häufigkeit von Mutationen anwenden.

Generell stellt sich auch bei Mutationen wieder die Frage, wie häufig sie auftreten sollten. Auch hierfür

gibt es wiederum keine Formel, an Hand derer sich ein geeigneter Wert berechnen ließe, sondern es gilt, einen geeigneten Wert erneut experimentell zu ermitteln. Es ist allerdings ratsam, den Anteil an Mutationen nicht zu hoch zu wählen, da das Verfahren ansonsten sehr stark vom Zufall geprägt wird und das Charakter der Optimierung in den Hintergrund tritt.

ELITEN SIND ERWÜNSCHT

Bei alledem gilt es, sicherzustellen, dass eine Population nicht insgesamt schlechter wird. Das kann

beispielsweise dann passieren, wenn die Nachkommen zwar besser sind als die bisher schlechtesten Mitglieder der Population, aber auch nicht besser als die bisher besten. Wenn dann die Lösungen, die als Eltern fungiert haben, nach einer Weile ebenfalls aus der Population entfernt werden, um beispielsweise ebenfalls der Variantenbildung Vorschub zu leisten, könnte es passieren, dass die Gesamtgüte einer Population sinkt.

Um dem vorzubeugen, ist der sogenannte *Elitismus* erwünscht. Der Begriff bedeutet, dass die wertvollsten Mitglieder einer Gesellschaft besonderen Schutz genießen und nicht ohne einen adäquaten Ersatz aus der Population entfernt werden dürfen. So wird der Erhalt der bisher besten gefundenen Lösung sichergestellt.

FAZIT

Der Ansatz, die Natur als Vorbild für die Gestaltung von Algorithmen zu verwenden, ist nicht neu. So ist auch die Idee für genetische Algorithmen nicht neu, sondern existiert bereits seit einigen Jahrzehnten. Durch den aktuellen Aufschwung der künstlichen Intelligenz, insbesondere im Bereich der neuronalen Netzwerke, erfahren auch Methoden wie genetische Algorithmen wieder mehr Aufmerksamkeit.

Besonders interessant ist das Weiterdenken der Möglichkeiten, die genetische Algorithmen bieten. Außer zur Optimierung von Problemen wie TSP können genetische Algorithmen beispielsweise auch dazu verwendet werden, autonom Computerprogramme oder Schaltpläne zu entwerfen: Vorgegeben ist hierbei das Ziel und eine Fitnessfunktion, mit der sich gefundene Lösungen bewerten lassen.

Zusammengefasst lässt sich festhalten, dass genetische Algorithmen sicherlich kein Werkzeug sind, das man tagtäglich einsetzen wird. Sie erweitern aber dennoch den Horizont und eignen sich für einige Aufgaben hervorragend, die man anderweitig gar nicht oder nur sehr ineffizient lösen könnte. Wie so oft ist es deshalb hilfreich, schon einmal über den Tellerrand geschaut und Wissen über gangbare Alternativen im eigenen Repertoire griffbereit zu haben.

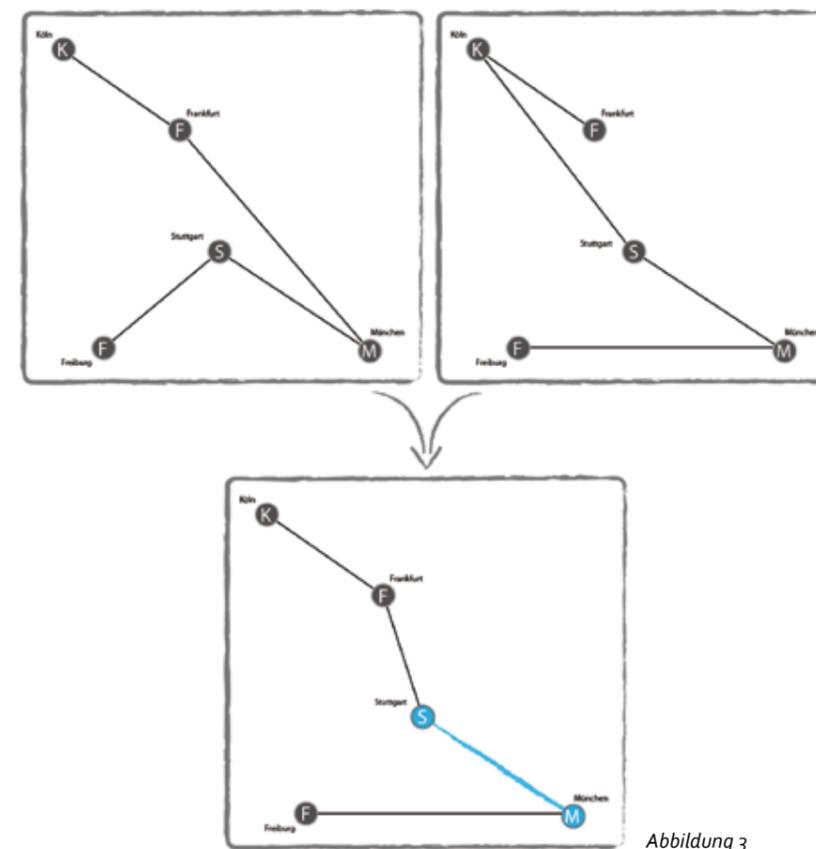


Abbildung 3

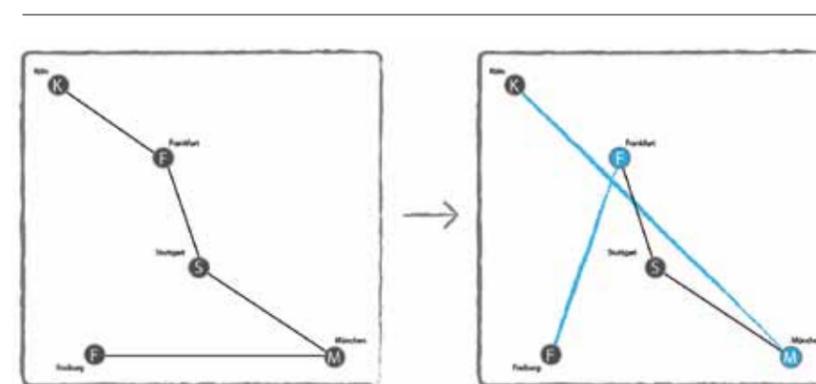


Abbildung 4



Foto: Rainer Sturm / pixelio.de

DIE KNIFFLIGEN FÄLLE BEIM TESTEN

Autor: STEFAN LIESER

Das automatisierte Testen von Softwaresystemen ist Stand der Technik. Jeder tut es. Oder sollte es zumindest. Vor einigen Jahren diskutierte man noch, ob automatisiertes Testen überhaupt sinnvoll sei. Schließlich müsste so die doppelte Menge Code geschrieben werden. Diese kurzfristige Argumentation ist glücklicherweise nicht mehr zu hören.

Die Optimierung, das Einfachste zu tun (KISS - Keep it Simple Stupid), sollte sich immer auf Tätigkeiten beziehen, die viel Zeit in Anspruch nehmen. Code zu schreiben ist eine Tätigkeit, Code zu lesen und zu modifizieren ist eine andere, die weitaus mehr Zeit in Anspruch nimmt. Es wäre töricht, den Entwicklungsprozess

auf die Phase des Schreibens zu optimieren, die kurz ist im Vergleich zur späteren Pflege des Codes. Auf Tests zu verzichten, optimiert auf den Zeitraum des Schreibens, statt auf den längeren Zeitraum von Lesen, Verstehen und Modifizieren.

Nun stellt sich allerdings weiterhin die Frage, wieso Teams sich beim Schreiben automatisierter Tests so schwertun. Nach meiner Beobachtung in Workshops der Clean Code Developer (CCD) School und meiner Beratungstätigkeit liegt das oft an der Tatsache, dass Entwickler die Hürde einiger kniffliger Fälle nicht nehmen. Im Folgenden möchte ich Ihnen daher einige dieser Herausforderungen vorstellen und Lösungen aufzeigen.

BLACKBOX ODER WHITEBOX?

Beim Testen unterscheiden wir Blackbox und Whitebox Tests. Bei Blackbox Tests wird die zu testende Funktionseinheit nur von außen betrachtet. Wir betrachten sie als „black box“, in die wir keinen Einblick haben. Wie es innen drin aussieht, wie diese Box funktioniert, das sehen wir nicht. Bei Whitebox Tests dagegen wissen wir, wie die Box von innen aussieht und wie sie funktioniert. Wir kennen alle Details, den Algorithmus, die Interna. Dieses interne Wissen nutzen wir in Whitebox Tests gezielt aus, während wir bei Blackbox Tests ausschließlich über die

Wenige Integrationstests



Abbildung 1: Testpyramide

öffentliche Schnittstelle testen.

Immer wieder wird diskutiert, ob man nicht besser auf Whitebox Tests verzichten sollte, weil diese bei internen Änderungen an der Implementation häufig zerbrechen. Zu empfehlen ist hier eine pragmatische Vorgehensweise. Das vollständige Ablehnen von Whitebox Tests erscheint dogmatisch und verkennt die damit verbundenen Chancen. Auf der anderen Seite sind Whitebox Tests mit einer gewissen Umsicht zu erstellen, denn tatsächlich droht das Risiko, dass diese Tests bei Änderungen zerbrechen. Am Ende geht es bei der Frage Whitebox oder Blackbox darum abzuwägen, zwischen Stabilität der Tests einerseits und leichter Testbarkeit andererseits. Whitebox Tests haben mit Sicherheit einen hohen Wert. Sie ermöglichen es, eine gesunde Struktur von Tests aufzubauen: Wenige Integrationstests über die öffentliche API, viele Unittests auf Interna. Die folgende Abbildung zeigt, wie die Anzahl von Tests verteilt sein sollte. Viele Unittest bilden die Basis. Darauf bauen einige wenige Integrationstests auf. Die Integrationstests kümmern sich um die vielen Details und Fallunterscheidungen, die Integrationstests prüfen, ob die einzelnen Einheiten im Zusammenspiel korrekt funktionieren.

ASPEKTE TRENNEN: INTEGRATION VS. OPERATION

Die Unterscheidung von Blackbox und Whitebox Tests wird deutlich, wenn wir über die Trennung von Integration und Operation sprechen. Das IOSP, das Integration Operation Segregation Principle, besagt, dass eine Methode entweder Integration oder Operation sein soll [1]. Der

folgende Ausschnitt zeigt das an einem einfachen Beispiel:

```
public class Configuration
{
    public static
    IDictionary<string, string>
    ToDictionary(string configuration) {
        var settings = SplitIntoSet
        tings(configuration);
        var keyValuePairs = SplitInt
        oKeyValuePairs(settings);
        var dictionary = CreateDict
        ionary(keyValuePairs);

        return dictionary;
    }

    internal sta-
    tic IEnumerable<string>
    SplitIntoSettings(string configura-
    tion) {
        return configuration.
        Split(,;');
    }

    internal static IEnumerable<Ke
    yValuePair<string, string>> SplitIn
    toKeyValuePairs(IEnumerable<string>
    settings) {
        foreach (var setting in
        settings) {
            var keyAndValue = set-
            ting.Split(,=');
            yield re-
            turn new KeyValuePair<string,
            string>(keyAndValue[0],
            keyAndValue[1]);
        }
    }

    internal static
    IDictionary<string, string> CreateDi
    ctionary(IEnumerable<KeyValuePair<st
    ring, string>> keyValuePairs) {
        var result = new
        Dictionary<string, string>();
        foreach (var keyValuePair
        in keyValuePairs) {
            result.Add(keyValuePair.
            Key, keyValuePair.Value);
        }
        return result;
    }
}
```

Die Methode ToDictionary gehört zur Kategorie Integration. Diese Methode ist für den Aufruf weiterer Methoden zuständig. Sie integriert diese Methoden. Die Methoden SplitIntoSettings, SplitIntoKeyValuePairs und CreateDictionary sind dagegen Operationen. Diese Methoden enthalten

die Domänenlogik (Logik, die das Thema der Anwendung betrifft) und rufen keine weiteren Methoden auf, von Methoden aus Frameworks einmal abgesehen. Code wird leichter lesbar und verständlicher, wenn wir bei Methoden klar zwischen Integration und Operation trennen.

TESTSTRATEGIE

Bei der klaren Trennung von Integration und Operation ist es sinnvoll, die Operationen isoliert zu testen. Diese Methoden sind Blätter im Abhängigkeitsbaum, sind also ohnehin schon isoliert. Die Integrationsmethoden dagegen haben Abhängigkeiten zu den Methoden, die sie integrieren. Sie isoliert zu testen würde bedeuten, die realen Methoden im Test durch Attrappen zu ersetzen. Technisch ist das zwar möglich, doch steht der Nutzen in keinem Verhältnis zum Aufwand. Die Integrationsmethoden testet man also nur mit Integrationstests. Folglich besteht die Teststrategie aus wenigen Integrationstests über die öffentliche API und vielen Unittests für die Operationen.

SICHTBARKEIT

Nun stellt sich die Frage, wie die Operationen in den automatisierten Unittests aufgerufen werden können. Normalerweise würden die Interna der Klasse auf private gesetzt. Damit wären die Operationen für automatisierte Tests nicht ohne weiteres erreichbar. Man könnte sie per Reflection aufrufen. Doch das macht viel Mühe. Mal ganz davon abgesehen, dass man dann beim Refactoring aufpassen müsste, weil die Methodennamen dann als Zeichenketten in den Tests auftauchen würden. Also setzt man die Operationen auf internal und ergänze das InternalsVisibleTo Attribut auf der Implementationsassembly. Beachten Sie, dass Sie den Assemblynamen der Assembly angeben müssen, der Sie den Zugriff auf das internal Symbol gestatten möchten. Nach Anlegen eines neuen



Abb. 2: Die Implementationsassembly gestattet der Testassembly den Zugriff auf die Interna

Projekts entspricht der Assemblyname dem Projektnamen. Sollten Sie den Projektnamen allerdings ändern, bleibt der ursprüngliche Assemblyname zunächst erhalten. Im `InternalsVisibleTo` Attribut muss zwingend der Assemblyname stehen, nicht der Projektname.

Auf diese Weise sind die Interna der Klasse für die Testassembly sichtbar und können daher automatisiert getestet werden. Es handelt sich hier um Whitebox Tests, weil die Tests nun Kenntnis haben über die interne Struktur und die Funktionsweise der Implementation.

Allerdings sind die Interna nun auch in der Implementationsassembly sichtbar. Die mit `internal` markierten Methoden der Klasse können aus anderen Klassen innerhalb derselben Assembly aufgerufen werden. Damit ist die Sichtbarkeit der Interna nicht nur auf die Tests ausgedehnt, sondern leider auch auf die Implementationsassembly. Diesen Nachteil nehmen wir zugunsten der guten Testbarkeit in Kauf. Innerhalb des Teams muss die Vorgehensweise allen Entwicklern bekannt sein, um zu vermeiden, dass Abhängigkeiten zu `internal` Methoden eingegangen werden. Ganz pragmatisch gesehen, ist dies ein sehr kleiner Nachteil. Auch ohne diese Teststrategie sollten Entwickler keine Abhängigkeit zu `internal` Methoden eingehen, ohne gut darüber nachzudenken, was dies für Folgen haben könnte. Man könnte sagen, dass `internal` auf derselben Ebene wie `private` steht: internes, privates Zeug, von dem man die Finger lässt. Regelmäßige Code Reviews können im Team dafür sorgen, dass Probleme mit Abhängigkeiten rechtzeitig erkannt werden.

EXCEPTIONS AUTOMATISIERT TESTEN MIT NUNIT

Im Kontext von automatisierten Tests fallen Exceptions in eine der beiden folgenden Kategorien:

- Eine getestete Methode löst selbst eine Exception aus, wenn sie einen Ausnahmezustand entdeckt.
- Während der Ausführung einer im Test befindlichen Methode kann eine Exception auftreten, auf die die Methode reagiert.

Alle anderen Fälle von Exceptions sind für automatisierte Tests nicht relevant. Insbesondere der häufigste Fall, dass eine Ausnahme zwar auftreten kann, aber nicht weiter behandelt wird, spielt beim Testen keine Rolle. Solche Fälle landen ohnehin beim globalen Exception Handler der Anwendung. Wenn der Code, der getestet werden soll, eine Ausnahme nicht behandelt, gibt es dazu auch nichts zu testen.

EINE EXCEPTION WIRD ERWARTET

Liegt der Fall vor, dass im regulären Programmfluss eine Exception ausgelöst werden kann, sollte es für diesen Fall einen automatisierten Test geben. Dieser dient einerseits dazu sicherzustellen, dass der Code auch zukünftig wie erwartet eine Ausnahme auslöst. Zusätzlich dient ein solcher Test der Dokumentation. Der Test drückt dann aus, unter welchen Umständen die Exception erwartet wird, sodass Entwickler, die den Test bzw. die zugehörige Implementation später lesen, weniger überrascht sein werden.

Das folgende Beispiel zeigt, wie mit NUnit [2] überprüft werden kann, ob eine Ausnahme ausgelöst wird.

```
[TestFixture]
public class ExceptionTests
{
    [Test]
    public void Exakte_Erwartung_über_eine_Ausnahme()
    {
        Assert.Throws<NotImplementedException>(
            () => Sut.DoSomethingThatThrows());
    }

    [Test]
    public void Irgendeine_Ausnahme() {
        Assert.Catch<Exception>(
            () => Sut.DoSomethingThatThrows());
    }
}

public static class Sut
{
    public static void DoSomethingThatThrows() {
        throw new NotImplementedException(„Bumsdi!“);
        //throw new NullReferenceException();
    }
}
```

Der erste Test erwartet, dass eine `NotImplementedException` ausgelöst wird. Wird irgendeine andere Exception ausgelöst, selbst wenn es eine Ableitung des erwarteten Exceptiontyps sein sollte, schlägt der Test fehl. Der zweite Test erwartet dagegen einen Exceptiontyp, der von `Exception` abgeleitet ist. Im Beispiel sind beide Tests grün.

Bei MSTest ist die Vorgehensweise eine andere. Hier wird durch Annotieren des Tests mit dem `ExpectedException` Attribut der Test nur dann als erfolgreich gewertet, wenn die benannte Ausnahme innerhalb der Testmethode ausgelöst wird.

```
namespace exceptions_mstest
{
    [TestClass]
    public class ExceptionTests
    {
        [TestMethod, ExpectedException(typeof(NotImplementedException), AllowDerivedTypes = false)]
        public void Exakte_Erwartung_über_eine_Ausnahme() {
            Sut.DoSomethingThatThrows();
        }

        [TestMethod, ExpectedException(typeof(Exception), AllowDerivedTypes = true)]
        public void Irgendeine_Ausnahme() {
            Sut.DoSomethingThatThrows();
        }
    }
}
```

Der (kleine) Nachteil: Die Exception kann irgendwo innerhalb des Tests auftreten. Die weiter oben gezeigte NUnit Syntax ist da deutlich spezifischer. Sie sollten also in jedem Fall beachten, dass die Testmethode knapp und fokussiert geschrieben wird, damit der Test nicht versehentlich grün ist, obschon die Ausnahme an einer anderen als der erwarteten Stelle ausgelöst wird.

DETAILS EINER EXCEPTION TESTEN

Manchmal ist es wichtig zu überprüfen, ob eine Exception mit den erwarteten Details ausgelöst wird. Vor allem die Eigenschaft `Message` kommt hier in Frage. Der folgende Test prüft, ob die Message einer Exception dem String „Bumsdi!“ entspricht.

```
[Test]
public void Eigenschaften_der_Ausnahme() {
    var exception = Assert.Catch<Exception>(
        () => Sut.DoSomethingThatThrows());
    Assert.That(exception.Message,
        Is.EqualTo(„Bumsdi!“));
}
```

Alternativ kann mit `Does.StartWith` überprüft werden, ob die Message mit dem String „Bum“ beginnt.

```
Assert.That(exception.Message, Does.StartWith(„Bum“));
```

Mit MSTest sieht der Test wie folgt aus:

```
[TestMethod]
public void Eigenschaften_der_Ausnahme() {
    try {
        Sut.DoSomethingThatThrows();
    }
    catch (Exception exception) {
        Assert.AreEqual(„Bumsdi!“, exception.Message);
        StringAssert.StartsWith(exception.Message,
            „Bum“);
    }
}
```

EINE EXCEPTION AUSLÖSEN

Wenn in der Implementation auf Ausnahmen reagiert wird, muss dieser Code natürlich ebenfalls automatisiert getestet werden. Die Methode verhält sich anders, wenn eine Ausnahme eintritt. Das bedeutet jedoch, dass man im Test die Ausnahme auslösen muss, auf die der zu testende Code reagieren soll. Mit den entsprechenden Werkzeugen ist das kein Problem, wie das folgende Beispiel zeigt. Sehen Sie hier zunächst eine Klasse, die in ihrer `FormatContent` Methode entweder den von `ReadContent` gelesenen string zurückgibt oder im Fall einer Ausnahme einen anderen string liefert.

```
public class SUT
{
    public string FormatContent() {
        string content;
        try {
            content = ReadContent();
        }
        catch (Exception) {
            return „Eine Ausnahme...“;
        }
        return content;
    }

    public string ReadContent() {
        return „Some content...“;
    }
}
```

Das sogenannte Happy Day Szenario lässt sich leicht testen. Hierbei geht es um einen Test für die Situation, dass alles glatt läuft.

```
[Test]
public void Happy_day() {
    Assert.That(new SUT().FormatContent(),
        Is.EqualTo(„Some content...“));
}
```

Interessanter wird der Fall, wenn nun überprüft werden soll, was das Resultat ist, wenn die `ReadContent` Methode eine Ausnahme auslöst. Im folgenden Test wird das Mock Framework `TypeMock Isolator` [3] verwendet.

```
[TestFixture, Isolated]
public class ExceptionsAuslösenTests
```

```
{
    [Test]
    public void Eine_Exception_auslösen() {
        var sut = new SUT();
        Isolate.WhenCalled(() => sut.ReadContent()).
            WillThrow(new Exception());

        Assert.That(sut.FormatContent(), Is.EqualTo(„Eine
        Ausnahme...“));
    }
}
```

Zunächst erzeugt man eine Instanz des System Under Test (SUT). Anschließend weist man den `TypeMock Isolator` mit der Methode `Isolate.WhenCalled` an, beim Aufruf der `ReadContent` Methode eine Ausnahme auszulösen. Zuletzt wird dann geprüft, ob der Aufruf der `FormatContent` Methode nun den erwarteten String liefert.

FLUCH ODER SEGEN?

Über den Einsatz solch leistungsfähiger Mock Frameworks wird unter Entwicklern immer wieder diskutiert. Da `TypeMock Isolator` und andere Produkte dieser Gattung unter Zuhilfenahme der Profiler API realisiert sind, kann man mit ihnen quasi alles durch Attrappen ersetzen. Damit wird es möglich, auch solchen Code automatisiert zu testen, der mit Abhängigkeiten eben nicht sauber umgeht. Das soll natürlich nicht dazu führen, dass Clean Code Developer [4] Prinzipien und Praktiken über Bord gehen können, wenn Ihre Tools nur ausreichend leistungsfähig sind. Sicher schneiden Sie Ihre Kartoffeln mit einem Messer klein, mit dem Sie ja auch Schaden anrichten könnten. Auf das leistungsfähige Werkzeug zu verzichten, weil man es auch falsch bedienen kann, ist keine besonders schlaue Vorgehensweise.

GRAFISCHE BENUTZERSCHNITTSTELLE

Das automatisierte Testen der grafischen UI einer Desktop Anwendung ist auf den ersten Blick eine größere Herausforderung. Doch mit ein paar simplen Tricks kommt man bereits sehr weit. Das folgende Beispiel stellt Ihnen einige einfache Möglichkeiten vor.

Die Abbildung zeigt eine simple UI, die eine Liste von Adressen anzeigt. Im unteren Bereich befindet sich eine `CheckBox`, die den danebenstehenden Button beeinflusst. Das Beispiel ist konstruiert, um daran einige Techniken zu erklären. Es wurde mit WPF erstellt. Die Techniken lassen sich problemlos auf WinForms übertragen.

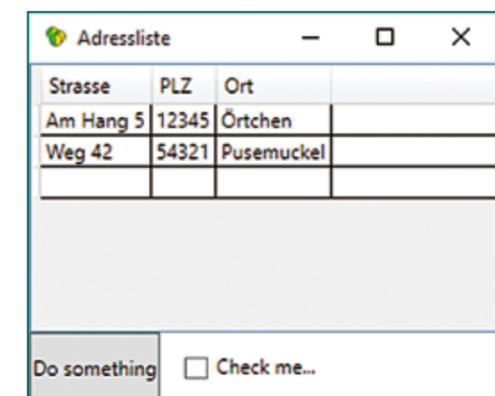


Abbildung 3: Beispiel UI

MVVM - MODEL VIEW VIEWMODEL

Es kommt ein View nebst ViewModel und Data Binding zum Einsatz. Da die Werte in der Tabelle nicht verändert werden können, verzichten wir darauf, im ViewModel Adresse das Interface INotifyPropertyChanged zu implementieren.

In den folgenden Listings sehen Sie den Xaml Code, das ViewModel Adresse sowie die Code Behind Datei.

```
<Window x:Class="ui.AdresslisteForm"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Adressliste" Height="300" Width="300">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="5*" />
      <RowDefinition Height="1*" />
    </Grid.RowDefinitions>

    <DataGrid x:Name="grid" AutoGenerateColumns="True" Grid.Row="0" />
    <StackPanel Grid.Row="1" Orientation="Horizontal" >
      <Button x:Name="button" Content="Do something" />
      <CheckBox x:Name="checkMe" Content="Check me..." Margin="14" />
    </StackPanel>
  </Grid>
</Window>

namespace ui
{
  public class Adresse
  {
    public string Strasse { get; set; }
    public string PLZ { get; set; }
    public string Ort { get; set; }
  }
}

using System;
using System.Collections.Generic;
using System.Windows;

namespace ui
{
  public partial class AdresslisteForm : Window
  {
    public AdresslisteForm() {
      InitializeComponent();
      button.Click += ButtonClickHandler;
      checkMe.Click += CheckBoxClickHandler;
    }

    public void SetAdressliste(IEnumerable<Adresse> adressliste) {
      grid.ItemsSource = adressliste;
    }

    public event Action ButtonClickEvent;

    internal void ButtonClickHandler(object sender, RoutedEventArgs e)
    {
      ButtonClickEvent();
    }

    internal void CheckBoxClickHandler(object sender, RoutedEventArgs
e) {
      if (checkMe.IsChecked.Value) {
        button.IsEnabled = false;
      }
      else {
        button.IsEnabled = true;
      }
    }
  }
}
```

DEN DIALOG MIT DATEN ANZEIGEN

Ein erster halb-automatisierter Test sorgt zunächst einmal dafür, dass der Dialog mit Daten gefüllt und dann angezeigt wird. Im realen Projekt ersparen Sie sich mit Hilfe solcher Tests, dass Sie die Anwendung starten und an die entsprechende Stelle navigieren müssen. In komplexen Anwendungen bedeutet das eine große Zeitersparnis. Es müssen keine Beispieldaten in den Datenbanken hergestellt werden, etc. Programmatisch lassen sich solche Testfälle viel besser lösen. Insbesondere kann ein Dialog so bereits getestet werden, bevor er in die Anwendung integriert wird.

```
using System.ComponentModel;
using System.Threading;
using NUnit.Framework;

namespace ui.tests
{
  [TestFixture]
  public class AdresslisteTests
  {
    private AdresslisteForm sut;
    private BindingList<Adresse> adressliste;

    [SetUp]
    public void Setup() {
      sut = new
        AdresslisteForm();
      adressliste = new
        BindingList<Adresse> {
          new Adresse {PLZ =
            „12345“, Ort = „Örtchen“, Strasse =
            „Am Hang 5“},
          new Adresse {PLZ =
            „54321“, Ort = „Pusemuckel“, Strasse =
            „Weg 42“}
        };
    }

    [Test,
    Apartment(ApartmentState.STA),
    Explicit]
    public void Mehrere_Adres-
    sen_anzeigen() {
      sut.
        SetAdressliste(adressliste);
      sut.ShowDialog();
    }
  }
}
```

Der Test trägt neben dem obligatorischen Test Attributen noch das Attribut Explicit, damit er nicht beim Ausführen aller Tests anspringt. Da der Test den Dialog mit ShowDialog öffnet, bleibt er solange am Bildschirm stehen, bis Sie ihn wieder schließen. Dieser Test ist eben nur halb-automatisiert: Es werden automatisiert Testdaten vorbereitet und

an den Dialog übergeben. Dann wird der Dialog geöffnet und bleibt zur Inaugenscheinahme am Bildschirm stehen. Damit das technisch funktioniert, muss das Attribut Apartment(ApartmentState.STA) ergänzt werden, da WPF andernfalls meckert.

TESTEN, OB EIN BUTTON EINEN EVENT AUSLÖST

Der folgende Test überprüft, ob beim Betätigen des Buttons der erwartete Event ausgelöst wird. Auch dieser Test ist halbautomatisiert. Im Test wird in der Arrange Phase eine Lambda Expression an den Event gebunden, die eine MessageBox anzeigt. Auf diese Weise kann man den Test starten, auf den Button drücken und dann beobachten, ob die MessageBox angezeigt wird. Interessanter werden solche Tests, wenn beim Event Parameter mitgeliefert werden. Die können dann in der Nachricht der MessageBox angezeigt werden, um so zu prüfen, ob die erwarteten Werte geliefert werden.

```
[Test, Apartment(ApartmentState.STA), Explicit]
public void Button_Handler_löst_Ereignis_aus_Message-
Box() {
  sut.ButtonClickEvent += () => MessageBox.
  Show(„Hello from the ButtonClickEvent“);
  sut.ShowDialog();
}
```

Solche halbautomatisierten Tests sind eine Übergangslösung in Legacy Code Projekten. Statt mit F5 die Anwendung zu starten und komplett manuell zu testen, lassen sich auf diese Weise einige Abläufe automatisieren. Wirklich rund wird die Sache, wenn der Test vollständig automatisiert abläuft, also auch ein Assert enthält, um eine Annahme zu überprüfen.

AUTOMATISIERTER TEST

```
[Test, Apartment(ApartmentState.STA)]
public void Button_Handler_löst_Ereignis_aus() {
  var count = 0;
  sut.ButtonClickEvent += () => count++;

  sut.ButtonClickHandler(this, new RoutedEventArgs());

  Assert.That(count, Is.EqualTo(1));
}
```

Dieser Test läuft nun komplett automatisiert. An den Event des Buttons wird eine Lambda Expression gebunden, die einen Zähler inkrementiert. Nun soll der Button gedrückt werden, um dann zu prüfen, ob der Zähler auf 1 steht. Würde man jetzt den Dialog mit ShowDialog öffnen, würde die Testausführung unterbrochen, bis der Dialog geschlossen wird. Folglich müsste der Dialog auf einem eigenen Thread geöffnet werden. Ferner bliebe dann noch die Frage offen, wie der Button programmatisch gedrückt werden kann. Am besten verwendet man hier eine viel einfachere Lösung: Rufen Sie im Test die Methode auf, die in der UI am Button Click Event hängt. Das folgende Listing zeigt noch einmal den relevanten Ausschnitt aus der Code Behind Datei:

```
button.Click += ButtonClickHandler;
...
internal void ButtonClickHandler(object sender, Rout-
edEventArgs e) {
  ButtonClickEvent();
}
```

Um die Methode ButtonClickHandler im Test erreichen zu können, setzt man sie auf internal. Details zum Thema Sichtbarkeit haben Sie weiter oben bereits kennengelernt. Nach dem gleichen Schema kann man nun automatisiert prüfen, ob beim Anhängen der CheckBox der Zustand des Buttons modifiziert wird.

```
[Test, Apartment(ApartmentState.STA)]
public void CheckBox_schaltet_Button() {
  Assert.That(sut.button.IsEnabled, Is.True);

  sut.checkMe.IsChecked = true;
  sut.CheckBoxClickHandler(this, new
RoutedEventArgs());
  Assert.That(sut.button.IsEnabled, Is.False);

  sut.checkMe.IsChecked = false;
  sut.CheckBoxClickHandler(this, new
RoutedEventArgs());
  Assert.That(sut.button.IsEnabled, Is.True);
}
```

Hier prüft man zunächst, ob der Button initial aktiviert ist. Anschließend simuliert man im Test das Anklicken der CheckBox, in dem wir den Zustand IsChecked setzen und dann den Handler CheckBoxClickHandler aufrufen, der am Click Event der CheckBox hängt. So können wir dann anschließend prüfen, ob der Button wie erwartet modifiziert wurde. Selbstredend gehört solche Logik nicht in den View. Besser wäre es, ein ViewModel zu verwenden, welches diese Logik enthält. Das ViewModel ist dann einfach automatisiert testbar und beeinflusst den View mittels Data Binding. Leider werden Tests in Projekten nach wie vor oft weggelassen. Sie dann später zu ergänzen bedeutet, dass man mit der Codesituation leben muss. In diesen Fällen sind automatisierte Tests auf die gezeigte Weise möglich.

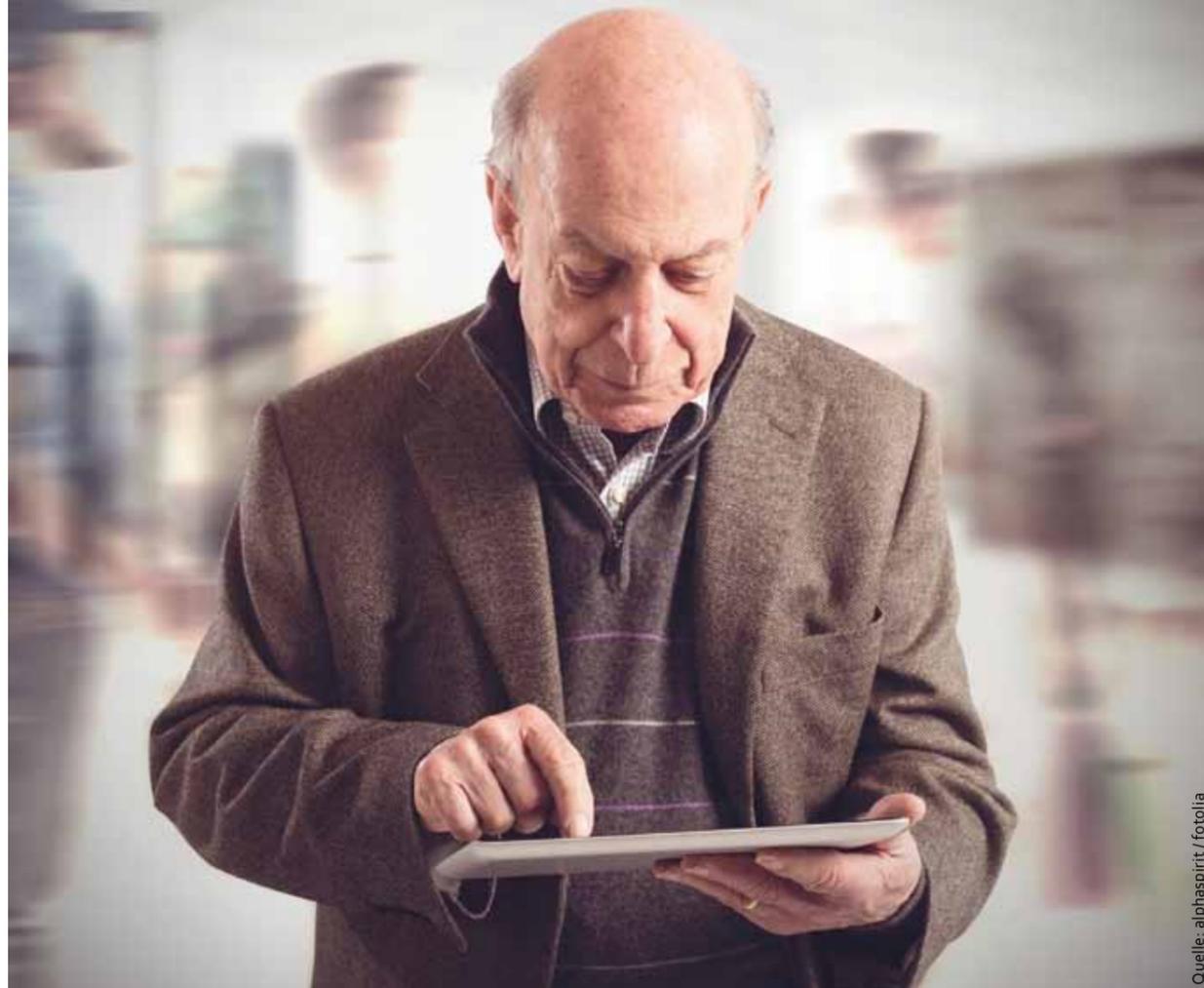
FAZIT

Das automatisierte Testen von Softwaresystemen ist einfach, wenn wichtige Prinzipien eingehalten werden. Vor allem das Thema Abhängigkeiten ist in den Blick zu nehmen. Durch Einhalten des IOSP wird die Situation deutlich verbessert. Unter [5] finden Sie Cheatsheets zu NUnit und MSTest, die Ihnen den Einstieg in die Syntax der Tests erleichtern.

QUELLEN

- [1] http://clean-code-developer.de/die-grade/roter-grad/#Integration_Operation_Segregation_Principle_IOSP
- [2] <http://nunit.org>
- [3] <http://typemock.com>
- [4] <http://clean-code-developer.de>
- [5] <http://refactoring-legacy-code.net/cheatsheet-unit-tests>

STARKE ZIELGRUPPE



Eine Herausforderung für Entwickler: Die IT für die Generation 65+.

Autoren: Dr. VEIKKO KRYPCZYK und OLENA BOCHKOR

Die Wirtschaft, insbesondere die Tourismusindustrie haben sie längst entdeckt und ihre Produkte passgenau ausgerichtet. Gemeint sind die älteren Mitbürger in unserer Gesellschaft. Sie verfügen über eine enorme Kaufkraft und wollen ihre Wünsche auch am Markt umsetzen. Im Gegensatz zur heutigen Generation der 40jährigen hat die heute 65+-Generation das Internet, Smartphone und E-Commerce nur zögerlich in ihren Alltag integriert. Um auch diese Kunden bestmöglich mit Apps und Webseiten zu bedienen, müssen einige wichtige Punkte beim Design und bei der Umsetzung beachtet werden.

Ein Grundsatz des modernen Marketings lautet, die Produktkonzeption bestmöglich an den relevanten Zielgruppen auszurichten. Dieses gilt sowohl für das eigentliche Produkt, als auch für die Informationsmöglichkeiten zum Produkt, zum Beispiel die betreffende Webseite. Die heutige Generation der 65+-Jährigen unterscheidet sich in vielen Punkten maßgeblich von ihrer Eltern- und noch mehr von ihrer Großelterngeneration. Üblicherweise standen diese Personen über die letzten Jahre intensiv im Berufsleben und befinden sich am Übergang in den Ruhestand oder sind bereits seit wenigen Jahren in Rente. Sie verfügen über beachtliche finanzielle Möglichkeiten, sowohl aus regelmäßigen Einkommen (Rente, Pension), als auch über Ersparnisse. Der wesentliche Unterschied ergibt sich aber aus ihren Interessen. Sie haben i.d.R. weiterhin ein großes Interesse an Reisen und Konsum von Gütern und Dienstleistungen. Hinzukommt der positive Umstand, dass sie nunmehr auch die zeitlichen Möglichkeiten einer freizeitorientierten Gestaltung ihres Alltages haben. Viele Bereiche der Wirtschaft haben diese Personengruppe längst entdeckt und ihre Produkte und Dienstleistungen an ihre Bedürfnisse speziell angepasst. Ein sehr gutes Beispiel ist die Tourismusindustrie. „Silver Ager“ oder ähnlich wird diese aktive und für die Wirtschaft hoch interessante Zielgruppe liebevoll bezeichnet. Dabei muss man wissen, dass weder die Bezeichnung noch die Altersabgrenzung eindeutig festgelegt und in der Praxis homogen gebraucht werden.

Die Einflüsse und Möglichkeiten des digitalen Wandels sind natürlich auch an dieser Zielgruppe nicht vorbeigegangen. Auch unsere älteren Mitbürger wollen aktiv die Möglichkeiten des Internets nutzen. Dabei ist jedoch ein Faktor nicht zu vernachlässigen. Die vorgestellte Generation ist nicht mit dem Internet aufgewachsen. Die digitale Gesellschaft mit all ihren Möglichkeiten – und auch ihre Gefahren – u.a. in Form von Internet, Smartphone und E-Commerce haben sie meist nur zögerlich in ihrem Alltag integriert. Oft wurden die neuen Techniken eine lange Zeit ignoriert und die Beschäftigung mit den sich daraus ergebenden Anforderungen hat

nur am Rande stattgefunden. Der digitale Wandel ist jedoch mit unverminderter Geschwindigkeit weitergegangen, ein Ignorieren moderner Technologien ist kaum noch möglich und auch aus bekannten Gründen von keiner Seite mehr erwünscht.

Für die IT-Dienstleister ergeben sich in diesem Kontext neue Aufgaben, aber auch viele Herausforderungen. Digitale Produkte, wie Webseiten, Webshops und Apps sind an die Maßgaben der Zielgruppe anzupassen. Neben Fragen eines angepassten Designs, gehört auch das Handling der digitalen Produkte auf den Prüfstand. Der vorliegende Artikel gibt einen Einblick in damit verbundenen Fragestellungen und beschreibt Lösungsideen.

SENIOREN IM NETZ

Im Jahr 2015 veröffentlichte das Statistische Bundesamt ein Bericht über die Generation 65+ in Deutschland. Das Fazit lautet: Deutschland hat den zweithöchsten Anteil älterer Menschen in der Europäischen Union. Einen höheren Wert konnte nur Italien aufweisen. Geht diese Entwicklung unverändert weiter, so kommt die Vorhersage zu dem folgenden Ergebnis: Im Jahr 2060 wird bereits jeder Dritte (33 %) in Deutschland mindestens 65 Jahre alt sein [1]. Das Internet ist heutzutage aus dem Alltag und Berufsleben nicht mehr wegzudenken. Die jüngeren Menschen sind damit aufgewachsen bzw. im Berufsleben ist eine intensive Nutzung obligatorisch. Bei der älteren Generation sieht es ein wenig anders aus: Für Senioren ist der Umgang mit der Technik noch immer nicht vertraut. Dennoch, die Zahl der Senioren, die über einen Online-Zugang verfügen, steigt kontinuierlich. Es gibt zahlreiche Kurse, welche die Senioren in das „unbekannte digitale Universum“ begleiten. Gadgets stehen auf Platz eins der Weihnachtswunschliste bei der Generation 65+. Die Untersuchungen zeigen, dass interessanterweise ältere Männer öfters in der digitalen Welt unterwegs sind, als Frauen. Ihr Interesse an Technik scheint höher zu sein. Das Internet wird vor allem für die Informationssuche und als Kommunikationsmittel genutzt. Bei der Informationssuche sind oft Webseiten zum Thema Gesundheit



Zu kleine Schrift und zu viele Symbole

gefragt. Ebenso beliebt sind Portale für Online-Shopping und Reisen. Häufig konsumierte Produkte sind Medikamente, Elektronikartikel und Urlaubsreisen. Ebenso werden häufig Eintrittskarten für die unterschiedlichsten Veranstaltungen von der Generation 65+ im Internet bestellt. Das Thema Online-Banking wird deutlich vorsichtiger angegangen. Neben einer großen Skepsis aus Gründen der Sicherheit, ist es die Bedienung, welche Barrieren für die Nutzung darstellen. Die Vertrautheit mit der Dateneingabe steigt massiv an, wenn das Online-Formular dem alten, über Jahre vertrauten Papierformular gleicht. In Zahlen ausgedrückt: Internet Banking wird von ca. 50% der männlichen und ca. 34% der weiblichen Senioren genutzt.

Aus den Darstellungen kann man schlussfolgern: Die Generation 65+ ist die am schnellsten wachsende Nutzergruppe im Internet. Damit wird ein neues Marktsegment geschaffen. Erfolgreich wird man aber nur sein, wenn man sie zielgruppengerecht anspricht.

USABILITY FÜR SENIOREN

Leider ist der Großteil der Internet-Auftritte nicht entsprechend auf die Zielgruppe der älteren Personen ausgerichtet. Ältere Menschen sind nur dann von einem Mehrwert durch die

Nutzung des Internets überzeugt, wenn die Webseiten im Umgang komfortabel und nicht kompliziert sind. Der Inhalt muss interessant und die Seite muss zufriedenstellend, effizient und effektiv benutzbar sein. Ein Problem bei der Konzeption einer Webseite ist es, das Gleichgewicht zwischen der visuellen Wahrnehmung und einer guten Usability zu finden.

Welche Aspekte man als Entwickler und Designer bei der Umsetzung seniorengerechter Internet-Auftritte und anderer digitaler Produkte wie zum Beispiel Apps beachten muss, stellen wir nachfolgend vor. Dabei ist zu berücksichtigen, dass sich die kognitive Leistungsfähigkeit, die feinmotorische Koordination und die Sehfähigkeit mit dem zunehmenden Alter verändern. Dies alles muss bei der Gestaltung von Anwendungen, Applikationen und Webseiten beachtet werden. Die Information auf der Webseite soll transparent angeordnet sein. Die Seite soll vom Inhalt und nicht von aufwendigen Designelementen dominiert werden. Der Grund: Ältere Benutzer lassen sich von irrelevanten Elementen schnell ablenken. Ebenso brauchen sie mehr Zeit um „wichtig“ von „unwichtig“ zu unterscheiden. Weiter muss die Navigationsstruktur eine gute Struktur haben. Die Suchmöglichkeiten sollen einfach und schnell erreichbar sein. Oft fällt den älteren Menschen eine exakte Positionierung des Mauszeigers schwer. Deswegen sollte man bei der seniorengerechten Gestaltung von Webseiten keine Steuerungselemente verwenden, welche den exakten Einsatz einer Maus erfordern. Ein weiterer zu beachtender Punkt ist: Ältere Benutzer scrollen i.d.R. nicht gerne! Dieses soll bei der Festlegung der Seitenlänge in Betracht gezogen werden. Hier gibt es einen Widerspruch zum aktuellen Trend des Einsatzes des Single-Page-Designs, d.h. alle Inhalte auf einer Seite – von oben nach unten – zu präsentieren. Dieses beliebte Mittel sollte bei einer älteren Zielgruppe nur bedingt oder in abgewandelter Form angewendet werden. Zusätzliche Menüpunkte erlauben es, innerhalb der Seite zu navigieren. Vertikales Scrollen sollte ebenso nur begrenzt eingesetzt werden. Horizontales und vertikales Scrollen

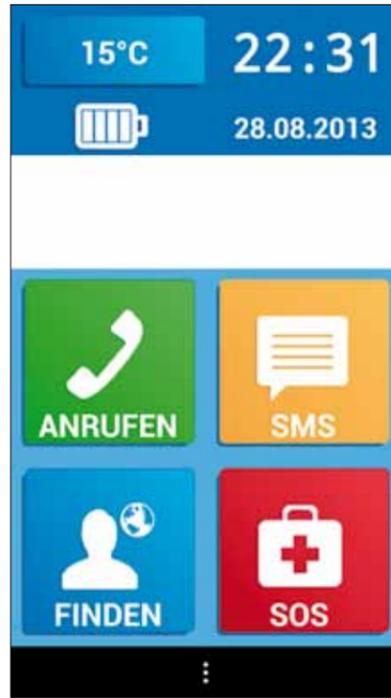


Abbildung 1: Die Benutzeroberfläche der App Simple Phone Senior rückt die wichtigsten Funktionen in den Fokus und macht damit die Smartphone-Benutzung seniorengerecht.

sollte man zusammen vermeiden. Die Marktforschung empfiehlt deshalb für wichtige Navigationselemente eine Positionierung unabhängig vom Scroll-Status, ähnlich einem „Floating Banner“.

Die Schrift sollte ausreichend groß dargestellt werden bzw. es soll eine Möglichkeit geben, die Schriftgröße einfach anzupassen. Dass diese Funktion jedes Betriebssystem bzw. jeder Internetbrowser bietet, spielt keine Rolle. Die Kenntnisse zur grundsätzlichen Anpassung können nicht vorausgesetzt werden.

Farben spielen eine wichtige Rolle. Wenige Farben werden von Senioren als positiv empfunden. Es soll möglichst kontraststarken Elementen gearbeitet werden. Der Einsatz von Animationen und kontrastschwachen Hintergrundbildern sollte unterbleiben. Fachwörter, Modewörter, Fremdwörter und Anglizismen sollen vermieden werden. Ein Beispiel: Der Begriff „Sitemap“ – durchaus anderen Nutzern bekannt – kam den älteren Mitbürgern oft sehr fremd vor [2], [3].

An welcher Auflösung sollte das Design ausgerichtet werden? Meist arbeiten Senioren nicht mit den neuesten Geräten, zum Beispiel haben sie

ein älteres Gerät von ihren Kindern/Enkelkindern übernommen. Ebenso stellen sie die Auflösung für eine bessere Lesbarkeit auf geringe Werte ein. 800x600 Pixel sind hier noch anzutreffen! Daher gilt: Ein dynamisches Design, welches sich unterschiedlichen Auflösungen anpasst, ist grundsätzlich sehr nutzerfreundlich und immer empfehlenswert. Wird allerdings ein statisches Design verwendet, so ist darauf zu achten, dass die wichtigen Elemente bei der Auflösung 800x600 Pixel gut sichtbar sind. Nach diesen Punkten kann man folgendes kurzes Zwischenfazit, für eine seniorengerechte Gestaltung einer Webseite geben:

- Vertrauen und Kompetenz vermitteln
- sich in die Gefühle und Situation der „Silver-Surfer“ versetzen
- die Senioren in ihrer Sprache ansprechen
- die wichtigen Informationen mehrmals wiederholen
- skalierbare Schrift, bekannte Schriftarten und einen großzügigen Zeilenabstand verwenden
- die traditionellen Werte betonen.

FORMULARE WIE AUF PAPIER

Der erste und wichtigste Tipp: Die digitalen Formulare sollen den originalen Papierformularen ähneln [3]. Der Umgang mit solchen Unterlagen ist den Silver-Age Nutzern vertraut. Für die Formulare gilt: Diese sollen nur die notwendigsten Felder enthalten, denn oft sind die Senioren verunsichert, ob die Daten auch vertraulich behandelt werden. So sollen beispielsweise Fragen zu Marktanalysen lieber unterlassen werden. Ein weiteres typisches Beispiel: Es macht Sinn nur ein Feld für die Eingabe der Straße und Hausnummer zu machen. Studien zeigen, dass diese Eingaben gleichzeitig in einem Feld vorgenommen werden. Bietet man zwei Eingabefelder an, kommt es oft zu unerwünschten Fehlermeldungen, welche für Verwirrung sorgen. Intern müssen beide Eingaben natürlich getrennt in der Datenbank verarbeitet werden. Dazu muss man die Zeichenkette mit Hilfe eines Algorithmus wieder trennen. Ein wenig Mehraufwand in der Programmierung kann

für ein deutliches Plus an Benutzerakzeptanz sorgen. Unsicherheiten entstehen oft auch bei der Eingabe eines Datums oder einer Uhrzeit. Hier sind genaue Vorgaben nützlich. Die Eingabe des Geburtsdatums sollte ohne Auswahllisten erfolgen. Die Eingabe per Tastatur ist vertrauter.

DIE SACHE MIT DEN APPS

Auch diese Zielgruppe tauscht vermehrt ihr nur zur Telefonie geeignetes Mobiltelefon in ein moderneres Smartphone. Gibt es spezielle Apps für Senioren? Ja die gibt es [4]. Einige Beispiele:

■ *Simple Phone Senior*: Der Name dieser App erklärt sich von alleine. Das Ziel: Die primären Funktionen eines Smartphone sofort zugänglich zu präsentieren (Abbildung 1). Es handelt sich um eine App, welche separat gestartet werden muss. Mit einem Klick auf den Home-Button landet der Nutzer wieder auf den Startbildschirm des Standard-Launchers. Tatsächlich ist diese App nur für Menschen mit extremen Sehschwächen oder motorischen Problemen beim Telefonieren hilfreich.

■ *Big Launcher*: Es handelt sich um keine klassische App, sondern tatsächlich um einen Launcher (Abbildung 2). Nur für ältere Personen? Eindeutig nicht! Es gibt genug junge Leute, die ein älteres Smartphone oder sogar noch ein klassisches Handy nicht eintauschen. Ihr Argument: Das Gerät kann alles was ich brauche! Endlose Menüs, kleine Buttons und Schriften finden sie im Alltag einfach nicht ergonomisch in der Handhabung. Big Launcher ermöglicht es, die komplette Benutzeroberfläche neu und einfach zu gestalten. Inwieweit diese dabei auch seniorengerecht gestaltet wird, kann jeder selbst entscheiden. Bestimmte Funktionen lassen sich bewusst einschränken oder verhindern. Dabei wird der Home-Screen in zwei Reihen und vier Zeilen eingeteilt. Die Seiten können themenbezogen eingerichtet werden. Der SOS-Button kann individuell konfiguriert werden, z.B. eine Textnachricht, die im Notfall an eine oder mehreren Personen gesendet wird. Ebenso ist es möglich, die aktuelle GPS-Position zusammen mit

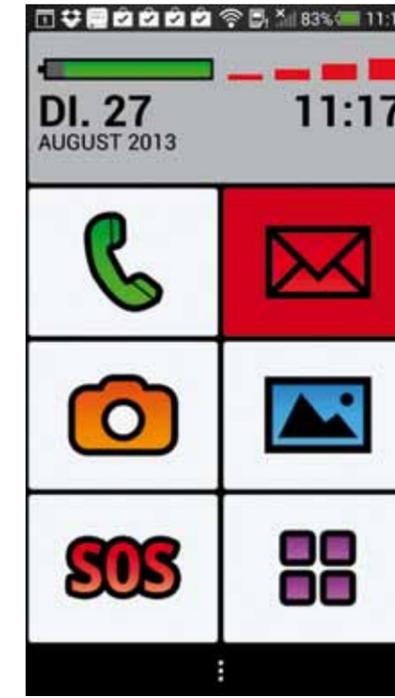


Abbildung 2: Big Launcher tauscht die klassische Benutzeroberfläche, für eine Konzentration auf das Wesentliche.

der Nachricht zu versenden. Eine Demoversion gibt es kostenlos. Für eine Vollversion muss man 8 Euro bezahlen.

■ *Protegon SOS*: Diese App richtet sich an alle Personen, welche aus beruflichen oder sonstigen Gründen rund um die Uhr einen Notruf an bestimmte Zielgruppen weiterleiten wollen oder müssen. Dabei handelt es sich nicht nur um Senioren, sondern auch um Menschen mit Behinderungen oder andere Personen, welche im Ernstfall auf fremde Hilfe angewiesen sind. Die Oberfläche der App besteht aus nur zwei Tasten bzw. Schaltflächen: rot für einen medizinischen Notfall und blau für sonstige Notfälle. Die zu übermittelnde Nachricht enthält die Standortdaten und ggf. ein Foto. Diese Informationen werden an die Sicherheitszentrale gesendet. Viele Hilfsstellen sind an diesen Dienst angeschlossen. Die Nutzung der App beträgt 6,95 Euro/monatlich.

■ *Thema Gesundheit*: Wie bereits erwähnt, gewinnt das Thema Gesundheit mit zunehmenden Alter immer mehr an Bedeutung. Oft handelt es sich um die alltäglichen Sorgen und es geht um die Anwendung von Hausmitteln. Die App *99 Tipps* für mehr Gesundheit beinhaltet übersichtlich

sortierte Ratschläge und Empfehlungen für jeden Tag. Die Bedienung ist einfach. Ebenso zum Thema Gesundheit gehört die App *Apotheke vor Ort*. Diese App zeigt die nächsten verfügbaren Apotheken und Notdienste. Ebenso kann man Medikamente vorbestellen und den Beipackzettel abrufen. *Pill Reminder* by *Drugs.com* erinnert an die Einnahme von Medikamenten. Dabei besteht eine Möglichkeit die Einstellungen so einzurichten, dass mehrere Anwender bzw. Familienmitglieder helfen können. *Blood Pressure Monitor* protokolliert wichtige Gesundheitsdaten. Mithilfe einer E-Mail-Export-Funktion kann die Kommunikation mit Ärzten und Apotheken vereinfacht werden.

HARDWARE

Kann man bestimmte Hardware als bevorzugt ausmachen? Wenn das so ist, kann sich die Programmierung der Software auch danach ausrichten. Leicht bedienbare Tablet-PCs bieten für die Senioren ein gute Alternative zum Computer. Touchscreens, Sprachsteuerung und größenverstellbare Symbole sprechen dafür. Die Geräte eignen sich gut zum Schreiben von E-Mails, zum Lesen von Büchern und Zeitschriften, zum bloßen Surfen im Internet oder zum Telefonieren. Andererseits gilt auch: Zum Erstellen von umfangreichen Dokumenten oder zur Bildbearbeitung sind immer noch klassische Desktop-PCs bzw. Laptops die bessere Variante. Welches Gerät eignet sich am besten für die Zielgruppe der Silver-Surfer? Das iPad als „Rentner-Rechner“ [5] [6]? Ganz falsch ist der Gedanke nicht! Der Funktionsumfang ist übersichtlich und einfach. Die Qualität des kontraststarken Bildschirms trägt dazu bei, dass die Texte gut lesbar sind. Die Bedienung ist sehr intuitiv, das Blättern von Seiten funktioniert wie in der realen Welt, der Text lässt sich einfach und schnell vergrößern und die Symbole trifft jeder, selbst wenn er noch nie ein Tablet bedient hat. Die Studien beweisen, dass die Hersteller des iPads auch die Silver-Generation erreicht haben. Apple punktet mit den Modellen iPad Air und Apple iPad mini. Eine Alternative zu diesen Geräten bieten die Hersteller Samsung, Sony und HTC. Das HTC Google

Nexus 9 gilt als eines der besten Android-Tablets. Vor allem überzeugt er durch die flüssige Performance und das überdurchschnittliche Display. Negativ ist jedoch vor allem die geringe Akku-Laufzeit. Samsung Galaxy Tab S, mit einem 8,4 oder 10,5 Zoll Display, kombiniert die Performance und guter Akkulaufzeit mit einem brillanten Display. Das Gerät hat sich zu einem Liebling der Kunden entwickelt. Nur die vergleichsweise lange Ladezeit sorgt für einen Punktabzug.

Zusammengefasst: Die Geräte müssen eine gewisse Mindestgröße aufweisen, damit sie von der älteren Generation akzeptiert werden. Noch mehr als bei anderen Personengruppen ist hier eine vertraute und intuitive Bedienung gefragt. Moderne Entwicklungen, wie eine gute Spracherkennung, bieten in der Zukunft viel Potenzial für eine Erhöhung der Benutzerakzeptanz. Voraussetzung ist, dass die Funktionen sehr gut funktionieren, denn die Bereitschaft zum Experimentieren, ist in der betrachteten Zielgruppe nicht so stark ausgeprägt.

FAZIT UND AUSBLICK

Die zielgruppengerechte Ausrichtung digitaler Produkte, wie Webseiten oder Apps für Smartphone und Tablet sowohl in Inhalt, als auch in Design und Bedienung ist nicht neu. Zunehmend ist jedoch die Ausrichtung an eine ältere Zielgruppe. Diese Personen sind nicht mit der Technik

„groß“ geworden. Der Antrieb ist die Aussicht auf neue Kunden und als IT-Dienstleister den Markt mit passgenauen Angeboten zu überzeugen. Dem aufmerksamen Leser dürfte deutlich geworden sein, dass dieses Thema einige Schnittmengen mit der barrierefreien Gestaltung von digitalen Produkten aufweist. Das wiederum würde aber einen eigenen Beitrag rechtfertigen.

LITERATUR UND LINKS

- [1] https://www.destatis.de/DE/PresseService/Presse/Pressekonferenzen/2015/generation65/Pressebrochure_generation65.pdf?__blob=publicationFile
- [2] <http://www.fit-fuer-usability.de/archiv/usability-fuer-senioren/>
- [3] http://old.hki.uni-koeln.de/studium/MA/MA_streich.pdf
- [4] <https://www.android-user.de/15-app-empfehlungen-fuer-senioren-und-den-ruhestand/>
- [5] <http://www.welt.de/wirtschaft/webwelt/article7909118/Das-iPad-erreicht-auch-die-Generation-65.html>
- [6] <https://www.basenio.de/senioren-ratgeber/technik/tablet-fuer-senioren-test-vergleich-2016-tipps-146/>

ANZEIGE

VisualStudio1

Hinter VisualStudio1 steckt viel mehr als eine Zeitschrift für die Microsoft Developer Community. VisualStudio1 steht für den aktiven Wissenstransfer von, mit und für Softwareentwickler. In unserem Magazin veröffentlichen wir detaillierte Artikel rund um das Thema Software-Entwicklung mit Microsoft-Technologien. Aber auch andere Plattformen werden betrachtet.

Visual Studio 1 – Zeitschriftabonnement

Die Visual Studio 1 erscheint mit 4 Exemplaren in einem Abstand von jeweils 3 Monaten. Erhältlich ist sie zu einem Jahrespreis von 32€ in Deutschland und 35€ im EU-Ausland. Als kleinen Bonus spendieren wir beim Erwerb eines Jahres-Abos eine gratis VS1-Tasse.

Studios – Film ab!

Informationsvideos z.B. von Konferenzen oder Trainings werden in unserem Videobereich regelmäßig upgedatet, damit Sie auch jenseits von Öffnungszeiten oder bei verpassten Events von unserem Knowhow profitieren können.

Hier gelangen Sie in unseren Videobereich: <http://studios.ppdv.de>



VISUAL STUDIO PREVIEW 4 WAS BRINGT ES?



Autor: ANNETTE HEIDI BOSBACH

Visual Studio hat in den letzten Jahren eine beeindruckende Entwicklung hingelegt: Mittlerweile taugt das Produkt sogar zur Entwicklung von Android-Applikationen.

Mit der vor wenigen Tagen erschienenen vierten Preview der Version "15" halten einige interessante Funktionen Einzug in das Produkt. Dieser Artikel soll Ihnen hier die Besten kurz vorstellen und Ressourcen anbieten, wo Sie mehr erfahren können.

SCHNELLERE INSTALLATION

Das Deployment von Visual Studio ist eine Geduldssübung. In Version 4 wird ein neuer Installationsprozess verwendet (<https://blogs.msdn.microsoft.com/visualstudio/2016/04/01/faster-leaner-visual-studio-installer/>). Auf einer Windows 10-VM vergingen nach dem Ausführen der Setup-Datei rund 30 Sekunden. Danach erschien ein Dialogfenster, in dem man die zu bearbeitenden Aufgaben anklicken konnte. Als erstes folgte die Aktivierung der Option "Universal Windows Plattform development", was zu einer Installationsgröße von 3,75 GB führte. Das Hinzufügen aller Komponenten führte laut dem Installationsassistenten zu einer Gesamtgröße von rund 4,85 GB.

DERZEIT OHNE AZURE

Zum Zeitpunkt der Drucklegung steht das Azure-Workload noch nicht zur Verfügung.

Die Berechnungen des Installationsassistenten erwiesen sich als mehr als optimistisch: Vor der Installation hatte die virtuelle Maschine 32,7 GB freien Speicherplatz, nach der Installation aller Workload waren es rund 5 GB.

Der Installationsassistent lädt die benötigten Daten dynamisch von einem Server aus dem Hause Microsoft: Fehler der Bauart "Failed to download the version requirements." sind in den meisten Fällen ignorierbar.

Ein witziges Detail ist, dass eine auf derselben Maschine installierte normale Version von Visual Studio 2015 während des Setup-Prozesses problemlos weiterfunktionerte. Das gilt leider nicht für frühere Versionen der Preview: v4 ist mit seinen Vorgängern nicht kompatibel, und verlangt deren Deinstallation im Rahmen der Setup Routine.

Ein sehr nettes Detail im Zusammenhang mit Visual Studio "15" ist, dass die IDE nicht nur unter Windows 10 funktioniert. Microsoft unterstützt auch ältere Betriebssysteme incl. Windows 7 ab Service Pack 1.

SPARE ZEIT UND NERVEN

Kleine Einsparungen können unterm Strich massive Auswirkungen entwickeln: Robert Crandall sparte seinem Arbeitgeber durch Eliminieren einer einzelnen Olive pro Erste-Klasse-Passagier rund 40 000\$ im Jahr.

Diese Philosophie wird auch in Visual Studio 15 verfolgt: Nach dem Start der IDE finden Sie in der Rubrik New ein Eingabefeld, das den bisher verwendeten Projektskelett-Auswahldialog

arbeitslos macht. Tippen Sie einfach einen Teil des Namens des gewünschten Skeletts ein, um sich an der in Abbildung 1 gezeigten Liste zu erfreuen.

Auch im Rest der IDE wurden Erweiterungen vorgenommen. Ein interessantes Feature ist die auf der Unterseite eingeblendete violette Zeile: Ist ein Projekt mit einem Versionskontrollsystem verbunden, sieht der Entwickler dort permanent hilfreiche Informationen zum Zustand seiner Codebasis.

In Zeiten der Mobilität - der durchschnittliche Entwickler hat heute wahrscheinlich ein Tablet, ein Notebook und mindestens eine Workstation - ist die automatische Synchronisation das effizienteste Mittel zur besseren Nutzung von Lebenszeit. Da Visual Studio seit einiger Zeit mit einem Onlinedienst verbunden ist, nutzte Microsoft mit Visual Studio 15 die Gelegenheit, dieses

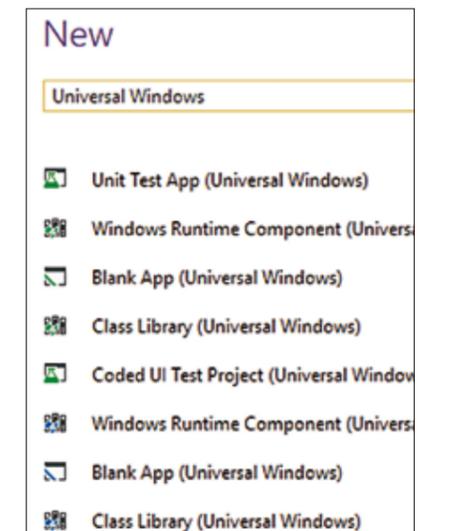


Abbildung 1: Wählen Sie nach Belieben!

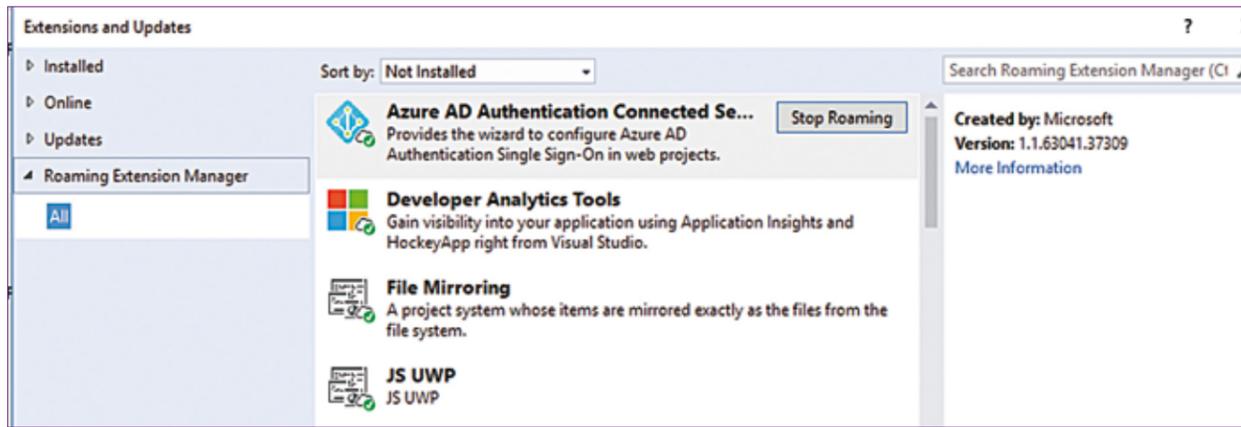


Abbildung 2: Die als Roaming markierten Erweiterungen lassen sich auf neuen Visual Studio-Instanzen installieren.

Feature zu erweitern. Der Roaming Extension Manager erlaubt Entwicklern das Festlegen von Extensions, die mit allen Instanzen geteilt werden sollen. Sie lassen sich sodann auf neuen Workstations mit minimalem Aufwand herunterladen (siehe Abbildung 2).

Wer mit mehreren Projekten in einer Projektmappe arbeitet, freut sich über eine kleine Erweiterung in der Toolbar. Microsoft spendiert nun von Haus aus eine weitere Combobox, in der man das nach dem Start anzuwerfende Projekt auswählen darf.

Visual Studio wird immer mehr als Editor für Sprachen und Dienste benutzt, die eigentlich nicht aus der Microsoft-Welt stammen. Die Preview von VS 15 trägt diesem Aspekt durch eine wesentliche Erweiterung des Editors bei, da er nun die in der Tabelle unterstützten Sprachen farblich hervorheben und/oder reflektieren kann.

IntelliSense war bisher auf genaues Tippen angewiesen: Machte der Entwickler einen Fehler beim Eingeben des Namens, so konnte die Logik dem betreffenden Namespace und/oder das betreffende Element nicht finden. Visual Studio 15 schafft an dieser Stelle mit einem auf klassischen Rechtschreibkorrekturen

basierenden Verfahren Abhilfe, das die auf modernen Workstations zur Verfügung stehende zusätzliche Rechenleistung zur Verbesserung des Suchprozesses einbindet (siehe Abbildung 3).

Die Umsetzung bzw. Nichtumsetzung von Code-Stil-Spezifikationen in Projekten führt mitunter zu massivem Unmut im Team: insbesondere dann, wenn die Regeln nicht zu 100% klar sind. Visual Studio 15 löst dieses Problem durch den in Abbildung 4 gezeigten Dialog. Er versteckt sich im References-Ordner des jeweiligen Projekts, das nun auch ein Element namens Analyzers enthält. Klicken Sie dieses mit der rechten Maustaste an und wählen Sie die Option "Open Active Rule Set". Falls sich die Option "Microsoft.AnalyzerPowerPack" nicht findet, können Sie das fehlende Feature per NuGet nachinstallieren.

Visual Studio überprüft in diesem Fall, ob der vom Entwickler erzeugte Code mit den "hauseigenen" Spielregeln übereinstimmt. Ist dies nicht der Fall, so kann das Produkt während der Kompilation nach Wunsch einen Fehler, eine Warnung oder nur eine Informationszeile anzeigen.

Eine andere Erweiterung von

IntelliSense spart Entwicklern bei der Erzeugung von auf Enums basierenden Selektionen Tipparbeit. Ein Beispiel dafür wäre folgende select-Anweisung, die bisher nur einen Teil ihres "Mater-Enums" abdeckt:

```
enum Days { Sat, Sun, Mon, Tue, Wed, Thu, Fri };
```

```
static void Main(string[] args) {
    Days aDay = Days.Sat;
    switch (aDay) {
        case Days.Mon:
            break;
        case Days.Tue:
            break;
    }
}
```

Visual Studio 15 blendet beim Vorhandensein einer derartigen Schleife ein Glühbirnensymbol ein, welches das Anlegen fehlender Teile automatisiert – Abbildung 5 zeigt, was die IDE bei unserer Beispielschleife anbieten würde.

NEUES FÜR C#

Seit der Einführung des .Net-Frameworks gewinnt C# immer mehr Popularität – unter anderen auf Kosten seines "Vorgängers" Visual Basic. Die mit Visual Studio 15 ausgelieferte Version von C# bringt Entwicklern eine Vielzahl neuer Funktionen mit, die bisherige Probleme der Sprache beheben sollen.

Eine von Rüstungs- und allgemeinen

REDE MIT!

Die Arbeiten an Visual Studio's Compilerinfrastruktur erfolgen seit einiger Zeit in der Öffentlichkeit. Wer teilnehmen möchte, kann dies im unter <https://github.com/dotnet/roslyn> bereitstehenden GitHub-Repository tun.

Farbliche Hervorhebung, teilweise Autocomplete	+ Code Snippets	+ Sprünge zu Deklaration
Bat	CMake	C++
Clojure	C++	C#
CoffeeScript	C#	Go
Docker	Go	Java
F#	Groovy	JavaScript
Groovy	HTML	PHP
INI	Java	TypeScript
JSON	PHP	VB
...

Ob hier der auch unter Linux verfügbare Visual Studio Core Pate stand?

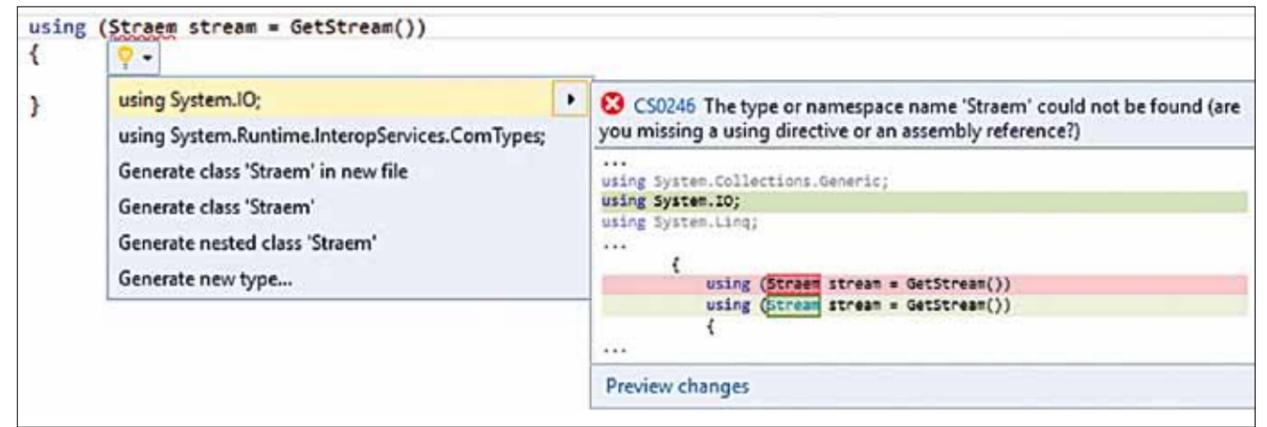


Abbildung 3: Dieser Rechtschreibfehler sorgt nun nicht mehr für Probleme (Bildquelle: Microsoft)

Elektronikern seit Jahren erwünschte Funktion – Entwickler mit wenig Fokus auf Elektronik rollen hier vielleicht die Augen – ist die Möglichkeit, binäre Konstanten direkt, also als Abfolge von Nullen und Einsen in das Codefile zu schreiben. Diese auf den ersten Blick kleine Änderung spart in der Praxis immens Zeit, weil sie die immer auftretenden Fehler beim Konvertieren von in Datenblättern abgedruckten binären Registern in Hexadezimalkonstanten eliminieren. Zum Ansehen dieses Feature reicht ein kleines Konsolenprogramm, das folgendermaßen aussieht:

```
static void Main(string[] args){
    int meinBinaer4 = 0b100;
    int meinBinaerSpacey = 0b1001_0001;
    Console.WriteLine(meinBinaer4 + „ „ +
        meinBinaerSpacey);
    Console.Read();
}
```

Die Nutzung des Präfixes ob leitet eine derartige binäre Ziffernfolge ein. Wer seine Binärgruppen aus optischen Gründen unterteilen möchte, da Register z.B. bei der Nutzung mit 32 oder gar 64bit-Prozessoren besonders lang werden, kann dies per _ tun.

Eine weitere Frage in diesem Zusammenhang ist die Feststellung der in ten Literalen verwendeten Endianness. Dies lässt sich durch folgendes Programm überprüfen:

```
static void Main(string[] args)
{
    int meinBinaerLang = 0b10000000_00000000;
    Console.WriteLine(meinBinaerLang);
    Console.Read();
}
```

Das Ausführen des Programms ergibt die Ausgabe der Zahl 32768. Damit ist erwiesen, dass C# im Big Endian-Modus arbeitet. Die Nutzung von _ ist nicht nur auf binäre Literale beschränkt: Sie können den Unterstrich auch einsetzen, um „klassische“ Zahlen leichter sichtbar bzw. mental angreifbar zu machen. Angemerkt sei zudem, dass Binärliterale nicht nur in C# zur Verfügung stehen. Wer in Visual Basic 15 programmiert, kann das Feature ebenfalls nutzen. Der einzige Unterschied ist, dass statt ob das Präfix &B zum Einsatz kommt.

```
Enum MouseEventFlags
    MOUSEEVENTF_ABSOLUTE = &B0000_1_0000_000000_0
    MOUSEEVENTF_LEFTDOWN = &B0000_0_0000_000001_0
```

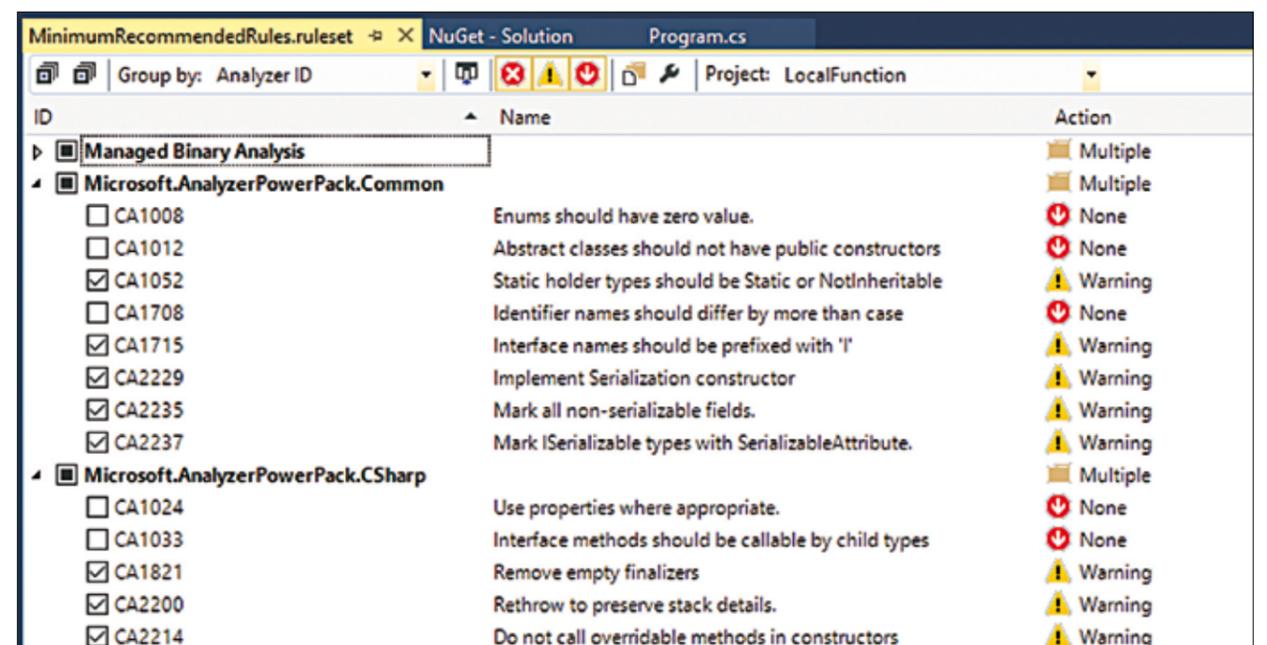


Abbildung 4: Diese Funktion spart die eine oder andere Streiterei im Team

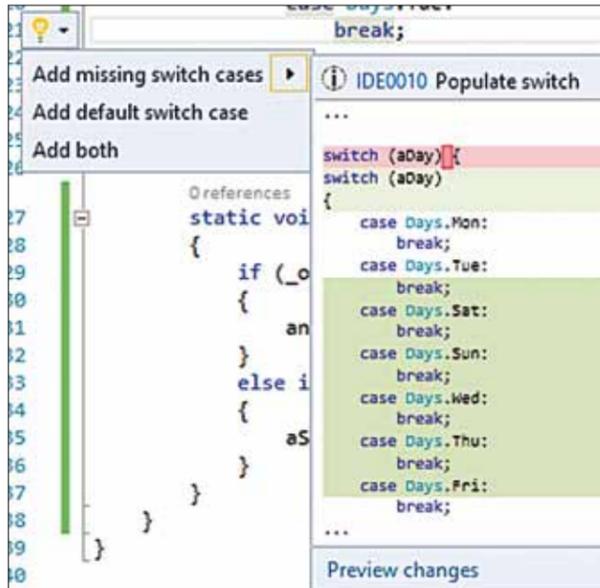


Abbildung 5: Das Anklicken der Glühbirne wird belohnt

EINGEBETTETE FUNKTIONEN

Das in der Kolumne zu NuGet gezeigte Beispiel ist ideal geeignet, um die Vorteile lokaler Funktionen zu illustrieren. Die Ermittlung des Maximums eines Felds wäre ein Codeteil, der nicht unbedingt alleine stehen muss. Mit C# 7.0 ließe sich der relevante Teil des Programms folgendermaßen umschreiben:

```
static void Main(string[] args){
    int[] myBinning = new int[26];
    int max = 0;
    void Max(){
        for (int i = 0; i < 26; i++){
            if (max < myBinning[i]) max = myBinning[i];
        }
    };
    Max();
}
```

Eine lokale Funktion ist insofern komfortabler als ihr globaler Kollege, als sie analog zu Javascript – die Elemente hören dort auf den Namen Closure – in der Lage ist, auf die Elemente ihrer Umgebung zuzugreifen. Man muss das Binningarray nicht separat exponieren, um seine Bearbeitung per Max möglich zu machen.

Dass dieses Auslagern bei einer einmaligen Ausführung der Maximumberechnung nur wenige Vorteile bringt, ist logisch – stellen Sie sich allerdings einen Algorithmus vor, in dem mehrere Maximumberechnungen hintereinander erforderlich sind.

Lokale Funktionen müssen übrigens nicht Parameterlos sein, und können sogar einen eigenen Rückgabewert anliefern. Eine diesbezügliche Variante des Programms würde folgendermaßen aussehen:

```
int Max(){
    for (int i = 0; i < 26; i++){
        if (max < myBinning[i]) max = myBinning[i];
    }
    return max;
};
max=Max(); //tautologisch
```

Ein weiteres lustiges Feature lokaler Funktionen ist, dass sie zur Nutzung von CallerInfo-Attributen befähigt sind.

Dies ist insbesondere beim Debuggen nützlich – die folgende Version von Max ist instrumentiert:

```
int Max([System.Runtime.CompilerServices.CallerMemberName] string _name="", [System.Runtime.CompilerServices.CallerLineNumber] int _liNu = 0){
    Console.WriteLine(„Aufrufer: „ + _name + „, Zeile: „ + _liNu);
    . . .
    return max;
};
```

Die Nutzung der unter <https://msdn.microsoft.com/en-us/library/mt653988.aspx> im Detail beschriebenen und nicht nur unter VS 15 zur Verfügung stehenden Attribute ist nicht sonderlich kompliziert. Erstellen Sie optionale Parameter (sie müssen optional sein), und lassen Sie sie beim Aufruf der Methode frei. Danach steht es Ihnen frei, die im Moment drei verfügbaren Callouts nach Belieben auszuwerten.

ERWEITERTE VERGLEICHSMÖGLICHKEITEN

Eine der wichtigsten Aufgaben im Leben des Entwicklers ist das Vergleichen von Soll- und Istwerten: Schleifen und andere Programmstrukturen wären ohne Vergleiche nicht realisierbar.

Bisher war die Vergleichssyntax von C# – insbesondere im Vergleich zu Regular Expressions und Ähnlichem – eher schwach ausgeprägt und entsprach im Großen und Ganzen dem, was Kerningham und Ritchie spezifizierten.

C# 7.0 schafft an dieser Stelle durch ein als Patterns bezeichnetes Feature Abhilfe. Als erstes Beispiel dafür möchte ich die Erweiterungen am is-Operator ansehen.

Die folgende Methode nimmt ein Objekt entgegen, und konvertiert es im Rahmen der Selektion entweder in Integer- oder in eine Stringrepräsentation:

```
static void doSomething(object _o) {
    if (_o is int anInt) {
        anInt++;
    }
    else if (_o is String aString) {
        aString=aString.ToLower();
    }
}
```

Interessant ist hier vor allem, dass das Pattern automatisch eine neue lokale Variable erzeugt, die den gewünschten Typ aufweist. Dies erspart Entwicklern den einen oder anderen Cast, und trägt zudem zu kompakterem Code bei.

Ein weiteres nettes Beispiel ist die Erweiterung des Case-Operators. Er ist nun zur Abarbeitung von Patterns und komplexen Abfragen befähigt. Ein Beispiel dafür ist die folgende von Microsoft in der Dokumentation bereitgestellte Selektion, die bisher nur mit if mögliches Verhalten realisiert und dabei die Frage aufwirft, in welcher Reihenfolge der Compiler die einzelnen Case-Testfälle abarbeitet:

```
switch(shape)
{
    case Circle c:
        WriteLine($"circle with radius {c.Radius}");
        break;
    case Rectangle s when (s.Length == s.Height):
        WriteLine($"{s.Length} x {s.Height} square");
}
```

ES GIBT MEHR!

Die Patternfunktionalität war „größer“ geplant; Microsoft musste laut <https://www.infoq.com/news/2016/05/csharp7-pattern-matching-removed> einige Funktionen entfernen, weil sie wahrscheinlich nicht rechtzeitig fertig werden

```
break;
case Rectangle r:
    WriteLine($"{r.Length} x {r.Height} rectangle");
    break;
default:
    WriteLine(„<unknown shape>“);
    break;
case null:
    throw new ArgumentNullException(nameof(shape));
}
```

Durch die Einführung von Patterns in Switch-Selektionen ergeben sich wesentliche Änderungen an der Reihenfolge der Abarbeitung: In „klassischem“ C# wäre wegen der zufälligen Exekutionsreihenfolge nicht sichergestellt, dass ein „quadratisches“ Rechteck nicht versehentlich in den Case für normale Rechtecke fällt.

Der Compiler arbeitet Switch-Kommandos nun von oben nach unten ab: Der erste zutreffende Case wird ausgeführt. Das gilt allerdings nicht für default, welches immer als allerletztes abgearbeitet wird.

EFFIZIENT VERPACKT

Das Zurückgeben von mehreren Werten aus einer Funktion erfolgte bisher – ganz wie bei Kerningham und Ritchie – durch Anlegen eines Structs. C# 7.0 schafft hier mit einem als Tupel bezeichneten Konstrukt Abhilfe: Es handelt sich dabei um eine Art „ad hoc“-deklariertes Datenfeld, das mehrere Untervariablen zusammenfasst.

Ein klassisches Beispiel für die Verwendung eines Tupels wäre die folgende Methode, die dem Aufrufer zwei Werte zurückliefert. Dieser spricht sie dann über ihre „sequentiellen“ Namen an:

```
static void Main(string[] args) {
    var something = worker();
    something.Item1=something.Item1;
}

static (string, string) worker() {
    return („Hello“, „World“);
}
```

In manchen Fällen kommt es bei der Kompilation dieses Beispiels zu einem Fehler der Bauart „Predefined type ‚System.ValueTuple‘z' is not defined or imported“. Er lässt sich per NuGet beheben – installieren Sie das betreffende Paket in der Kommandozeile (!) durch Eingabe von „Install-Package System.ValueTuple -Pre“.

Eine weitere interessante Möglichkeit ist das Vergeben von Namen an die einzelnen Elemente des Tupels. Auf diese Art und Weise kann der Aufrufer auf die Elemente leichter zugreifen: Berücksichtigen Sie allerdings, dass die Item-Syntax weiter zur Verfügung steht.

```
static void Main(string[] args) {
    var something = worker();
    something.a=something.b;
}
static (string a, string b) worker() {
    return (a: „Hello“, b: „World“);
}
```

SCHNELLER AUTOMATISCH

Auth-Variablen waren bis hier insofern lästig, als man sie nicht mittels var vorausdeklarieren konnte. Dies führte zum in nachfolgenden Schema aufgebauten Code:

```
public void PrintCoordinates(Point p) {
    int x, y; // have to „predeclare“
    p.GetCoordinates(out x, out y);
    WriteLine($"{x}, {y}");
}
```

Microsoft möchte diesen Missstand in C# 7.0 durch eine Erweiterung von Runtime und Parser beheben, um Entwicklern folgendermaßen verkürzte Implementierungen zu ermöglichen:

```
public void PrintCoordinates(Point p) {
    p.GetCoordinates(out int x, out int y);
    WriteLine($"{x}, {y}");
}
```

Leider ist diese Funktion laut <https://blogs.msdn.microsoft.com/dotnet/2016/08/24/whats-new-in-csharp-7-0/> nicht Teil der Preview 4 – noch ist nicht klar, ob das Feature bis zum Erscheinen von Visual Studio 15 fertig wird.

MAIN LERNT LAUFEN

Insbesondere in kleineren Programmen sieht man immer wieder eine main()-Methode, die einen asynchronen Runner aufruft. Dies liegt daran, dass der Sprachstandard von C# bisher keine asynchronen Main-Methoden erlaubt.

Laut dem in GitHub öffentlich zugänglichen Entwicklungsplan (siehe <https://github.com/dotnet/roslyn/issues/7476>) überlegen die Entwickler derzeit, ob sie diese Situation entschärfen wollen. Die eingebrachte Spezifikation sieht die Unterstützung folgender neuer Einsprungpunkte vor:

```
async Task Main()
async Task<int> Main()
async Task Main(string[])
async Task<int> Main(string[])
```

Aufgrund potentieller Probleme mit existierendem Code ist allerdings noch nicht sicher, ob diese Funktion auch wirklich in das Finale C# einfließen wird – wer die Geschicke der Spezifikation beeinflussen will, kann dies unter der oben genannten URL tun.

FAZIT

Das Erweitern eines reifen Produkts ist immer schwierig: Aktuelle Versionen von Visual Studio sind extrem leistungsfähig. Der offene und kundennahe Ansatz von Visual Studio 15 erleichtert Microsoft die Pflege insofern, als reife Produkte eigentlich nur durch massive Berücksichtigung von Kundenfeedback verbessert werden können. Klar ist indes schon jetzt, dass die neuen Features an vielen Stellen Zeit sparen.

Dem Windows Subsystem für Linux auf der Spur

Autor: TIM BOROWSKI

Was ist das Windows Subsystem for Linux?

Das Windows Subsystem for Linux (WSL) ist eine von Microsoft entwickelte Technologie, um Anwendungen, nativ unter Windows laufen lassen zu können, die für Linux gebaut wurden. Das Ausführen von Programmen im WSL unterscheidet sich technisch gesehen signifikant von den herkömmlichen Technologien, wie einer virtuellen Maschine (VM) oder dem Cross-Compiling. In diesem Artikel gehen wir dem WSL auf den Grund.

Der Kernel und das Betriebssystem

Ein Betriebssystem ist mehr als nur der Kernel. Windows setzt auf den hauseigenen NT-Kernel und diverse andere Betriebssysteme wie Ubuntu, Fedora, Arch und Android setzen auf den Linux Kernel oder einem Fork dessen, der von dem Finnen Linus Torvalds initiiert wurde. In der Technik, der Philosophie und im Lizenzmodell sind sich der NT-Kernel und der Linux Kernel in kaum einem Punkt ähnlich.

Kernelname	Linux	Windows NT
Technik	Monolithisch	Modular
Lizenz	u.a. GPLv2	Proprietär
Entwickler	Diverse	Microsoft

Was macht der Kernel?

Der Kernel ist im Grunde aus der Sicht eines Nutzers mit das Irrelevanteste, was es gibt. Ein Kernel sorgt sich um die Organisation von Ressourcen wie Arbeitsspeicher, Prozesse, Threads. Er plant diese Prozesse und Threads und stellt sonstige grundlegende Funktionalitäten wie Rechte- und Benutzerverwaltung zur Verfügung. Ein Betriebssystemkern für sich alleine genommen bietet keinen Mehrwert. Man benötigt Programme, die auf diesem Kern aufbauen.

```
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <string.h>
4
5 const int stdoutHandle = 1;
6 const int stderrHandle = 2;
7
8 void printFileHandle(int handle){
9     // Stringkonkatenation
10    char str[9] = "";
11    sprintf(str, "%d", handle);
12    char fileHandleMessage[38] = "Open Success, FileHandle = ";
13    strcat(fileHandleMessage, str);
14    strcat(fileHandleMessage, "\n\n");
15
16    // Systemcall, schreibt das FileHandle in die Standardausgabe
17    write(stdoutHandle, fileHandleMessage, 38);
18 }
19
20 /**
21  * Beispielanwendungen mit Linux - SystemCalls open, write, close
22  */
23 int main()
24 {
25     // schreiben in die Ausgabe mit normalen Write-Befehl als Systemcall
26     write(stdoutHandle, "Program started \n\n", 18);
27
28     // holen eines FileHandles als SystemCall, daher nicht fopen, sondern open
29     int filedesc = open("file.txt", O_WRONLY | O_CREAT);
30     if(filedesc < 0){
31         return 1;
32     }
33
34     // Ausgabe des FileHandles
35     printFileHandle(filedesc);
36
37     // SystemCall zum Schreiben in eine Datei
38     // write gibt zurück, wieviele Zeichen geschrieben worden sind.
39     int length = 11;
40     int written = write(filedesc, "Hello File\n", length);
41
42     // Schließen des Dateistreams
43     close(filedesc);
44
45     // Prüfen, ob das Schreiben erfolgreich war
46     if(written != length)
47     {
48         write(stderrHandle, "Error writing file.txt\n");
49         return 1;
50     }
51
52     write(stdoutHandle, "Writing successful \n\n", 21);
53     return 0;
54 }
```

Abbildung 1: Beispielanwendung mit einigen System Calls

System-Calls

Wie kann ein Programm den Kernel nutzen? Jetzt geht es richtig tief runter auf die Ebene von Registern und Interrupts. Wenn wir bspw. in C einen

Dateizugriff programmieren, dann arbeiten wir bereits unbemerkt mit solchen System-Calls. Der Compiler bzw. die Library sorgt dafür, dass bei einem Dateizugriff genau der richtige System-Call ausgeführt wird. Dafür



Abbildung 2: Aktivieren des Windows-Features

müssen die richtigen Speichereinheiten in der CPU, die sogenannten Register und der Stack mit den richtigen Informationen befüllt werden und dann die Kontrolle vom Benutzerprogramm auf den Kernel übergeben werden. Dies geschieht mit einer speziellen CPU-Instruktion, die dann eine Ringtransition zur Folge hat. Die x86 Prozessorarchitektur spezifiziert vier Ringe, die meisten Architekturen bieten nur zwei an. Zu jedem Zeitpunkt ist die CPU in genau einem Ring. Abhängig vom aktuellem Ring dürfen bestimmte Aktionen ausgeführt werden und andere nicht. Der Benutzercode läuft im Ring 3 und der Kernelcode läuft im Ring 0. Wird der Kernel aufgerufen, dann liest dieser zu Beginn die Register bzw. den Stack und führt die gewünschte Aktion aus. Am Ende befüllt der Kernel die Register mit den erwarteten Rückgabewerten und gibt die Kontrolle dem aufrufenden Programm zurück. Diese Schnittstellen nennt man auch ABI (Application Binary Interface).

Hier liegt auch der Grund, warum die Programme für unterschiedliche Kernel extra kompiliert werden müssen. Das ABI von NT und Linux sind nicht kompatibel. Für den NT-Kernel müssen die Register schließlich mit anderen Werten befüllt werden, als für den Linux-Kernel. Startet man ein für den NT-Kernel kompiliertes Programm auf einem Linux Rechner, so wird der System-Call schiefgehen und umgekehrt genauso. Logisch oder? Hat man dieselbe Prozessorarchitektur

(x86, ARM, MIPS, usw.) und denselben Kernel, so läuft das Programm auch auf unterschiedlichen Betriebssystemen. Ein Binary, das für Ubuntu x64 gebaut wurde, läuft auch prinzipiell unter einem Fedora oder Arch, welche für x64 gebaut wurden, da sie alle auf den Linux Kernel setzen. Der Einfachheit sind hier Fremdbibliotheken und unterschiedliche Kernelversionen aus der Betrachtung ausgenommen. In diesem Programm werden bereits mehrere System-Calls ausgeführt. Mit Absicht wurde in diesem Programm open(.) anstelle von fopen(.) genutzt, denn open(.) erzeugt einen System-Call während fopen(.) eine Library nutzt, die den System-Call ausführt. Es handelt sich bei fopen() demnach strenggenommen um einen Function-Call. Kompiliert man das Programm mit einem C-Compiler für die Linux Architektur, dann wird bei open() das %rax mit 2 befüllt. In %rax steht immer die eindeutige System-Call Nummer. Das %rdi Register zeigt auf den Anfang eines Dateipfads und %rdx ist in diesem Fall 0. Den Rückgabewert findet man wieder in %rax. Mehr Informationen zu diesem Thema, finden Sie unter den sogenannten Linux System-Call Tables.

Wie funktionieren System-Calls für Linux unter NT?

Wir nehmen ein Programm, das für einen Linux-Kernel gebaut wurde. Gestartet wird das Programm in einem sogenannten Pico-Prozess.

Win32 Prozesse haben verglichen mit Linux Prozessen einen anderen Aufbau. Es gibt in den Linux Prozessen bspw. keinen initialen Thread und keine NTDLL.DLL, über die kommuniziert werden könnte. Bei System Calls werden auch die Register, der Stack und so weiter ordnungsgemäß für den Linux Kernel serviert. Der NT-Kernel kann die System-Calls aus einem Pico-Prozess nicht direkt beantworten. Er interpretiert den Inhalt der Register als Müll und unbrauchbar. Jetzt kommt der Knackpunkt. Der NT-Kernel lässt das Programm nicht abstürzen, da er weiß, dass die Anfrage aus einem Pico-Prozess stammt. Die Informationen werden an die Kernel-Treiber (LXSS.SYS und LXCORE.SYS) weitergegeben. Diese Treiber sind quasi Plug-Ins vom NT-Kernel und enthalten keinen Linux Code, sorgen aber dafür, dass der Kernel die Anfragen beantworten kann. Open könnte bspw. auf das äquivalente NtOpenFile() gemapped werden. Der NT-Kernel unterstützt

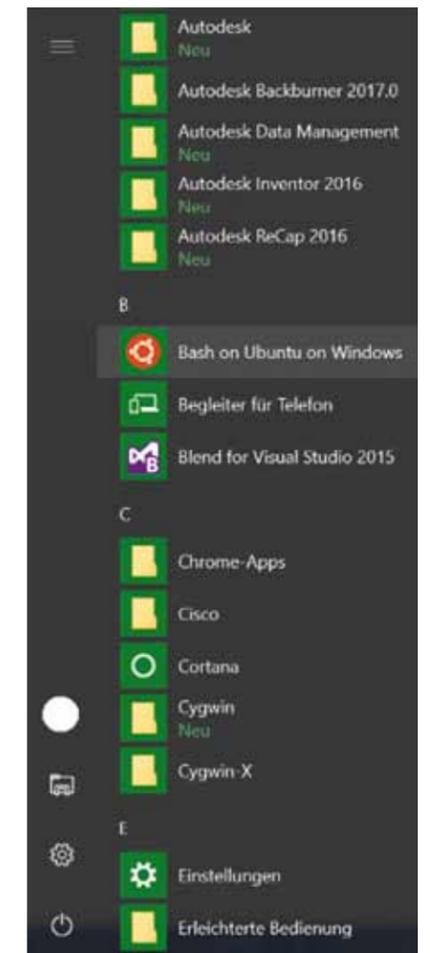


Abbildung 3: Eintauchen in die BASH



Abbildung 4: Kapselung der Benutzer

auch das Kopieren von Prozessen wie `fork()`, diese Funktionalität ist aber für Win32 Prozesse nicht zugänglich. Pico-Prozesse nutzen allerdings diese Funktionalität über die Kernel-Treiber. Der NT-Kernel ist schon im Design darauf ausgelegt worden, mehrere Prozessformate mit einem Kernel zu unterstützen. Zunächst war es aus Abwärtskompatibilitätsgründen entstanden, nun macht das WSL davon Gebrauch, um Linux Prozesse zu unterstützen. Mehr Informationen über den internen Aufbau gibt es im MSDN-Blog [2].

HANDS-ON WSL

Voraussetzung ist, dass man das Anniversary Update von Windows 10 installiert hat oder man im Windows Insider Programm ist. Erfüllt man die Anforderung, so kann man unter Systemsteuerung -> Programme und Features -> Windows-Feature aktivieren oder deaktivieren -> Windows-Subsystem for Linux (Beta) aktivieren. Hat das System das Feature hinzugefügt, sieht man das unscheinbare Programm „Bash on Ubuntu on Windows“.

Öffnen wir das Programm, so sehen wir ein Terminal, in dem die Bash Shell läuft. Die Shell läuft dabei ganz normal im Windows Usermode.

```

timbo@NBRV0PX1:~/mnt/c$ cd /mnt/
timbo@NBRV0PX1:~/mnt/c$ ls
ls: cannot access hiberfil.sys: Permission denied
ls: cannot access pagefile.sys: Permission denied
ls: cannot access swapfile.sys: Permission denied
total 16
drwxr-xr-x 2 timbo timbo 0 Aug 20 15:42 .
drwxr-xr-x 2 root root 0 Apr 11 2014 ..
drwxr-xr-x 2 timbo timbo 0 Aug 20 13:23 .bashrc
-rw-r--r-- 1 timbo timbo 6178 Aug 21 15:05 .bash_history
-rw-r--r-- 1 timbo timbo 220 Jul 16 16:42 .bash_logout
-rw-r--r-- 1 timbo timbo 3717 Jul 26 17:33 .bashrc
drwxr-xr-x 2 timbo timbo 0 Aug 18 22:09 .cache
drwx----- 2 timbo timbo 0 Jul 26 17:37 .config
drwx----- 2 timbo timbo 0 Jul 26 17:11 .dbus
drwx----- 2 timbo timbo 0 Aug 20 13:24 .dbus-keyrings
drwxr-xr-x 2 timbo timbo 0 Aug 20 13:19 .gconf
drwx----- 2 timbo timbo 0 Jul 27 19:43 .gnome2
drwx----- 2 timbo timbo 0 Jul 27 19:43 .gnome2_private
drwx----- 2 timbo timbo 0 Jul 26 17:11 .local
drwx----- 2 timbo timbo 0 Aug 20 13:31 .vnc
-rw-r--r-- 1 timbo timbo 675 Jul 16 16:42 .profile
drwx----- 2 timbo timbo 0 Aug 14 22:19 .ssh
drwx----- 2 timbo timbo 0 Aug 20 13:31 .vnc
-rw-r--r-- 1 timbo timbo 302 Aug 20 13:18 .xauthority
-rw-r--r-- 1 timbo timbo 1762 Aug 20 13:16 .xsession-error
-rw-rw-rw- 1 timbo timbo 35211 Aug 20 15:42 .zcompdump
timbo@NBRV0PX1:~/mnt/c$ cd Users/
timbo@NBRV0PX1:~/mnt/c$ cd Users/

```

Abbildung 5: Navigieren in Eigene Dateien

hinweisen. Während Windows auf den Backslash setzt (Bsp.: C:\Users\...) setzt Linux auf den Slash (Bsp.: /home/...).

Kapselung der Benutzer, root ist nicht gleich root

Die Kapselung der Nutzer und der Dateien sieht folgendermaßen aus: Jeder Windowsnutzer hat ein eigenes, separates Subsystem. Wenn der Windowsbenutzer A in den root-Ordner schreibt, so ist diese Datei nicht für den Windowsbenutzer B sichtbar. Beide Subsysteme sind vollständig voneinander gekapselt. In einem reinen Linuxsystem sind alle Dateien im Rootverzeichnis für alle Nutzer mit entsprechenden Berechtigungen sichtbar. Selbst ein Superuser oder Rootuser in einem Linux Subsystem kann nur auf die Dateien in seinem Subsystem zugreifen. Dadurch braucht man sich auch keine Sorgen machen, wenn man seinen Rechner mit anderen Windowsnutzern teilt, die ebenfalls das Subsystem nutzen.

Achtung: Wenn ein Windows Administrator Zugriff auf die Daten eines Nutzers hat, so hat er auch Zugriff auf dessen Subsystem und kann dort auch Dateien manipulieren und einsehen.

Meine Dateien aus dem WSL nutzen

Eigentlich würde man erwarten, dass man mit `/home/Benutzername` bzw. `~` in das Windowsverzeichnis `%userprofile%` kommt. Dies ist allerdings nicht der Fall, der Grund dafür wird später gezeigt. Wie kommt man dann in den Windows Benutzerordner? Über das `mnt` Verzeichnis! Das Windowsystem ist in das Subsystem über das `/mnt/` Verzeichnis eingemountet.

```

timbo@NBRV0PX1:~/mnt/c/Users/tborowski$ touch permissions
timbo@NBRV0PX1:~/mnt/c/Users/tborowski$ ls -l | grep permissions
-rwxrwxrwx 1 root root 0 Aug 21 17:46 permissions
timbo@NBRV0PX1:~/mnt/c/Users/tborowski$

```

Abbildung 6: Datei erstellen und Rechte auslesen

In `/mnt/` sehen wir also alle Dateien, auf die wir als normaler Windows Benutzer auch Zugriff haben. Erstellen wir unsere erste Datei aus dem WSL innerhalb eines „Windows Ordners“. Dazu kann man z.B. folgenden Befehl benutzen:

```
echo „Hello from Subsystem“ >> hello.txt
```

Im entsprechenden Ordner findet sich anschließend eine Datei `hello.txt` mit dem Inhalt „Hello from Subsystem“. Mit `cat hello.txt` lassen sich Dateinhalte anzeigen und mit `nano vim` etc. lassen sich diese direkt in der Shell manipulieren.

Rechteverwaltung von Windowsdaten

Greift man auf Windowsdateien, die unter `/mnt/c` liegen, zu, so werden die „Linux-Rechte“ aus den „NTFS Rechten“ abgeleitet. Folgerichtig wird bei solchen Dateien auch kein `chmod` und andere Rechtemanipulationen unterstützt.

`touch` erzeugt eine Datei, wenn sie nicht existiert. Im obigen Beispiel wurde die Datei `permissions` angelegt, welche auch im Windows Explorer im Homeverzeichnis auffindbar ist. Was bedeutet `-rwxrwxrwx 1 root root`? Die `1` wollen wir an der Stelle ignorieren. `R` bedeutet `read`, `w` bedeutet `write` und `x` bedeutet `execute`. Angeordnet sind sie in 3 Gruppen: Besitzer, Gruppe und alle anderen. Der `-` bedeutet, dass es sich um eine normale Datei handelt. Abschließend kommen der Besitzer und die Gruppe. Folglich haben Dateien, die aus dem WSL in einem „Windows Ordner“ angelegt werden den Besitzer `root` und die Gruppe `root`. Dabei haben sowohl der Besitzer, als auch die Gruppe und auch alle anderen den Zugriff `read`, `write` und `execute`. Selbiges Ergebnis erhält man, wenn man aus Windows eine Datei anlegt.

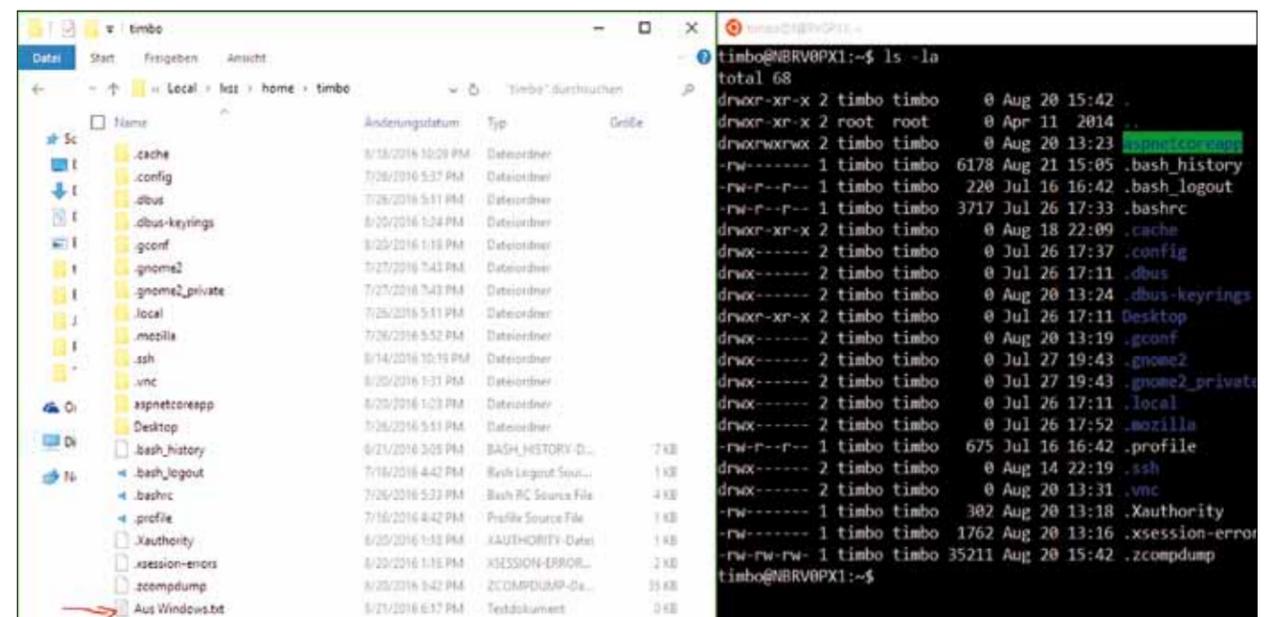


Abbildung 8: Angelegte Datei ist im WSL nicht zu finden

```

timbo@NBRV0PX1:/etc$ ls -la
total 589
drwxr-xr-x 2 root root 0 Aug 20 15:41 .
drwxr-xr-x 2 root root 0 Aug 21 15:06 ..
drwxr-xr-x 2 root root 0 Mar 23 21:44 acpi
-rw-r--r-- 1 root root 2981 Mar 23 21:42 adduser.conf
-rw-r--r-- 1 root root 10 Mar 23 21:45 adjtime
drwxr-xr-x 2 root root 0 Aug 21 12:17 alternatives
-rw-r--r-- 1 root root 112 Jan 10 2014 app.conf
drwxr-xr-x 2 root root 0 Mar 23 21:43 apm
drwxr-xr-x 2 root root 0 Mar 23 21:44 apparmor

```

Abbildung 7: Dateien im WSL unterstützen volles Rechtssystem

Wo liegen die Dateien des WSL

Die Dateien eines WSL von einem Nutzer liegen in `%Local AppData%\lxss\`. Navigieren wir in das Verzeichnis `/etc` und lassen uns die Rechte ausgeben.

Prinzipiell unterstützen die Dateien, die unter `%Local AppData%\lxss\` liegen, die komplette Rechtevergabe! In diesen Ordnern wird auch `chmod` voll unterstützt.

Dafür gibt es einen anderen Nachteil. Die dort liegenden Dateien werden vollständig vom WSL verwaltet. Das bedeutet, dass wenn wir aus Windows in diesen Ordnern Dateien hinzufügen oder schlimmstenfalls beim Laufen des WSL aus Windows dort Dateien löschen, es zu inkonsistenten Zuständen kommen kann. Legen wir eine Datei im Homeverzeichnis aus Windows an und schauen uns die Ausgabe aus dem WSL an.

Die Datei „Aus Windows.txt“ wird nicht im WSL angezeigt. Als Faustregel gilt: Manipuliere niemals Dateien aus Windows in diesen Ordnern! Microsoft arbeitet aktiv daran, die Interoperabilität zu erhöhen.

Grafische Ausgabe

Besteht die Welt von Linux nur aus Terminals und Shells? Keineswegs! Deswegen schauen wir uns an, wie wir ein Linux-Programm mit grafischer Ausgabe auf unserem Windowsrechner zum Laufen bekommen.

Linux setzt bei der Grafikausgabe auf ein Server-Client Prinzip. Das bedeutet, dass Programme, welche eine

```

timbo@NBRV0PX1: ~
timbo@NBRV0PX1:~$ firefox
Sandbox: unexpected multithreading found; this prevents using namespace sandboxing.
error: XDG_RUNTIME_DIR not set in the environment.
Error: cannot open display: :0.0
timbo@NBRV0PX1:~$

```

Abbildung 9: Starten von Firefox ohne X-Server scheitert

grafische Ausgabe wünschen als Client agieren und dem sogenannten X-Server die Inhalte übermitteln. Bei der klassischen Ubuntu Linux-Distribution ist automatisch ein X-Server [3] installiert. Auf unserem Windows-Rechner ist normalerweise kein X-Server installiert. Wenn wir ein Programm starten, das einen X-Server benötigt, bekommen wir eine Ausgabe wie die folgende:

Error: cannot open display: :0.0. Falls du kein Firefox installiert hast, dann installiere ihn mit apt-get und dem apt-get Befehl. `sudo apt-get install firefox`

Was uns fehlt ist ein X-Server auf einem Windows Rechner. Eine einfache, empfehlenswerte Möglichkeit ist Xming [4]. Daneben gibt es eine Menge Alternativen wie Cygwin/X oder MobaXterm.

Was wir auf dem Bild sehen, ist das gnome-control-center, firefox und den Dateexplorer Nautilus. Tipp: Starte Programme aus der bash mit einem abschließenden &, damit die Konsole für den neuen Prozess nicht blockiert wird. Zur Erinnerung: Das was wir hier sehen sind Programme, die für Linux kompiliert wurden und nativ in einem Windowsbetriebssystem ausgeführt werden!

Installieren von Mono

Nun gehen wir in die andere Richtung. Wir schreiben ein Programm in Visual Studio für Windows mit Windows Forms. Anschließend starten wir das kompilierte Programm unter Windows. Zuletzt wird das Programm mittels Mono im WSL gestartet. Installiert wird mono wie folgt:

Fügt den Public-Key einer neuen Paketquelle in die Schlüsselliste der Paketverwaltung hinzu:

```

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA-6A19B38D3D831EF

```

Schreibt den Downloadstring in die Datei mono-xamarin.list:

```

echo „deb http://download.mono-project.com/repo/debian wheezy main“ | sudo tee /etc/apt/sources.list.d/mono-xamarin.list

```

Updatet alle Paketquellen:

```
sudo apt-get update
```

Installiert Mono vollständig:

```
sudo apt-get install mono-complete
```

Wir haben nun Mono für Linux installiert. Dies sind Binaries, welche die .NET Umgebung für Linux bereitstellen. Diese Mono Binaries sind für Linux kompiliert worden. Wir nutzen diese Libraries, um ein für Windows geschriebenes Programm in einem Pico-Prozess auf einem Windowsrechner auszuführen.

Windows Forms Projekt mit Mono

Wenn Mono installiert ist, navigieren wir mit dem Dateexplorer Nautilus in den Pfad, in welchen wir das Windows-Forms Projekt gebaut haben und führe es mit gewohntem Doppelklick aus. Über /mnt/c/ gelangt man an die Windows Dateien.

Von oben nach unten sieht man den Visual Studio Designer, das ausgeführte Programm unter Windows und abschließend das mit Mono ausgeführte Programm im WSL.

ASP.NET Core

Nun wollen wir ASP.NET Core für Linux in unserem WSL benutzen. Auch wenn wir das Programm bzw. den Webservice später auf einem Linux ausführen wollen, hindert uns nichts daran, dies im gewohnten Visual Studio zu entwickeln. .NET-Core hat erstmal nichts mit Mono zu tun. Das Framework .NET-Core ist eine von Microsoft

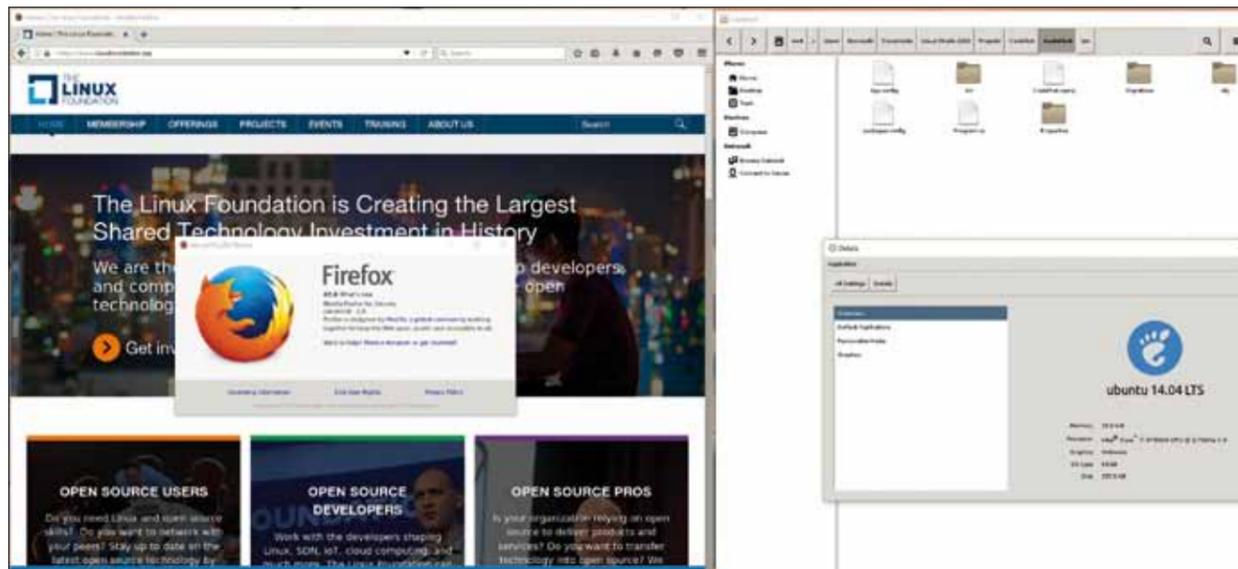


Abbildung 10: Firefox, Nautilus und gnome-control-center mit grafischer Benutzeroberfläche

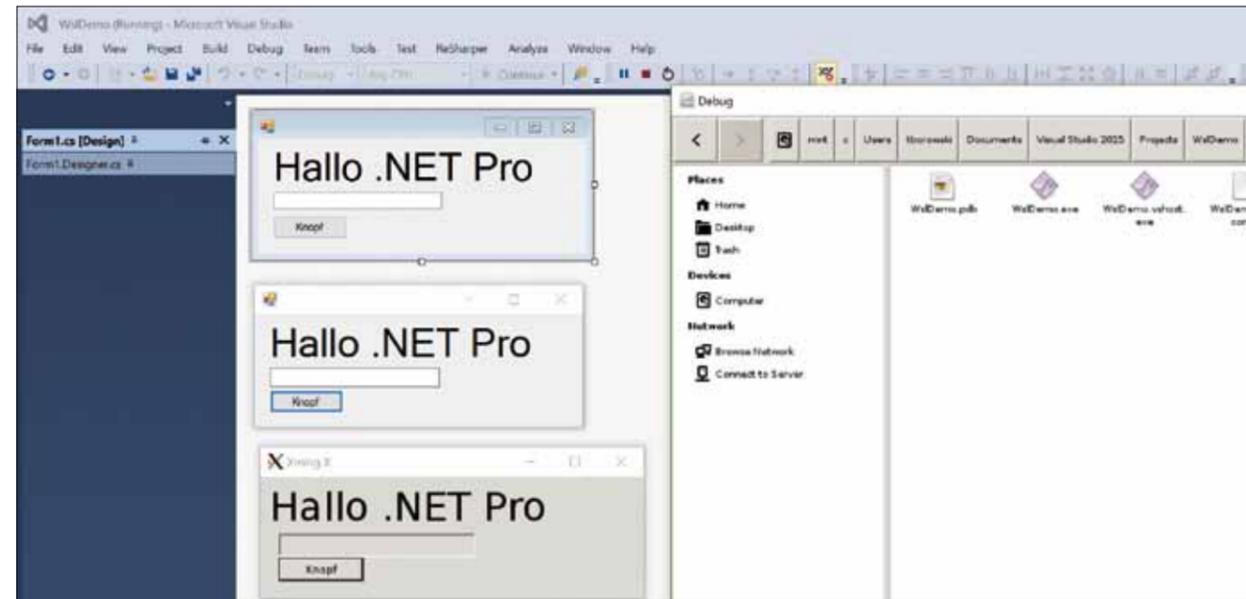


Abbildung 11: .NET Anwendung im Visual-Studio Designer, Programm unter Windows und im WSL

zur Verfügung gestellte, Open-Source Implementierung für eine Teilmenge des .NET-Frameworks. Dabei ist das .NET-Core speziell dafür ausgelegt, dass es auch unter Linux lauffähig ist. Wir erstellen also ein Projekt mit Visual C# und wählen „.NET Core“ aus. Abschließend wählen wir „ASP.NET Core Web Application (.NET Core)“ aus. Im darauffolgenden Fenster wählen wir „Web-API“ aus.

Nun müssen die NuGet-Pakete heruntergeladen werden. Dazu klicke mit der rechten Maustaste auf die Projektmappe und wähle „Restore Packages“. Die Binaries für Windows werden heruntergeladen und danach lässt sich das Programm bereits aus dem Visual Studio starten.

Unter der Adresse `http://localhost:5000/api/values` begrüßt uns der ValuesController und gibt [„value1“, „value2“] aus.

Nun beenden wir den Server und wollen das Projekt mit .NET Core für Linux ausführen.

ASP.NET Core Installation im WSL

Zuerst fügen wir der Datei dotnetdev.list den String `deb [arch=amd64] https://apt-mo.trafficmanager.net/repos/dotnet-release/ trusty main` hinzu:

```

sudo sh -c „echo „deb [arch=amd64] https://apt-mo.trafficmanager.net/repos/dotnet-release/ trusty main“ > /etc/apt/sources.list.d/dotnetdev.list“

```

Wie bei der Monoinstallation, muss der Public-Key für die neue Paketquelle hinzugefügt werden:

```

sudo apt-key adv --keyserver apt-mo.trafficmanager.net --recv-keys 417A0893

```

Updaten der Repositories:

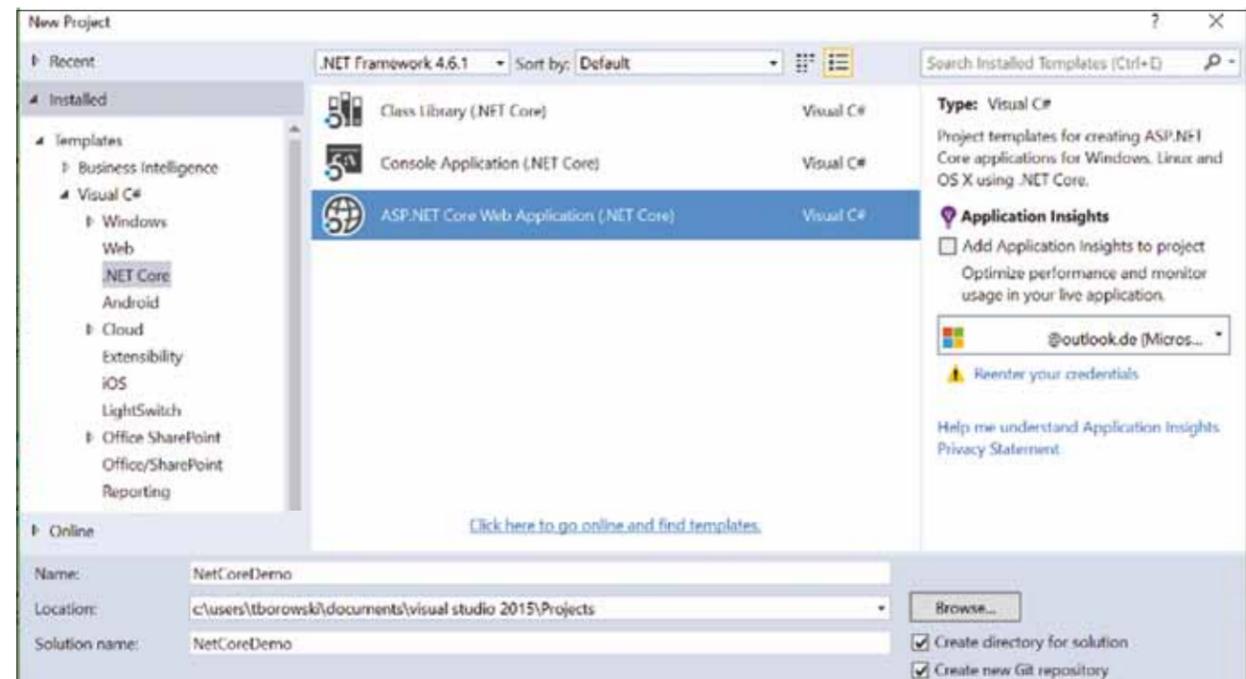


Abbildung 12: Anlegen eines ASP.NET Core Web Application (.NET Core) Projektes

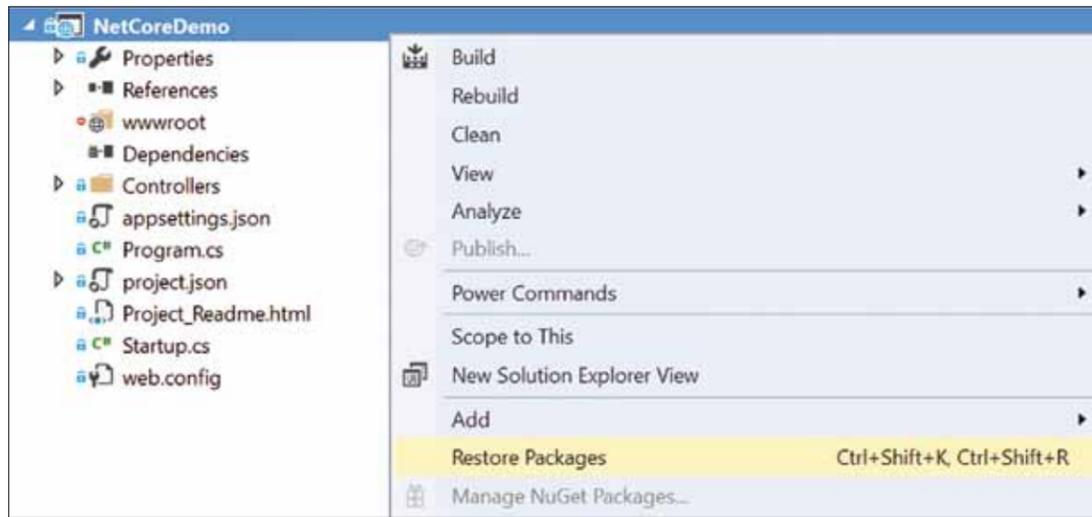


Abbildung 13:
NuGet Pakete
herunterladen

`sudo apt-get update`

Jetzt wurden die Quellen für .NET -Core in DPKG eingetragen. Daraufhin kann man diese mit DPKG einfach installieren, indem man `sudo apt-get install dotnet-dev-1.0.0-preview2-003121` ausführt. Tipp: Es wird in der Regel immer empfohlen, Pakete über die Paketverwaltung hinzuzufügen. Mit der Paketverwaltung, ähnlich wie bei Android, werden Programme automatisch auf dem neuesten Stand gehalten und Konflikte zwischen installierten Programmen können so erkannt werden.

Ausführen von ASP.NET Core im WSL

Nun haben wir im Windows Subsystem .NET Core installiert. Jetzt wollen wir unsere Webapplikation in diesem laufen lassen. Dazu müssen wir nichts am Quellcode ändern und auch keinen IIS konfigurieren oder sonstige Änderungen an irgendwelchen Konfigurationsdateien vornehmen! Folgendermaßen kann man einen ASP.NET Service auf .NET-Core Basis starten.

Zunächst navigiert man mit der Bash in den Projektordner, in welchem sich die .sln Datei befindet. Danach navigiert man in den src/ Ordner und in den Projektordner. In diesem befinden sich die project.json und die

Source-Files der Controller. Hier können wir, nachdem wir die Dependencies dafür installiert haben, den Web-Service starten. Mit `dotnet restore` werden die notwendigen Abhängigkeiten installiert. Die Abhängigkeiten sind in der project.json definiert und werden automatisch über das .NET-Core aufgelöst und installiert. Jetzt können wir den Server starten.

`ASPNETCORE_URLS="https://*:5000" dotnet run`

Wie erwartet startet der ASP.NET Core Service im WSL. Verbinden wir uns mit dem Service über die Adresse `http://localhost:5000/apo/values`, sehen wir in der Bash, wie unsere Anfrage an den ValuesController verarbeitet wird.

VERWEISE

- [1] System Call Table: <https://filippo.io/linux-syscall-table/>
- [2] Aufbau vom WSL: <https://blogs.msdn.microsoft.com/wsl/>
- [3] X-Server https://de.wikipedia.org/wiki/X_Window_System
- [3] X-Server für Windows <http://www.straightrunning.com/XmingNotes/>

Warum brauchen wir einen .NET Standard?

Dieser Artikel ist eine Zusammenfassung und Übersetzung eines Blog Eintrages von Immo Landwerth, Group Program Manager Microsoft

.NET hat sich in den über 15 Jahren seiner Entwicklung in mehrer Zweige gespalten. Auf der einen Seite ist dies eigentlich eine wirklich gute Sache. Beispielsweise wurde das .NET Compact Framework geschaffen, um zur restriktiven Hardware des Handys in der 2000-Ära zu passen. Das gleiche gilt heute: Unity als Zweig von Mono kann auf mehr als 20 Plattformen ausgeführt werden.

Aber auf der anderen Seite entsteht ein massives Problem für Entwickler Code für mehrere .NET Plattformen zu schreiben, weil es keine einheitliche Klassenbibliothek gibt. (s. Abbildung 1)

Es gibt derzeit drei relevante Varianten von .NET mit drei verschiedenen Basisklassenbibliotheken. Ein Ende ist nicht absehbar, da neue Betriebssysteme unterstützt werden oder es für bestimmte Gerätefunktionen angepasst wird.

Um diesem .NET Wildwuchs Herr zu werden kommt .NET Standard ins Spiel. (s. Abbildung 2)

Der Entwickler muss nur eine Basisklassenbibliothek kennen, die auf allen .NET Plattformen ausgeführt werden kann. Auch der Anbieter von Erweiterungen muss sich nicht um verschiedene API Levels kümmern um auf NuGet sein Paket zu platzieren.

Zu aller erst stellt .NET Standard sicher, dass alle .NET Plattformen die gleiche API für die Basisklassenbibliothek teilen. Egal ob man eine Desktop-Anwendung programmiert, mobile App oder den Cloud-Dienst verwendet. Zweitens werden mit .NET Standard die meisten Klassenbibliotheken global verfügbar. Die Basis ist innerhalb des .NET Bibliothek

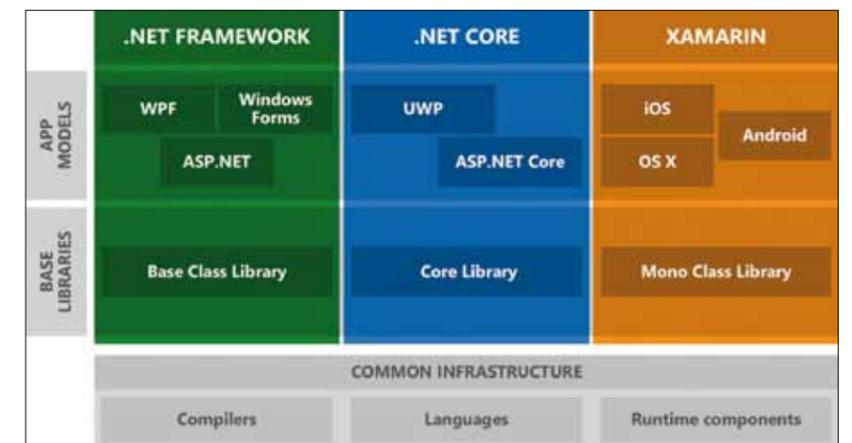


Abbildung 1

Ökosystems konsistent. Dabei unterscheidet sich der technologische Ansatz von den seit längerem genutzten PCL (Portable Class Library) erheblich. Während das Werkzeug Ihnen hilft, Binärdateien zu produzieren, die auf mehreren Plattformen arbeiten, zwingt es die verschiedenen APIs der Basisklassenbibliotheken zu berücksichtigen. Ein weiterer wichtiger Aspekt von .NET Standard ist die

Abwärtskompatibilität der API. Höhere Version bietet schlicht zusätzliche APIs. Mit PCLs ist das nicht unbedingt der Fall: die Menge der verfügbaren APIs ist das Ergebnis der Schnittmenge zwischen den ausgewählten Plattformen. Vergleicht man .NET Framework und .NET Core Xamarin/Mono, wird man feststellen, dass .NET Core die API weniger umfangreich ist (mit Ausnahme von OS-spezifische APIs). Es

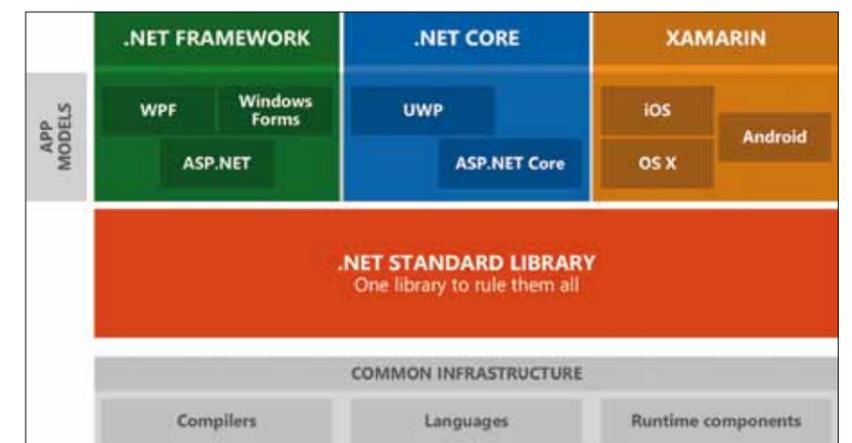


Abbildung 2

kommt zu drastischen Unterschieden in der Verfügbarkeit von grundlegenden APIs (z. B. Netzwerk- und Kryptografie-APIs). Das zweite Problem mit .NET Core sind grundlegende Änderungen wie Reflection. Damit ist es aufwändig .NET zu .NET Core zu portieren. .NET Standard vereinheitlicht dies auch plattformübergreifend.

Dies stellt auch den Autor eines NuGet Paketes vor Probleme. Das Ziel ist es, die .NET Plattform in kleinere NuGet Pakete aufzuteilen. Das funktioniert recht gut, wenn all diese Komponenten explizit mit der Anwendung bereitgestellt werden können, weil man diese getrennt aktualisieren kann. Wenn das Ziel eine abstrakte Spezifikation, z. B. PCLs oder .NET Standard, ist funktioniert jedoch diese Geschichte weniger gut. Die spezifische Kombination der anhängigen DLL Versionen, die für jeden Satz von Plattformen ausgeführt werden können muss. .NET Standard löst all dies, da es nur den Satz von erforderlichen APIs darstellt, gibt es keine Notwendigkeit, es noch weiter zu unterteilen. Das einzige wichtige ist die Version, welche wirkt wie eine API-Ebene: Je höher die Version desto mehr API Funktionen.

WAS GIBT ES NEUES IN .NET STANDARD 2.0?

Zeitgleich mit .NET Core 1.0 wurde .NET Standard eingeführt. Es gibt mehrere Versionen des .NET Standard, die die API-Verfügbarkeit auf allen aktuellen Plattformen definieren. (s. Tabelle 1)

.NET Platform	.NET Standard							
	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	→	→	→	→	→	→	1.0	vNext
.NET Framework	→	4.5	4.5.1	4.6	4.6.1	4.6.2	vNext	4.6.1
Xamarin.iOS	→	→	→	→	→	→	→	vNext
Xamarin.Android	→	→	→	→	→	→	→	vNext
Universal Windows Platform	→	→	→	→	10.0	→	→	vNext
Windows	→	8.0	8.1					
Windows Phone	→	→	8.1					
Windows Phone Silverlight	8.0							

Tabelle 1

Die Pfeile zeigen, dass die Plattform eine höhere Version von .NET Standard unterstützt. Zum Beispiel unterstützt .NET Core 1.0 .NET Standard Version 1.6. Wenn eine Anwendung auf .NET Framework 4.5 und .NET Core 1.0 laufen soll, kann man sich an .NET Standard 1.1 ausrichten.

.NET Framework 4.6.1 implementiert bereits alle APIs, die Teil des .NET Standard 2.0 sind.

Wenn man eine Bibliothek für .NET Standard entwickelt, kann man auf zwei Arten von anderen Bibliotheken verweisen.

.NET Standard, wenn die Version kleiner oder gleich der Version ist. Portable Klassenbibliotheken, wenn deren Profil auf eine .NET Standard Profil zugeordnet werden kann und diese Version niedriger oder gleich der Version die im Projekt zum Einsatz kommt. (s. Abbildung 3)

PCLs und .NET Standard ist auf NuGet faktisch wenig verbreitet. Das .NET Framework dominiert bei weitem. (s. Tabelle 2)

In .NET Standard 2.0 wird es auch möglich, dass .NET Standard Bibliotheken existierende .NET Assembly zu referenzieren über einen Kompatibilitäts Layer, in der Informatik als Shim bezeichnet. (s. Abbildung 4)

Natürlich funktioniert dies nur für jene Fälle, in denen .NET Framework-Bibliothek APIs verwendet, die für .NET Standard zur Verfügung stehen. Darum ist das nicht die bevorzugte Methode für den Bau von Bibliotheken, die über verschiedene Plattformen hinweg .NET verwenden möchten. Dieser Kompatibilität Shim dient jedoch als Brücke, um existierende

Target	Occurrences
.NET Framework	46,894
.NET Standard	1,886
Portable	4,501

Tabelle 2

	1.4	1.5	1.6	2.0
.NET Framework	4.6.1	4.6.2	vNext	4.6.1

Tabelle 3

Bibliotheken in .NET Standard weiter zu verwenden ohne diese neu zu schreiben oder weg zu werfen.

.NET Framework 4.6.1 verfügt über die höchste Verbreitung. Daher möchte Microsoft sicherstellen, dass es .NET Standard 2.0 implementieren kann.

.NET Core eine viel kleinere API-Satz als .NET Framework oder Xamarin. Die Unterstützung von .NET Standard 2.0 bedeutet, dass die verfügbaren Funktionen deutlich ausgebaut werden müssen. Da .NET Core nicht mit dem OS aber mit der App installiert wird, benötigt das .NET Standard 2.0 Unterstützung nur für Updates des SDK und der NuGet-Pakete.

Xamarin unterstützt bereits die meisten APIs, die Teil des .NET Standard sind. Aktualisierung funktioniert mit wachsender Tendenz ähnlich wie .NET Core. In der Tat wurden die meisten von ihnen bereits den stabilen Zyklus 8 Release/Mono 4.6.0 hinzugefügt.

Somit erklärt sich auch, welche Versionen von .NET Framework die Version von .NET Standard unterstützen. (s. Tabelle 3)

Die Auswirkung dieser Design-Änderung war überraschend gering. Die Analyse aller Pakete auf NuGet.org, mit .NET Standard 1.5 oder höher ergab, dass nur sechs nicht-Microsoft Pakete diese nutzen. Mit diesen wurde direkt eine Lösung gesucht und gefunden um Anrufe mit APIs zu ersetzen, die mit .NET Standard 2.0 kommen.

Auf dem Weg zum .NET Standard wurden zunächst alle APIs berücksichtigt, die in beiden .NET Framework und Xamarin verfügbar sind. Erforderliche APIs sind Funktionen die alle Plattformen bieten und für Cross-Plattform Lösungen als nötig

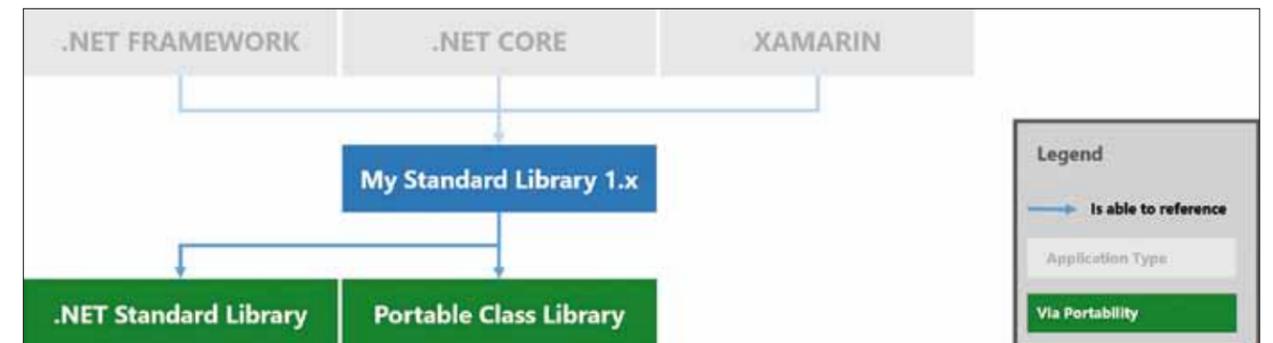


Abbildung 3

Übliche Versionierung hätte zur Folge, dass .NET Standard 2.0 nur durch eine neuere Version von .NET Framework unterstützt werden würde, angesichts der Tatsache, dass die neueste Version von .NET Framework (4.6.2) nur .NET Standard 1.5 unterstützt. Dies würde bedeuten, dass die Bibliotheken kompiliert gegen .NET Standard 2.0 nicht auf der überwiegenden Mehrheit der .NET Framework Installationen laufen würde.

Um .NET Framework 4.6.1 .NET Standard 2.0 unterstützen zu können, mussten alle APIs von .NET Standard, die in .NET Standard 1,5 und 1,6 eingeführt wurden wieder entfernt werden.

Die Auswirkung dieser Design-Änderung war überraschend gering. Die Analyse aller Pakete auf NuGet.org, mit .NET Standard 1.5 oder höher ergab, dass nur sechs nicht-Microsoft Pakete diese nutzen. Mit diesen wurde direkt eine Lösung gesucht und gefunden um Anrufe mit APIs zu ersetzen, die mit .NET Standard 2.0 kommen.

Auf dem Weg zum .NET Standard wurden zunächst alle APIs berücksichtigt, die in beiden .NET Framework und Xamarin verfügbar sind. Erforderliche APIs sind Funktionen die alle Plattformen bieten und für Cross-Plattform Lösungen als nötig

betrachtet werden. Optionale APIs, sind Plattform-spezifischen oder sind Teil der älteren Technologien.

Optionale APIs sind nicht Teil des .NET Standard, aber stehen als separate NuGet-Pakete zur Verfügung. Es wird versucht, diese als Bibliotheken mit .NET Standard zu adressieren, so dass deren Umsetzung von jeder Plattform konsumiert werden kann. Allerdings mit Grenzen wie die Zugriff auf die Windows Registry.

Um einige APIs optional zu machen mussten andere APIs entfernt werden, die zum erforderlichen API-Satz gehören. Zum Beispiel gibt es in .NET Code Access Security, CAS ist eine ältere Komponente. Dies macht es nötig, alle Mitglieder aus diese Typen nutzen, die Teil des CAS, entfernen, wie z.B. Überladungen, akzeptieren.

Die aktuelle Dokumentation von .NET Standard 2.0 .NET befindet sich in einem GitHub Repository und ist nicht final.

Eine der größten Herausforderungen für Multi-Plattform-Klassenbibliotheken ist es zu vermeiden, dass man sich auf den kleinsten gemeinsamen Nenner reduziert. Gleichzeitig läuft man Gefahr versehentlich Bibliotheken zu erstellen, die in der Praxis wesentlich weniger Portabel sind als beabsichtigt.

In PCLs wurde das Problem mit

mehreren Profilen, die jeweils die Schnittpunkte mit den Plattformen darstellen, gelöst. Damit lässt sich die maximale API für jede Plattform nutzen. Der .NET Standard definiert den Satz von APIs, die alle .NET Plattformen implementieren müssen.

Das wirft allerdings die Frage auf, wie modelliert man APIs, die auf allen Plattformen implementiert werden können? APIs mit der Fähigkeit Funktionen per Reflection zur Laufzeit zu erstellen. Dies kann nicht auf .NET Plattformen funktionieren, die nicht über einen JIT-Compiler, wie .NET Native UWP oder über Xamarin iOS verfügen. Betriebssystem-spezifische APIs wie Win32 sollen leicht konsumierbar sein. Ein gutes Beispiel ist der Windows-Registrierung. Die Umsetzung hängt an der zugrundeliegenden Win32, die keine Entsprechungen auf anderen Betriebssystemen hat.

Man kann auf solche API verzichten und sie einfach nicht implementieren und damit auf den Komfort der Zielplattform verzichten. Man kann sie aber auch implementieren, aber im Falle des Falles eine PlatformNotSupportedException werfen. Dies würde bedeuten, dass der größtmögliche Satz an APIs unabhängig davon, ob sie überall oder nicht unterstützt werden, vorhanden ist.

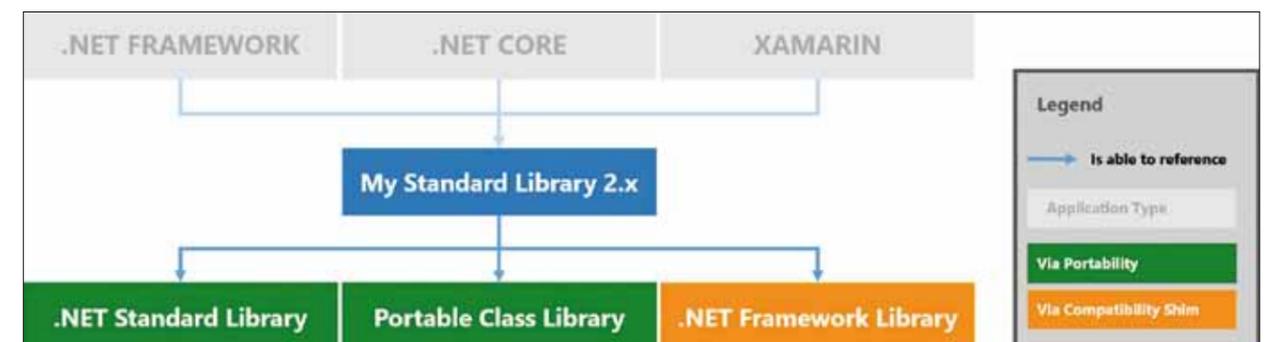


Abbildung 4

Der dritte Weg ist es die API zu emulieren. Mono nutzt die Registry als eine API über .INI Dateien. Natürlich kann man so keine Betriebssystem Infos auslesen, es funktioniert aber recht gut für die Fälle, wo die Anwendung die Registrierung einfach verwendet, um einen eigenen Status und Benutzer-Einstellungen zu speichern.

Microsoft kombiniert einfach alle drei Methoden. Es wird der Satz von APIs definiert, die alle .NET Plattformen implementieren müssen oder sollten, so dass plattformübergreifende Bibliotheken schreiben einfach und intuitiv ist.

Die allgemeine Strategie für den Umgang mit Technologien, die nur auf einigen .NET Plattformen zur Verfügung stehen sind explizit NuGet-Pakete, für .NET Standard. Es wird ein extra NuGet-Paket erstellt für die zusätzliche Funktion eines Betriebssystem-spezifischen API-Aufrufs.

Diese Strategie funktioniert gut für in sich abgeschlossen APIs und kann somit in ein separates Paket verschoben werden. Für Fälle, wo einzelne Mitglieder Typen verwenden die nicht seperierbar sind, verwendet Microsoft den zweiten und dritten Ansatz: Plattformen haben diese APIs, aber sie können entscheiden, eine Exception zu werfen oder zu emulieren.

Die Windows-Registry ist so eine eigenständige Komponente, die als separates NuGet Paket bereitgestellt wird (Microsoft.Win32.Registry). So kann man aus .NET Core diese nutzen, aber eben nur auf Geräten die Windows ausführen. Entweder man sorgt dafür dass die Anwendung immer auf Windows ausgeführt wird, oder der Code muss angemessen reagieren. Microsoft kündigt aber Werkzeuge an, um bei der Erkennung von diesen Fällen zu helfen.

Die Nutzung von AppDomains ist in modularen LOB Anwendungen weit verbreitet. Es gibt APIs, die zur Schaffung von App-Domänen, z. B. Abrufen der Liste der geladenen Assemblys. Deswegen ist der Großteil Bestandteil von .NET Standard. Einige APIs z. B. die sich um die Erstellung derselben kümmern werfen auf nicht unterstützten Plattformen Exceptions, wie .NET Core. Reflection ausgelöster Code ist relativ

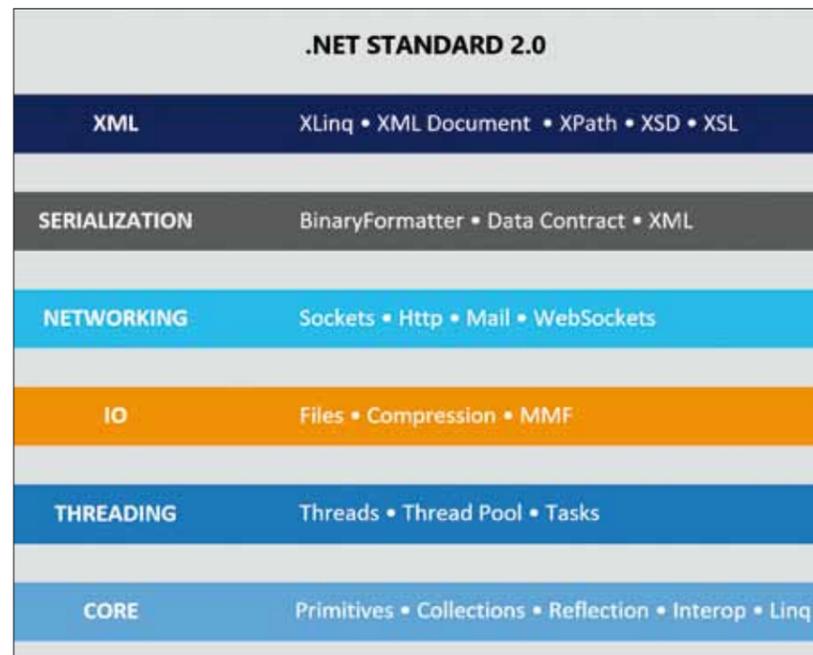


Abbildung 5

eigenständig und so ist der Plan analog zum Registry-Zugriff vorzugehen. Es gibt andere APIs, die logisch auf die Möglichkeit, Code später zu kompilieren, zugehören z. B. die Expression Tree-Methode oder die Möglichkeit, reguläre Ausdrücke zu kompilieren. In einigen Fällen wird ihr Verhalten emuliert, während in anderen Fällen wie bei regular Expression Kompilierung eine Exception geworfen wird.

Im Ergebnis werden Konzepte vorhanden sein, die möglicherweise nicht auf allen .NET Plattformen verfügbar sind. Spezialfunktionen werden in der Regel auf explizit zu verweisende einzelne Pakete ausgelagert. In seltenen Fällen können einzelne Aufrufe Ausnahmen auslösen.

.NET Core wurde entworfen um mit seinen Referenz Assemblies bestmöglich in die .NET Portability Strategie zu passen. Dies erschwert aber auch das Hinzufügen von neuen APIs. Man muss jedes Mal entscheiden wo diese APIs überall zur Verfügung stehen. Schlimmer noch, aufgrund der Regeln zur Versionierung bedeutet das auch, dass entschieden werden muss, welche Kombination von APIs, in welcher Reihenfolge zur Verfügung gestellt werden.

Neue Erweiterungen setzen auf bestehenden APIs auf („out-of-Band“). Das ist der bevorzugte Weg für Technologien, wo dies problemlos möglich ist, denn es bedeutet auch

.NET Entwickler testen mit den APIs und geben Feedback. Dies wurde für immutable Collections mit großem Erfolg getan.

Für Features, welche die eine Runtime wie die Common Language erfordern, ist dies viel schwieriger. Man kann nicht einfach alles in ein NuGet-Paket stecken.

Das ist schwieriger auf Plattformen, die eine System Runtime wie das .NET Framework haben, jedoch auch schwieriger im Allgemeinen, weil mehrere Versionen der Runtime für verschiedene Zwecke (z.B. JIT Vs AOT) im Einsatz sind. Innovativ können diese in der Praxis über das gesamte Einsatzspektrum nicht gleichzeitig ausgerollt werden. Das Schöne an .NET Core ist, dass diese Plattform komplett eigenständig sein soll und sich auch mit der App ausliefern lässt. In der Zukunft wird man Neuerungen damit zuerst in .NET Core sehen, weil es am wenigsten Einfluss auf andere Komponenten hat.

Um .NET Core unabhängig von anderen .NET Plattformen entwickeln zu können wurden die Portabilität-Mechanismen von .NET Core getrennt. .NET Standard ist definiert als eine unabhängige Referenz-Assembly, die von allen .NET Plattformen erfüllt ist. Jede der .NET Plattformen verwendet einen anderen Satz von Referenzassemblys und man kann somit neue APIs in was auch immer frei

hinzuzufügen. Microsoft kann dann basierend auf Nutzung, entscheiden ob diese API dem .NET Standard hinzugefügt werden und damit universell verfügbar sind.

Portabilität von .NET Core zu trennen, hilft die Entwicklung von .NET Core zu beschleunigen und Experimente mit neuen Features zu vereinfachen. Anstatt künstlich API Design auf bestehenden Plattformen zu stützen, können wir einfach die Schicht verändern, die es betrifft, um die Funktion zu unterstützen.

Das Hinzufügen von neuen APIs in .NET Core ist kein Selbstzweck, um sie in den .NET Standard zu überführen. Das Ziel für .NET Standard ist Konsistenz zwischen den .NET Plattformen zu erstellen. Also neue Mitglieder, die bereits Bestandteil der Norm sind, werden automatisch berücksichtigt, wenn der Standard aktualisiert wird.

Als Autor einer Bibliothek sollte man auf .NET Standard umstellen, weil es die plattformspezifischen PCL ersetzt. Im Falle von .NET Standard ist 1.x die Menge der verfügbaren APIs PCLs sehr ähnlich. Aber .NET Standard 2.x hat eine deutlich größere API festgelegt und ermöglicht es gezielt .NET Framework Bibliotheken zu referenzieren.

Eine Herausforderung mit PCLs ist, dass zwar mehrere Zielplattformen erreicht werden können, es aber am Ende auf ein bestimmtes .NET Profile abzielt. Dies gilt insbesondere für NuGet-Pakete, weil in den Lib Ordnern alle unterstützten Plattformen Namen aufgelistet werden müssen. Ein Beispiel portable-win+net45+wp80+win81+wp81+MonoAndroid10+MonoTouch10+Xamarin.iOS10. Dies sieht nicht nur seltsam aus, sondern verursacht Probleme, wenn neue Plattformen die Unterstützung der gleichen APIs zeigen sollen. .NET Standard vermeidet dieses Problem, weil sie eine Version des Standards zu Zielen, die Plattforminformationen, z.B. nicht enthalten ist. Als Beispiel eines Ordner Namens netstandard1.5.

PCLs unterstützt derzeit eine breitere Palette von Plattformen und nicht alle Profile haben eine entsprechende .NET Standard Version.

PCLs verbietet Abhängigkeiten von APIs und Bibliotheken,

die die ausgewählten Plattformen nicht ausgeführt werden können. So werden PCL-Projekte nur auf andere PCLs verweisen, die eine Obermenge der Plattformen abzielen, auf die Ihre PCL adressiert. .NET Standard ist ähnlich, aber es ermöglicht zusätzlich Verweise auf .NET Framework Binaries, die de-facto-Währung im Bibliothek Ökosystem sind. Also, mit .NET Standard 2.0 erhält man Zugriff auf eine viel größere Anzahl von Bibliotheken.

Mit dem Kommando Zeilen Werkzeug Apiport kann man sehen, wie kompatibel die Codebasis mit den verschiedenen Versionen von .NET Standard ist. Man findet den Download auf github.com/Microsoft/dotnet-apiport.

Weiterhin gibt die .NET Standard-Dokumentation Einblick in die adressierten Plattformen.

Möchte man beispielsweise wissen ob man .NET Standard 2.0, .NET Standard 1.6 verwenden soll kann man gegen beide prüfen.

```
> apiport analyze -f C:\src\mylibs\ -t „.NET Standard,Version=1.6“
Standard,Version=2.0“
```

ZUSAMMENFASSUNG

Mit .NET Standard erstellte Pakete erlauben eine einfachere und gemeinsame Nutzung sowie die Wiederverwendung von Code zwischen mehreren .NET Plattformen. Mit .NET Standard 2.0 steht Kompatibilität im Vordergrund. Zur Unterstützung der .NET Standard 2.0 in .NET Core und UWP wird die Plattformen um vieles APIs erweitert. Dazu gehört auch einen Kompatibilität Shim, der erlaubt Binaries zu referenzieren, die mit dem .NET Framework kompiliert wurden.

Mit Blick in die Zukunft empfiehlt sich die Verwendung von .NET Standard statt Portable Klassenbibliotheken. Die Werkzeuge für .NET Standard 2.0 entstehen im gleichen Zeitraum wie die bevorstehende Veröffentlichung von Visual Studio, Codename „Dev 15“.

Besseres Arbeitsklima durch Verwendung neuer Rhetorik

- Ist mir Scheiß egal!**
 Ich sehe keinen Grund zur Besorgnis.
- Was habe ich mit dem Scheiß zu tun?**
 Ich war nicht von Anfang an in diesem Projekt involviert.
- Das mach ich sicher nicht du Blödmann!**
 Es gibt technische Gründe, die mir die Erledigung dieser Aufgabe unmöglich machen.

- Verdammt Scheiße, diese Vollidioten haben mir nichts gesagt!**
 Wir müssen unsere interne Kommunikation verbessern.
- Dieser Trottel versteht überhaupt nichts!**
 Er ist mit dem Problem nicht vertraut!
- Mir doch Wurst, du Depp!**
 Bedauerlicherweise kann ich Ihnen in diesem Punkt nicht weiterhelfen.



WINDOWS SERVER CONTAINER die nächste Ära der Virtualisierung

Autor: STEFAN OBER

Windows Server Container sind extrem praktisch wenn es darum geht Ressourcen zu sparen und so viele Anwendungen wie möglich zu virtualisieren. Doch was sind Container eigentlich?

Die Grundidee der Container stammt von dem für Linux Systeme gedachten System Docker. Docker ist eine offene Plattform für verteilte Anwendungen für Entwickler und System-Administratoren.

Stellen wir als erstes Mal den Unterschied zwischen Server Containern und normaler Virtualisierung dar. Bei normaler Virtualisierung über Hyper-V oder auch VmWare hat jede virtuelle Maschine ein komplettes Betriebssystem, welches Ressourcen wie RAM und Plattenspeicher verbraucht. (Abbildung 1)

Bei Windows Server Containern teilen sich die einzelnen Container/Anwendungen allerdings die gleichen Betriebssystemdateien. Erst wenn eine Änderung der Dateien erfolgen soll, werden diese Änderungen mit dem effizienten "copy-on-write" Vorgang von Docker in den Container geschrieben. (Abbildung 2)

Jeder neue Container baut auf einem neuen frischen Betriebssystem auf. Was es möglich macht auch Anwendungen die sich normalerweise

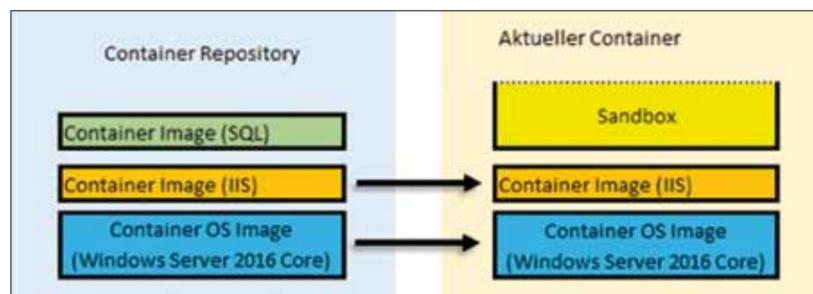


Abbildung 3: Bausteine

untereinander nicht vertragen sozusagen auf dem gleichen Server laufen zu lassen, da diese durch eine Namespace-Isolation voneinander getrennt sind und sich daher gegenseitig nicht beeinflussen können. Der weitere große Vorteil an diesem System ist, dass es möglich ist jeden Container in seinen Ressourcen zu beschränken um bei aufwendigen Aufgaben nicht den kompletten Container Host auszulasten. Als Host Betriebssystem/Container Host kann auch der neue NanoServer von der Technical Preview 4 des Server 2016 dienen. Durch dieses extrem schlanke Betriebssystem erhöht sich die Einsparung der

Ressourcen nochmal merklich da der NanoServer nach eigenen Erfahrungen nach der Installation weniger als 2 GB an Festplattenspeicher und auch weniger als 512 MB Ram (ca. 174 mb mit installierten IIS) belegt und somit ideal dafür geschaffen ist.

Besonders wichtig für uns sind drei Bausteine:

- **Container OS Image:** Das "Betriebssystem", das der Container darstellt und welches nicht verändert werden kann und soll.
- **Container Image:** Ehemalige Container die wir uns als "Vorlage" erstellt haben und mehrfach benutzen können.
- **Sandbox:** Hier landen alle Änderungen im aktuell laufenden Container da die Container Images vom laufenden Container nicht verändert werden können.

Sinnbildlich gesehen sieht das Ganze dann wie in Abbildung 3 gezeigt aus.



Abbildung 1: Virtualisierung ohne Container



Abbildung 2: Virtualisierung mit Container

Listing 1: Skriptdownload

```
wget - uri https://aka.ms/tp4/Install-ContainerHist -
OutFile C:\Install-ContainerHost.ps1
```

Listing 2: Neuer Container

```
New-Container -Name IIS-Host -ContainerImageName
WindowsServerCore -SwitchName „Virtual Switch“
```

Listing 3: Erstellen einer HTML-Datei

```
"Hello World aus einem Windows Server Container" >
C:\inetpub\wwwroot\index.html
```

Die Container lassen wir mit einem Baukasten System aufeinander aufbauen was einen großen Vorteil für Webentwickler darstellt. Diese könnten sich z.B. einen neuen Container auf Grundlagen vom Container OS Image erstellen und in diesen dann den IIS installieren und grob vorkonfigurieren. Alle Änderungen befinden sich jetzt in der Sandbox des Containers. Aufgrund dieser können wir jetzt ein neues Container Image erstellen um die Installation vom IIS als Vorlage für verschiedene Projekte verwenden zu können. Das heißt wir haben uns einmal die "Mühe" der Konfiguration gemacht und können jetzt die Websites in einen neuen Container legen der als Grundlage das IIS Image und das Container OS Image hat. Diesen neuen Website Container können wir jetzt auch ganz einfach auf andere Server oder Geräte übertragen solange die "unteren" zwei Images auf diesen auch vorhanden sind. Entwickler haben so den Vorteil, dass sie die gleiche Unterlage auf Test-Servern sowie auf dem Produktiv-System haben und so ihre Anwendungen oder in diesem Beispiel Websites sehr leicht transportieren können. Die Container werden standardmäßig über einen Virtuellen Switch vom Typ "Intern" auf die Netzwerkverbindung über NAT verbunden. Wollen wir also Zugriff auf die Website unseres Containers müssen wir zusätzlich noch den Port 80 oder 443 über ein statisches "NAT Mapping" durchleiten. Zusätzlich müssen auch die entsprechenden Ports an der Firewall des Container Hosts geöffnet werden.

DIE INSTALLATION

Wir haben mehrere Möglichkeiten einen Windows Server Container Host zu installieren und zu konfigurieren. Wir können z.B. den Containerhost als virtuelle Maschine auf einem bestehenden Hyper-V Host installieren. Oder aber wir installieren direkt das "physikalische" System als Container Host. Die einfachste Bereitstellung erfolgt über die Skripte die uns Microsoft zur Verfügung stellt. Mit dem Befehl in Listing 1 können wir uns über die Powershell das Skript direkt herunterladen.

Wenn wir anschließend das Skript starten lädt dieses unter anderem das Container OS Image herunter, was

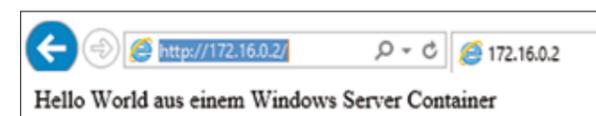


Abbildung 4: Die erste Seite aus dem Container

einige Zeit in Angriff nehmen kann. Was ansonsten noch geladen wird hängt ganz davon ab mit welchen Parametern wir das Skript ausführen. Wird das Skript mit dem Parameter -VmName ausgeführt, installiert dieses in eine leere Hyper-V Maschine noch das Server 2016 TP4 Datacenter Core Betriebssystem mit aktivierten Container Features.

Nach der erfolgreichen Installation des Containerhosts überprüft man mit dem Cmdlet "Get-ContainerImage" ob die Ausgabe das Image "Windows Server Core" aufweist. Anschließend kann mit dem cmdlet aus Listing 1 ein neuer Container erstellt werden

Mit dem Parameter Name setzen wir den Namen des neuen Containers, der Parameter ContainerImageName gibt den Namen des Images an auf das aufgebaut werden soll. Der Switch Name zeigt hier auf einen Switch der standardmäßig eingerichtet wird und über NAT mit dem physikalischen Netzwerkadapter verbunden ist. Anschließend können wir über das cmdlet "Start-Container -Name IIS-Host" den neuen Container starten und uns anschließend über eine Remote Powershell Session mit diesem Container verbinden. Speziell hierfür hat das Cmdlet Enter-PSSession den Parameter "-ContainerName" erhalten. Es ist empfohlen zusätzlich den Parameter "-RunAsAdministrator" zu benutzen um auch administrative Aufgaben umzusetzen. Nachdem wir verbunden sind installieren wir über die PowerShell den IIS Server und beenden danach den Container mit dem cmdlet Stop-Container. Um diese Vorlage jetzt öfters verwenden zu können muss ein neues Container-Image erstellt werden. Dafür wird das cmdlet "New-ContainerImage" verwendet welchem noch die Parameter "-ContainerName IIS-Host", "-Name IIS-Image", "-Publisher demo" und zu guter Letzt der Parameter "-Version 1.0" mitgegeben werden. Wenn dieser Befehl ausgeführt wird, wird der Inhalt der aktuellen Sandbox des Containers in ein neues Container-Image gepackt, sodass wir dieses öfter verwenden können.

Zum Testen erstellen wir einen neuen Container wo wir als Namen des Images jetzt "IIS-Image" verwenden um den neuen Container aufgrund des neuen Images aufzubauen. Nach dem Erstellen starten wir den Container und verbinden uns in diesen um eine kleine Test Seite zu hosten. Dafür löschen wir im wwwroot-Ordner die Standard Seite (iisstart.htm) und erstellen mit dem Befehl aus Listing 3 eine einfache kleine Testseite.

Wenn wir danach vom Containerhost aus mit dem Internet Explorer auf die Seite navigieren wird diese auch angezeigt. (siehe Abbildung 4)

FAZIT

Die nun erstmalig in Windows Server 2016 zur Verfügung stehenden Container, bieten die Möglichkeit für kleine Anwendungen abgeschottete Systeme zu erstellen. War es bisher schon möglich über vollständige Virtuelle Maschinen dies zu erreichen, bieten die Container einen Ressourcen schonenden Weg. In Hinblick auf den Trend zu MicroServices ist das ein überaus nützliches neues Feature im Windows Server.

NEUE BÜCHER

HTML5 UND CSS3 DAS UMFASSENDE HANDBUCH

von Jürgen Wolf



Buch € 44,90
E-Book € 39,90
1264 Seiten,
2., aktualisierte und
erweiterte Auflage
2016, gebunden
ISBN
978-3-8362-4158-8

Das umfassende
Handbuch zum Ler-
nen und Nachschla-
gen! Mit den neuen

Webstandards von HTML5, CSS3 und JavaScript können Sie faszinierende Webseiten gestalten. Nutzen Sie die neuen Möglichkeiten: Video, Audio, 2D- und 3D-Grafiken, lokaler Speicher, abgerundete Ecken, Schatten, unterschiedliche Deckkraft, Transparenzen, Einsatz beliebiger Schriften, neue Farbangaben, u.v.m.
www.rheinwerk-verlag.de/html5-und-css3_4129

GRUNDKURS C

von Jürgen Wolf



Buch € 12,90
E-Book € 11,90
428 Seiten,
2., aktualisierte und
überarbeitete Auflage
2016, broschiert
ISBN 978-3-8362-4114-4

Die kompakte und
aktuelle Einführung
im praktischen Ta-
schenbuch-Format.
Alle Sprachgrundla-
gen werden kurz und
übersichtlich darge-
stellt; viele Codebeispiele und Tabellen sowie eine
Funktionsreferenz machen das Buch zum nützlichen
Begleiter. Auch Einsteiger lernen schnell, ihre ersten
Programme zu schreiben. Mit Übungsaufgaben zur
Lernkontrolle.

www.rheinwerk-verlag.de/grundkurs-c_4108

LET'S CODE ANDROID! APPS ENTWICKELN MIT ANDROID STUDIO

von Sebastian Witt



Download € 39,90
DVD € 39,90
Video-Training, 2016,
Spielzeit 7 Stunden,
Windows, Mac, Linux,
iOS (App erhältlich im
App Store)
ISBN 978-3-8362-4334-6

Der Weg zur eigenen
Android-App beginnt
hier! Sebastian Witt zeigt
Ihnen, wie Sie mit Andro-
id Studio entwickeln. Sie
erfahren, wie Sie Android Studio installieren und ein-
richten, ein modernes Design umsetzen und Ihre Apps
veröffentlichen. Außerdem lernen Sie, wie Sie Ihren
App-Code aktuell halten für neue Android-Versionen.
Programmieren Sie live mit und erleben Sie, wie eine
App Schritt für Schritt entsteht. Lernen Sie, wie Sie
Ihr Design für alle Geräte anpassen und sichergehen,
dass Ihre App auch auf Smart-TV und Smartwatch
eine gute Figur macht.

www.rheinwerk-verlag.de/lets-code-android_4232

SHAREPOINT 2016 FÜR ANWENDER

DAS UMFASSENDE HANDBUCH

von Nicole Enders



Buch € 49,90
E-Book € 44,90
1093 Seiten,
2., aktualisierte und
erweiterte Auflage
2016, gebunden
ISBN
978-3-8362-4164-9

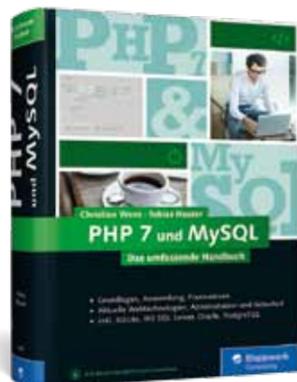
Dieses Buch zeigt
Ihnen, wie Sie als An-
wender mit SharePoint
schnell zu profession-
ellen Lösungen gelan-

gen. Es vermittelt Ihnen alle nötigen Grundlagen zur
Arbeit mit SharePoint und führt Sie Schritt für Schritt
zur optimalen Lösung der an Sie gestellten Aufgabe.
Übungsaufgaben helfen Ihnen dabei, häufige Anforde-
rungen zu meistern.

[www.rheinwerk-verlag.de/
sharepoint-2016-fur-anwender_4131](http://www.rheinwerk-verlag.de/sharepoint-2016-fur-anwender_4131)

PHP 7 UND MYSQL DAS UMFASSENDE HANDBUCH

von Christian Wenz, Tobias Hauser



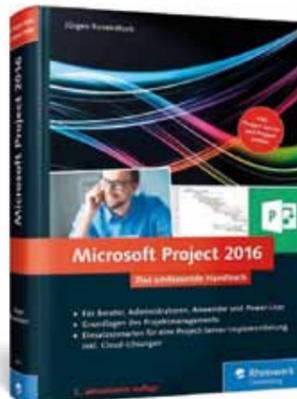
Buch € 44,90
E-Book € 39,90
1039 Seiten,
2., aktualisierte und
erweiterte Auflage
2016, gebunden
ISBN
978-3-8362-4082-6

Sie möchten dynami-
sche Webseiten mit PHP
und MySQL program-
mieren? Mit diesem
umfassenden Handbuch

erhalten Sie eine praxisorientierte Einführung, die
Sie von den Sprachgrundlagen bis hin zur Entwick-
lung professioneller Anwendungen führt. Neben dem
Einsatz von MySQL erfahren Sie, wie Sie auch weitere
Datenbanksysteme effektiv verwenden können. Inkl.
aller neuen Sprachfeatures von PHP 7 wie neue Opera-
toren, Datentypen für Parameter und Rückgabewerte
von Funktionen, geändertes Error-Handling u. v. m.
www.rheinwerk-verlag.de/php-7-und-mysql_4090

MICROSOFT PROJECT 2016 DAS UMFASSENDE HANDBUCH

von Jürgen Rosenstock



Buch € 59,90
E-Book € 54,90
767 Seiten,
3., aktualisierte Auf-
lage 2016, gebunden
ISBN 978-3-8362-4111-3

Ein branchenübergrei-
fend gültiges Szenario
hilft Ihnen, nahezu
alle Funktionen von
Project und Project
Server technisch sowie
prozess- und metho-
denbezogen zu verstehen.

Von der ersten strategi-
schen Überlegung zur Einführung über die technische
Implementierung bis zur umfassenden Anwendung al-
ler Komponenten. Das umfassende Handbuch ermög-
licht es Ihnen, ein Gesamtverständnis der komplexen
Materie des Projekt-Managements zu erhalten und in
Ihrem jeweiligen Teilgebiet zu vertiefen.

[www.rheinwerk-verlag.de/
microsoft-project-2016_4107](http://www.rheinwerk-verlag.de/microsoft-project-2016_4107)

IT-HANDBUCH FÜR FACHINFORMATIKER

DER AUSBILDUNGSBEGLEITER

von Sascha Kersken



Buch € 34,90
E-Book € 30,90
1304 Seiten,
7., aktualisierte und
erweiterte Auflage
2015, gebunden
ISBN
978-3-8362-3473-3

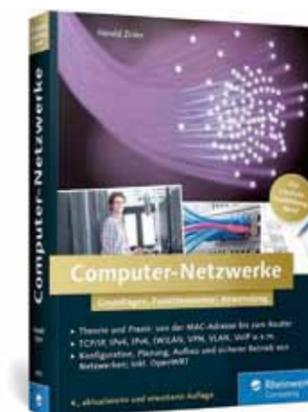
Das Buch vermittelt
alle Grundlagen der In-
formationstechnik, wie
sie Fachinformatiker

in ihrer Ausbildung benötigen: Computerhardware,
Betriebssysteme, Netzwerktechnik, -protokolle und
-anwendungen sowie Grundlagen der Programmie-
rung, Datenbanken und Multimedia. Jetzt mit vielen
Prüfungsfragen und Praxisübungen.

[www.rheinwerk-verlag.de/
it-handbuch-fur-fachinformatiker_3756](http://www.rheinwerk-verlag.de/it-handbuch-fur-fachinformatiker_3756)

COMPUTER-NETZWERKE GRUNDLAGEN, FUNKTIONSWEISE, ANWENDUNG

von Harald Zisler



Buch € 24,90
E-Book € 21,90
449 Seiten,
4., aktualisierte und
erweiterte Auflage
2016, broschiert
ISBN
978-3-8362-4322-3

Als beruflicher An-
wender, Student oder
Auszubildender benö-
tigen Sie zuverlässiges
Grundlagenwissen für
die Arbeit mit moder-
ner Netzwerktechnik.

Deshalb finden Sie hier neben
allen wichtigen Grundlagen zur Planung, Aufbau und
Betrieb vor allem eins: bewährte Netzwerkpraxis!

[www.rheinwerk-verlag.de/
computer-netzwerke_4224](http://www.rheinwerk-verlag.de/computer-netzwerke_4224)

SQL SERVER

PERFORMANCEPROBLEME ANALYSIEREN UND BEHEBEN

von Robert Panther



Buch 12,90 € / 13,40 € (A) / 18,90 CHF
Seiten: 132
Erschienen: September 2016
ISBN: 978-3-86802-162-2

Datenbanken bilden das Rückgrat nahezu jeder Businessanwendung. Dabei spielt oft die Performance eine entscheidende Rolle. Solange diese akzeptabel ist, wird das Thema gerne unterbewertet. Wenn sich

jedoch die Beschwerden von Anwendern häufen, dass lange Wartezeiten oder gar Time-outs entstehen, wird das Thema Performance schnell kritisch. Dieses Buch stellt für alle Versionen bis einschließlich SQL Server 2016 in kompakter Form dar, was zu prüfen ist, und wo mit möglichst geringem Aufwand schnell eine Verbesserung der Performance erzielt werden kann.

<https://entwickler.de/press/sql-server-238962.html>

55 WORDPRESS-TIPPS

SO GELINGT IHR WEBAUFTTRITT

von Vladimir Simović



Preis: 19,90 € / 20,50 € (A) / 28,50 CHF
Seiten: 136
Erschienen: September 2016
ISBN: 978-3-86802-164-6

WordPress ist eines der meistverwendeten Content-Management-Systeme, wenn es um die Umsetzung von Blogs, Onlineshops und anderen Internetpräsenzen geht. Frei verfügbar und schnell installiert, lässt es

sich gut an die eigenen Bedürfnisse anpassen. Dieses Buch liefert Ihnen genau hierfür realitätsnahe und praxiserprobte Anregungen und Anleitungen. In 55 Tipps erfahren Sie, wie Sie mehr aus Ihrer WordPress-Installation herausholen. Der Autor verrät Ihnen unter anderem, wie Sie WordPress noch besser vor Angriffen schützen, eine Installation schneller machen oder Ihren Administrationsbereich komfortabler gestalten.

<https://entwickler.de/press/55-wordpress-tipps-248077.html>

> KOLUMNE

Best of NuGet: WriteableBitmapEx

Autor: ANNETTE HEIDI BOSBACH

Microsoft's NuGet-Paketmanager öffnet die Welt des Drittanbietercodes für Jedermann: mussten Bibliotheken bisher umständlich integriert werden, kann man sich nun mit einem Klick auf den Rücken von Riesen stellen.

In dieser ab Ausgabe II 2016 erscheinenden Kolumne möchte ich Ihnen die besten NuGet-Pakete vorstellen: Neben weitverbreiteten Klassikern, die jeder Entwickler kennen sollte, möchte ich Ihnen dabei auch einige Geheimtipps vorstellen, die in meiner Rüstungselektronik- und systemtechnischen Arbeit behilflich sind.

Insbesondere im Bereich der Rüstungselektronik gilt, dass die Probleme nur dann lösbar bzw. diskutierbar sind, wenn man sie darstellen kann. Da das Nutzen von Chart-Bibliotheken ein Problem für sich ist und die meisten Systemhäuser Diagrammalgorithmen im Haus haben, entsteht über kurz oder lang das Bedürfnis, "klassische" Linien auf den Bildschirm zu bringen.

Normalerweise setzt man als Entwickler an dieser Stelle auf XAML-Konstruktionen. Diese zeichnen sich ab einer gewissen Komplexität durch schlechte Performance aus: Nicht verwunderlich, muss der Grafikstack doch Hunderte oder gar Tausende von Vektoren gleichzeitig verwalten. WriteableBitmaps wären ob ihrer Pixelorientiertheit besser geeignet, werden von Microsoft aber ohne Zeichen-API bereitgestellt.

NUGET ZU HILFE

Die von Rene Schulte entwickelte WriteableBitmapEx-Bibliothek ermöglicht Entwicklern das aufwandsarme Zeichnen in Grafiken, deren Backend durch eine Instanz von WriteableBitmap dargestellt wird.

Zur Vorführung dieses Programms wollen wir eine kleine WP-Applikation verwenden: Integrieren Sie nach der Erzeugung des Projektskeletts das NuGet-Paket WriteableBitmapEx in der aktuellen Version 1.5.0.

Aus Bequemlichkeitsgründen erledige ich die Anzeige des Bitmaps über ein Image-Steuer-element, das Sie folgendermaßen im XAML-Code anlegen:

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Image x:Name="image"
        HorizontalAlignment="Left" Height="340"
```

```
Margin="10,10,0,0" VerticalAlignment="Top" Width="620"/>
</Grid>
```

Damit sind wir zum Programmieren bereit. Die WriteableBitmap-Instanz wird als eine Membervariable des Formulars angelegt:

```
public sealed partial class MainPage : Page{
    WriteableBitmap myBitmap;
    Das eigentliche Erzeugen im Konstruktor erfolgt im Großen und Ganzen nach den schon bekannten Regeln. Interessant ist nur, dass meine Bitmap nun diverse Erweiterungsmethoden aufweist:
    public MainPage(){
        ...
        myBitmap = BitmapFactory.New(200, 200);
        using (myBitmap.GetBitmapContext()){
            myBitmap.Clear(Colors.White);
            myBitmap.DrawLine(0, 0, 200, 200, Color.FromArgb(255, 255, 0, 255));
        }
        image.Source = myBitmap;
    }
}
```

Die Nutzung von GetBitmapContext ist insofern relevant, als sie einen Gerätekontext zurückliefert. Dieser erlaubt uns das automatische Aktualisieren des Bitmaps: Sobald die using-Struktur den Kontext abträgt, wird das Bitmap invalidiert.

DIAGRAMM FÜR JEDERMANN

Histogramme sind eine besonders effiziente Darstellungsform, die meiner Meinung nach viel zu wenig

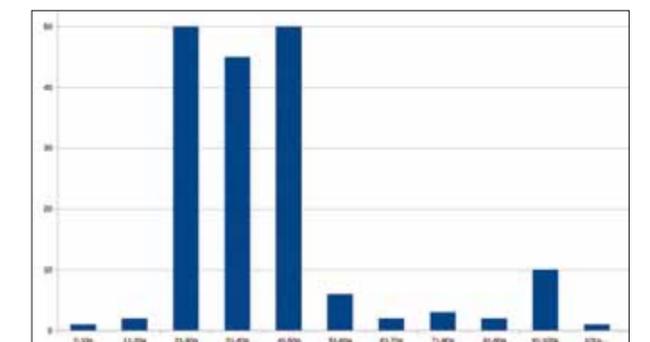


Abbildung 1: Keine Angst vor dem Histogramm!

IMPRESSUM

Herausgeber

ppedv AG
Marktler Straße 15b
84489 Burghausen

Vorstand: Hannes Preishuber
redaktion@visualstudio1.de
www.visualstudio1.de
Tel.: +49 89 60665044
Fax: +49 8677 98894335

Chefredakteur (V.i.S.d.P.)

Dipl.-Ing.(FH)
Hannes Preishuber
CEO ppedv AG

Redaktion

Dipl.-Ing. (FH) Stephan Hahnel
Head of Content ppedv AG
stephanh@ppedv.de

Klara Waltenberger
Mediendesignerin
klaraw@ppedv.de

Anzeigenleitung

Visualstudio1@ppedv.de
+49 8677 9889-60

Redaktionsverwaltung

Kathleen Ergezinger
KathleenE@ppedv.de
+49 8677 9889-60

Autoren

Heiko Angermann
Tim Borowski
Olena Bochkor
Annette Heidi Bosbach
Fabian Deitelhoff
Tam Hanna
Dr. Eckhard Hauenherm
Dr. Veikko Krypczyk
Stefan Lieser
Hannes Preishuber
Naeem Ramzan
Golo Roden
Walter Saumweber
Maximilian Schweigerdt
David C. Thömmes
Ralf Westphal

Layout/Satz:

Ulrike Ammann, Burghausen
ulli.am@web.de

Druck

WIRmachenDRUCK GmbH
Druckerei & Medienproduktion
Mühlbachstr. 7
D - 71522 Backnang

Tel.: +49 711 995 982-20
Fax: +49 711 995 982-21
info@wir-machen-druck.de
www.wir-machen-druck.de

Bildquellen

Titelfoto: Klicker/pixelio.de
S.2: Valentina Razumova / Dreamstime.com
S.14: QuidoX / pixelio.de
S.24: Tony Hegewald / pixelio.de
S.28: David C. Thömmes (privat)
S.29: Microsoft
S.30: Tim Reckmann / pixelio.de
S.52: koya79_iStock_ ThinkstockPhotos
S.56: Rainer Sturm / pixelio.de
S.62: alphaspirt / fotolia
S.83: k3733301/fotosearch.de

Bezugspreise

Einzelhefte: 8,50 EUR (D)
9,50 EUR (EU)
Jahresabo: 32 EUR (D)
35 EUR (EU)

Erscheinungsweise

4 x jährlich

Haftung

Für Fehler im Programmcode oder Text kommt eine Haftung nur bei grober Fahrlässigkeit in Betracht. Für unaufgefordert eingesandte Manuskripte, Datenträger, Produkte und Fotos wird keine Haftung übernommen.

©2016

Alle Rechte vorbehalten, Vervielfältigung nur mit Genehmigung der ppedv AG.
Amtsgericht Traunstein:
HRB 12703

Anwendung findet. Zur Erklärung zeigt Abbildung 1 ein Histogramm von Keksgößen. Die von meinem Automaten gebackenen Kekse sind zum größten Teil zwischen 20 und 50 g schwer. Aufgrund eines Fehlers gibt es leider manchmal auch 100 g schwere Kekse. [s. Abb. 1]

Ein eleganteres Beispiel ist das Anzeigen von Bildinformationen. Ich nutze zur Vorführung dieser Funktion eine Bild aus dem geistigen Eigentum meines Arbeitgebers, das ein altes Nokia zeigt. WriteableBitmapEx erleichtert uns das Laden von Ressourcen nach folgendem Schema:

```
async void runner()
{
    myWorkBitmap = await BitmapFactory.New(1,
    1).FromContent(new Uri(BaseUri, @"/assets/AltesNokia.jpg"));
    using (myBitmap.GetBitmapContext())
```

Damit kann ich mich an die Statistik heranwagen. Eine für didaktische Gründe ausreichende Implementierung durchläuft alle Pixel, um ihre Rotwerte in einem als Binningspeicher bezeichneten Array zu lagern. Dieses stellt ein Feld dar, das die "relativen Häufigkeiten" der einzelnen Elemente beschreibt:

```
int[] myBinning = new int[25];
for (int x = 0; x < myWorkBitmap.PixelWidth; x+=50)
{
    for (int y = 0; y < myWorkBitmap.PixelHeight; y+=50)
    {
        byte r = myWorkBitmap.GetPixel(x, y).R;
        myBinning[r / 10]++;
    }
}
```

Insbesondere bei größeren Bildern – mein JPEG hat eine Auflösung von 1539x2971 – ist die Nutzung der Methode GetPixel im höchsten Maße unsinnig, weil sie bei jedem Durchlauf eine neue Instanz des Bitmaps lockt. Das liegt an der Art der Speicherverwaltung: Im Garbage Collector zeigt sich dies durch das in Abbildung zwei gezeigte Verhalten.

Durch Setzen von ausreichend hohen Sprungweiten – in meiner VM erwies sich in Wert von 200/200 als brauchbar – können wir so weit kommen, ein grundlegendes Binning-Programm zu erzeugen. Seine Anzeige

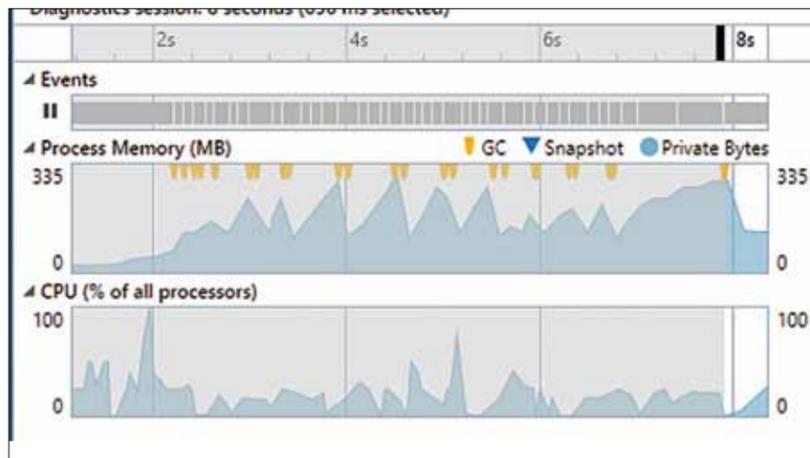


Abbildung 2: Jeder Aufruf erzeugt eine neue Kopie des Bitmaps

am Bildschirm lässt sich sodann folgendermaßen bewerkstelligen:

```
int max = 0;
for (int i = 0; i < 26; i++) {
    if (max < myBinning[i]) max = myBinning[i];
}
myBitmap.Clear(Colors.White);
for (int i = 0; i < 26; i++)
{
    int myHeight = 100 * myBinning[i] / max;
    if (myHeight == 0) continue;
    myBitmap.FillRectangle(i * 10, 100-myHeight,
    (i + 1) * 10, 100, Colors.MediumVioletRed);
}
```

An dieser Stelle ist die continue-Anweisung interessant. Sie ist erforderlich, weil die Bibliothek ein Problem bei Rechtecken mit "identischen" Anfangs- und Endwerten aufweist – die Abbildungen 3 und 4 zeigen den Unterschied.

FAZIT

WriteableBitmapEx ist ideal geeignet, wenn Sie kleinere Operationen mit nicht allzu großen Grafiken durchführen wollen: das Zeichnen von Diagrammen und anderen Visualisierungen ist ein perfektes Beispiel.

Ab einem gewissen Komplexitätsgrad – Stichwort Implementierung von Convolution Kerneln – ist die Nutzung von hardwarebeschleunigten API sinnvoller: Denken Sie daran, dass WriteableBitmapEx die gesamte Arbeit am Hauptprozessor erledigt. Dies ist insbesondere auf mobilen Geräten nicht sonderlich energieeffizient.



Abbildung 3: Mit dem „Bugfix“ funktioniert alles problemlos



Abbildung 4: Wer ihn entfernt, bekommt "hängende" Rechtecke.



Fit für Windows 10

Mit dem MCSA Windows 10 demonstrieren Sie Ihre Kenntnisse zu Konfiguration, Verwaltung und Wartung eines Windows 10-Unternehmenssystems. Für den erfolgreichen Abschluss dieser Zertifizierung müssen Sie diese Prüfung absolvieren:

70-697: Configuring Windows Devices

In nur **6 Tagen** coachen Sie unsere **Microsoft Certified Trainer** mit den offiziellen Unterlagen

MOC: 697—1: Configuring Windows Devices

MOC: 697—2: Deploying and Managing Windows 10 Using Enterprise Services.

Nutzen Sie den kompletten Tag von **08:00 bis 20:00 Uhr** und bereiten Sie sich in **kleinen Gruppen** und **Campusatmosphäre** an **modernsten Standorten** auf die **integrierte Zertifizierungsprüfung** vor.

Die **12-monatige Kurs-Wiederholungs-Garantie** rundet das Lern-Komplettpaket ab.

Dank **FAST PASS** Modell verlieren Sie weniger Zeit im Unternehmen und können nach nur 6 Tagen direkt mit Ihrer Zertifizierung durchstarten.

Unsere MasterClass zum MCSA Windows 10

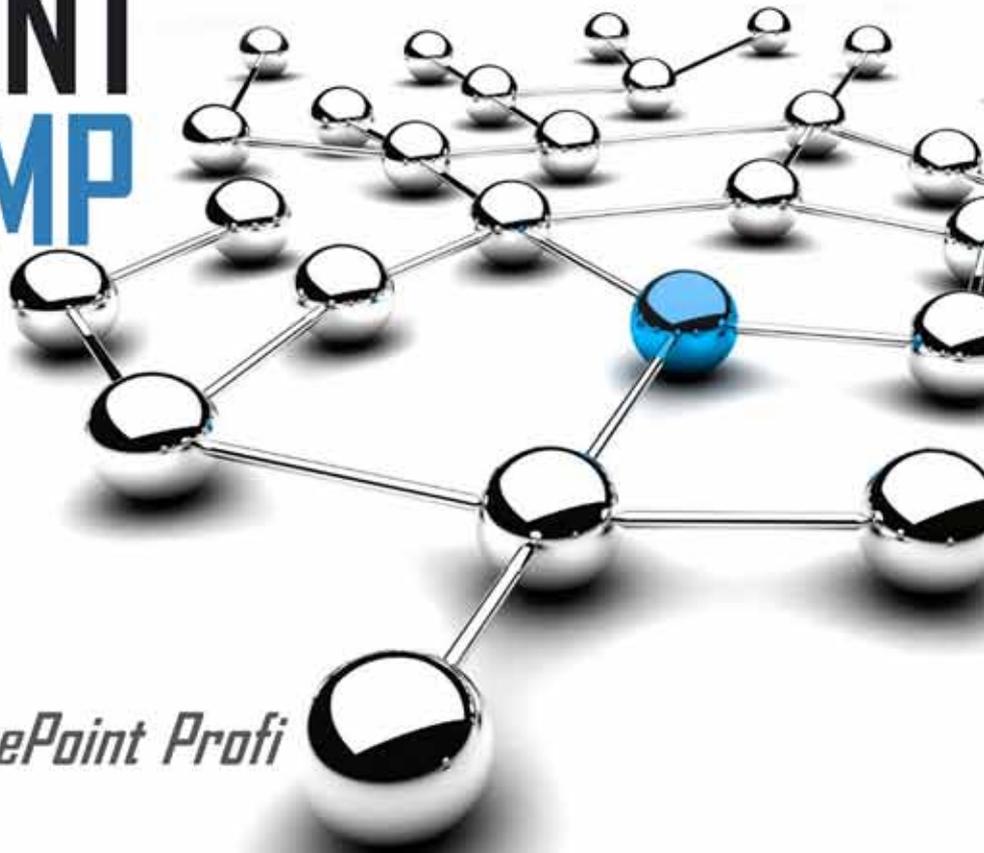
Jetzt Platz sichern!

Mehr Informationen unter:

Tel.: 030—201 430 4 oder minerva@ppedv.de



SHAREPOINT CAMP



Berlin

07. – 11. November

München

12. – 16. Dezember

In 5 Tagen zum *SharePoint Profi*

Melden Sie sich noch heute an und sichern Sie sich Ihren Wunschtermin - die Teilnehmerplätze sind begrenzt!



SharePointCamp.de

XAMARIN CAMP



München
21. - 25. Nov.
Berlin
05. - 09. Dez.

Die besondere Camp Idee der ppedv bietet das optimale Lernkonzept mit mehr Praxisübungen, Industrie-Experten und besonderem Lernambiente - ab 2.499 €



5 Tage Intensiv-Training
<http://xamarin.camp>