DIE ZEITSCHRIFT FÜR DIE MICROSOFT DEVELOPER COMMUNITY

VisualStudio1.de

65
2017

DE 8,50 EUR
A/CH 9,50 EUR
EU 9,50 EUR

BEWEGTE BILDER IM GIF-FORMAT

BLUESCREEN MIT QR-CODE

SCHAU DER WELLENFORM AUFS MAUL!

TESTABDECKUNG MIT NCOVER

FLUENT ASSERTIONS

MASCHINELLES LERNEN

(01)41960661085006

ppedv

Learn .NET

...better than pizza

1. ARCHITEKTUR
2. C# / C++ / CORE
3. XAMARIN / WPF
4. UWP
5. ASP.NET
6. TFS / TESTING
7. ENTITY FRAMEWORK
8. SCRUM & KANBAN

FRESH DELIVERY

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

ORDER NOW!

- +49-8677-988 952 FOR AUSTRIA: +43 1 236 01932
- @PPEDV
- PPEDVAG
- PPEDV.DE/DOTNET

ppedv.de/dotnet

FATAL ERROR – HIDDEN CASH FLOW

von STEPHAN HAHNEL

Wann und wie trifft man Entscheidungen, wenn es um die Anschaffung von Hard- und Software oder um den Einkauf von Dienstleistungen im IT-Bereich geht?

Im Zeitalter von Highspeed-Trading an den Börsen, bei dem in Sekunden oder deren Bruchteilen Millionen von Euro den Besitzer wechseln, kann es auch beim Einkauf Sinn machen, auf die Preisentwicklung der Produkte zu achten. Wie schnelllebig die Entwicklung von Technik geworden ist, sollte jedem, der in diesem Bereich arbeitet, mehr als bewusst sein. Denn welches Unternehmen der freien Wirtschaft möchte gern mehr bezahlen, als es muss. Ständig wird versucht, den günstigsten Preis zu erhaschen. Für den einen oder anderen ist das schon fast ein Sport geworden. Dass dies aber nicht immer möglich ist, steht außer Frage, denn nicht immer kann man den idealen Zeitpunkt abwarten und muss gegebenenfalls auch zu einem ungünstigen Zeitpunkt einkaufen.

Durchaus fragwürdig dagegen ist, wie es sein kann, dass Hardware mit einem Wert von rund 27 Millionen Euro eingekauft, aber nicht benutzt wird und mit jedem weiteren Tag an Wert verliert. Bei der Hardware handelt es sich um Router, die das Bundesinnenministerium im Jahr 2011 eingekauft hat, was aus dem Jahresbericht des Bundesrechnungshofs hervorgeht, der am 15. November 2016 veröffentlicht wurde.

Die Fehlleitung dieser Ressourcen beruht darauf, dass das Projekt „Netze des Bundes“ nicht wie anfangs geplant mit eigenen Personal, sondern mit Personal von einem externen Dienstleister besetzt wurde. Der neue Dienstleister wollte die bereits gekauften Router nicht übernehmen und der Verkäufer verweigerte die Rücknahme. Letztendlich verschenkte das Bundesinnenministerium die Router an andere Behörden, welche aber auch kaum Verwendung dafür fanden.

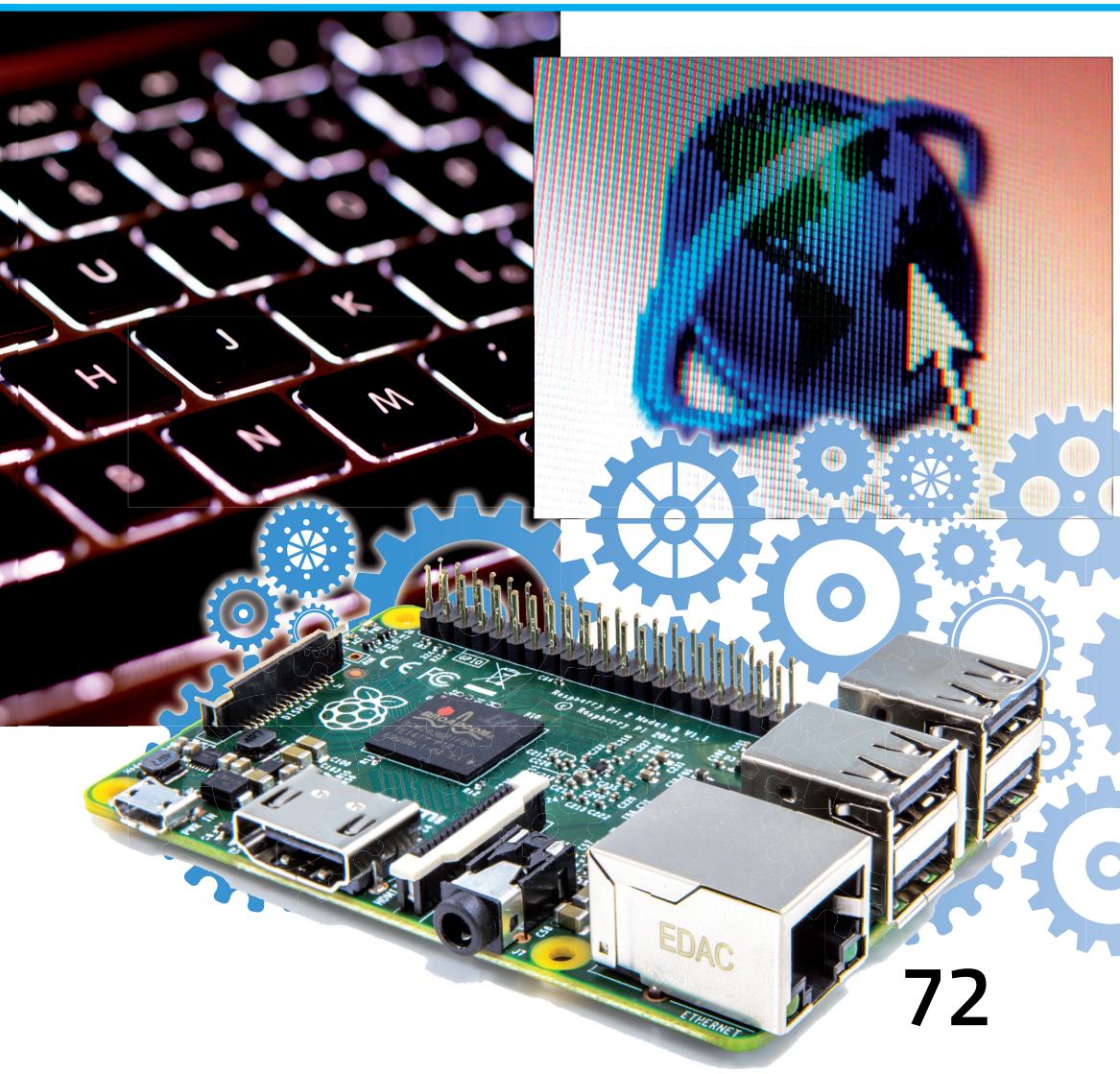
Kann dies in einem marktorientieren privaten Unternehmen auch passieren? Auf den ersten Blick wird vermutlich jeder schreien: „Natürlich nicht!“ Aber relativiert man die gigantisch klingenden 27 Mio. Euro zum Jahreshaushalt der Bundesrepublik (2016), so sind dies nur noch 0,0085%. Das ist weniger als ein zehntausendstel der Gesamtausgaben des Bundes.

Bei einem Jahresumsatz eines mittelständigen Unternehmens von rund 1 Mio. Euro entspräche das gerade einmal 100 Euro.

Natürlich möchte ich hier nicht das Problem von Steuerverschwendungen schönreden, aber in dem zuletzt genannten Kontext wird der eine oder andere sicher mit mir übereinstimmen, dass auch in anderen marktorientieren privaten Unternehmen z. B. ein Monitor oder ein Notebook ungenutzt im Schrank liegt, obwohl man bei der Anschaffung ausgiebige Preisvergleiche durchgeführt hat.
Der gute Vorsatz für 2017: Dieses Jahr wird alles besser!

Designed by Asierromero/Freepik.com

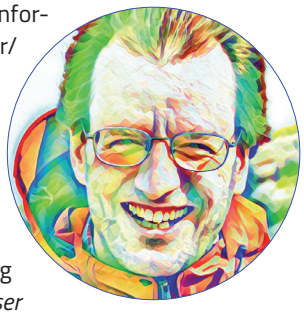
INHALT AUSGABE 65/2017



DIE AUTOREN DIESER AUSGABE

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.*

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. Twitter: *@StefanLieser*



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*; Twitter: *@goloroden*

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*

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.*

ALEXANDER SCHWAHL ist als Programmierer von C# (.net) Anwendungen tätig und entwickelte in seiner Vergangenheit Web-basierte Raumverwaltungssoftware mittels C# (ASP, .net, MVC)

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

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.*

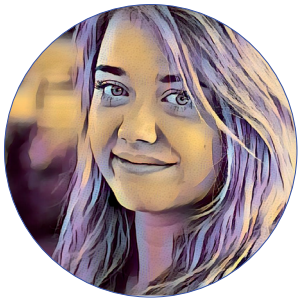
ANNETTE HEIDI BOSBACH betreut Legacysysteme der Tamoggemon Holding k.s. und ist zudem für die Bearbeitung politischer und militärischer Aufträge aus dem Osten und aus Afrika zuständig.

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.

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*
Twitter*: @tamhanna*

JULIA ZACHSKORN

ist Trainer und Autor bei der ppedv AG. Ihre Fachgebiete sind dabei Asp.Net (WebAPI, MVC, WebForms) und AngularJS. Besonders liegt ihr das Zusammenspiel der verschiedenen Technologien. Seit 2016 ist sie Teil des Trainerteams und ist seitdem deutschlandweit bei den entsprechenden Kursen anzutreffen. E-Mail: *juliaz@ppedv.de*



STEPHAN HAHNEL

hat Informatik an der Fachhochschule Lausitz studiert. Seit kurzer Zeit lebt er in Burghausen, wo er bei der ppedv AG als Head of Content tätig ist. Hier koordiniert er unteranderem die Inhalte des Magazins sowie des Videoportals Studios. Für neue Ideen und Feedback hat er immer ein offenes Ohr. E-Mail: *stephanh@ppedv.de;*
Twitter: *@ VisualStudioOne*

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.

LARS KRÄMER

ist leidenschaftlicher Softwareentwickler aus dem Microsoft-Ökosystem. Außerdem ist er in verschiedenen Blogs und Foren aktiv und nebenbei als Rezensionist ist für den Rheinwerk-Verlag tätig.
Website: *http://www.lkraemer.de*
E-Mail: *info@lkraemer.de*

IMPRESSUM

Herausgeber
ppedv AG, Marktler Str. 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/ Stellv. Chefredakteur
stephanh@ppedv.de

Simon Schachtel
Mediendesign
simons@ppedv.de

Anzeigenleitung
visualstudio1@ppedv.de
+49 8677 9889-60

Redaktionsverwaltung
Kathleen Ergezinger
kathleenE@ppedv.de
+49 8677 9889-60

Autoren
Tam Hanna, Lars Krämer, Dr. Veikko Krypczyk, Olena Bochkor, Fabian Deitelhoff, Ralf Westphal, Golo Roden, Stefan Lieser, Walter Saumweber, Annette Heidi Bosbach, Julia Zachskorn, Maximilian Schweigerdt, Hannes Preishuber, Alexander Schwahl, Stephan Hahnel

Layout/Satz:
U. Ammann, Burghausen

Druck
WIRmachenDRUCK GmbH
Druckerei & Medienproduktion
www.wir-machen-druck.de

Bezugspreise
Einzelhefte: 8,50 EUR (D)
9,50 EUR (EU)

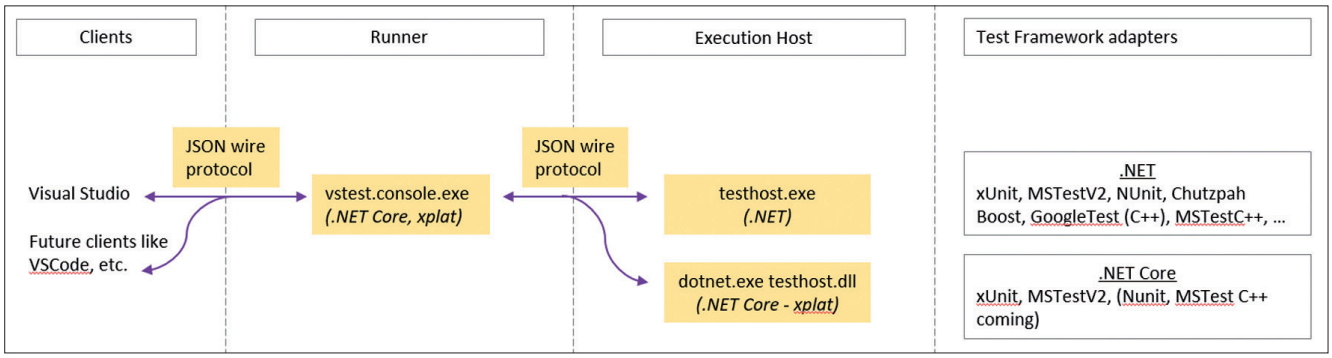
Jahresabo: 32 EUR (D)
35 EUR (EU)

Erscheinungsweise
4 x jährlich

Bildquellen
Titelfotos: Microsoft News Center; Aaron Burden
S. 3/49/53 Freepik.com
S.13 Ethan Vincent
S.14 Julia Vetter
S.15 Patrick Schuchardt
S.16 Martin Müller/pixelio.de
S.18 Jannoono28/freepik.com
S.32 Rainer Sturm/pixelio.de
S.38 Klicker/pixelio.de
S.44 Nadine Papenfuß
S.56 VLADGRIN/Shutterstock
S.60 Tim Reckmann/pixelio.de
S.86 Mart Virkus

Open-Source-Freigabe für VS Test

Seit kurzer Zeit hat Microsoft nun auch die Visual Studio interne Testumgebung als Open-Source-Projekt freigegeben. Der Code von VS Test ist nun, wie zahlreiche andere Projekte auf GitHub, dem webbasierter Online-Dienst, der Software-Entwicklungsprojekte auf seinen Servern zur Verfügung stellt, unter: *https://github.com/Microsoft/vstest* zu finden. Der Code ist mit einer MIT Lizenz versehen und kann somit auch für nicht Open-Source-Projekte verwendet werden. Mit dieser Testumgebung können z.B. Unit-Tests für die Programmiersprachen C#, C++, JavaScript und Python durchgeführt werden. Hinzukommt, dass es Erweiterungen für Testframeworks wie zum Beispiel Google Tests, xUnit, NUnit, Karma und Jasmine gibt.



Die gelben Bereiche gibt es bereits Open-Source

Bildquelle: Microsoft

Tesla liefert neue Fahrassistenz-Software „Autopilot 2" aus

Der kalifornische Elektroautohersteller Tesla installiert seit Januar eine neue Version seiner Fahrassistenz-Software „Autopilot" über das Internet. Dies betrifft alle Fahrzeuge der neusten Generation, somit auch die Modelle, die bereits im Oktober 2016 ausgeliefert wurden. Unter Umständen kann es passieren, dass die Kameras nach der Installation neu justiert werden müssen, was unbedingt zu beachten ist. Die derzeitige Begrenzung der Maximalgeschwindigkeit (rund 72km/h) für die Spurhalte-Funktion „Autosteer" könnte laut Musk wegfallen, wenn ausreichend Daten über die Fahrzeuge gesammelt wurden. Mit dem neuen „Autopiloten" will man einen weiteren Schritt in Richtung autonomes Fahren gehen.

Bildquelle: Tesla

Bildquelle: Microsoft

Azure Deutschland kostenlos testen

Seit Ende letzten Jahres ist es möglich verschieden Azure-Dienste der Microsoft Cloud Deutschland bis zu einem Wert von 170 Euro kostenlos zu testen. Mit der Gutschrift könnten Sie sich z.B. 30 Tage lang 14 virtuelle Computer, 40 SQL-Datenbanken oder eine Speicherkapazität von 8 TB bereitstellen. In dieser Zeit können sie ohne versteckte Kosten, Web-Apps, mobile Apps und API-Apps, die Redis Cache, Search oder Content Delivery Network nutzen, erstellen. Oder nutzen Sie Big Data mit Machine Learning, Streaming Analytics und Hadoop. Mehr Informationen gibt's unter: *https://azure.microsoft.com/de-de/free/germany/*

Smart-Workspace-Technologie – Dell-Canvas

Bildquelle: Dell

Im Januar Dell stellte auf der Consumer Electronics Show (CES) in Las Vegas eine Reihe von Neuheiten vor. Darunter auch der Dell-Canvas, der als Weiterentwicklung des Smart-Desk-Konzepts gilt. In der Kategorie der Smart-Workspace-Technologie soll der Dell-Canvas Designer und Content-Entwickler bei ihrer kreativen Arbeit optimal unterstützt. Damit Benutzer auf natürliche Weise wie mit Stift und Papier arbeiten können, kann der 27 Zoll große QHD-Smart-Workspace schräg oder flach auf einem Schreibtisch eingesetzt werden. Dell ist dabei das problemlose Zusammenspiel mit nahezu jedem Windows-10-Gerät sowie die einfache Verwendung der Softwarelösungen der Dell-Partner wie zum Beispiel Adobe, Autodesk, Avid und Microsoft besonders wichtig.

IoT-Retail-Plattform von Intel

Intel versucht mit Core-i7-Prozessoren und Low-Power-Sensoren die im Einzelhandel verwendete APIs, Software, Hardware und Sensoren auf einer Plattform zu vereinen. Aus diesem Grund ist die Intels Responsive Retail Platform (RRP) auch mit Sensoren und Software anderer Anbieter kombinierbar. Anhand der auf der Plattform will man versuchen Technologien wie Kassensysteme, Software, APIs und Sensoren zusammenzubringen und die Entwicklung und Einführung von Software und Diensten für das Internet der Dinge besonders im Handel voranzutreiben. Ein denkbares Einsatzszenario wäre z. B. die Überwachung von Lagerbeständen. Im Mittelpunkt der Plattform stehen die stromsparenden Sensoren von Intel mit RFID-Funktionalität.

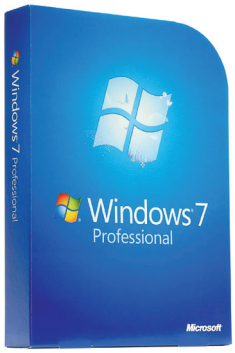
Bildquelle: Intel

Digitalministerium

Die Schaffung eines Digitalministeriums fordert Bundesverkehrsminister Dobrindt. Das neue Ministerium soll zum Beispiel für die Vernetzung der Dinge (IoT), Industrie 4.0, automatisiertes Fahren, digitale Bildung und nicht zu vergessen, den Datenschutz zuständig sein. In der BamS sagte Dobrindt: „Damit können wir unsere Schlagkraft deutlich erhöhen und uns gemeinsam mit der Wirtschaft an die Spitze kämpfen in der neuen digitalen Weltordnung.“ Aktuell liegen die Zuständigkeiten für die digitale Agenda der Bundesregierung verteilt beim Innen-, Verkehrs-, Wirtschafts- und Bildungsministerium.

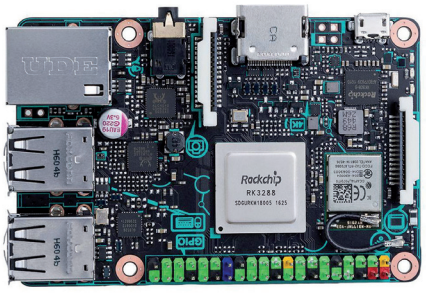
Vorzeitiges „Auf Wiedersehen“ zu Windows 7

In 3 Jahren soll laut Microsoft der erweiterte Support für Windows 7 endgültig auslaufen. Microsoft Deutschland empfiehlt deshalb den rechtzeitigen Umstieg auf Windows 10, da es an Januar 2020 keine Sicherheits-Updates, Aktualisierungen und keinen technischen Support seitens Microsoft mehr geben wird. Mit Windows 7 bot Microsoft 2009 erstmals den Weg in die Cloud, aber heute kann das Betriebssystem nicht mehr mit den gestiegenen Sicherheitsanforderungen mitziehen. Bereits vor zwei Jahren endete der grundlegende Support von Windows 7, der im Allgemeinen fünf Jahre andauert. Nutzer erhalten zwar noch wichtige Sicherheits-Updates, neue Funktionen bleiben aber Windows 10-Nutzern vorbehalten. Bei Support-Anfragen steht Privatkunden weiterhin der Online-Support unter *http://www.support.microsoft.com* zur Verfügung.



Bildquelle: Microsoft

Asus Tinker Board

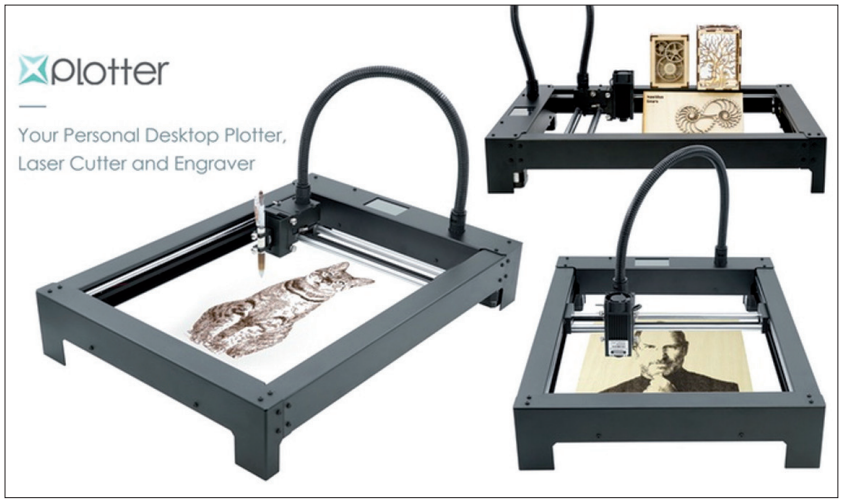


Bildquelle: ASUS

Asus stellt nun auch eine Alternative zum Raspberry Pi bereit. Jede Menge Schnittstellen und Einsatzmöglichkeiten bringt der kleine Mini-PC mit. Dieser erscheint gegenüber seinem direkten Konkurrenten mit einem schnelleren Prozessor und mehr RAM. Hinzu kommt 4K-Videowiedergabe, Gigabit-Ethernet und 192 kHz/24-bit Audio-Unterstützung. Das Tinker Board soll laut Asus Debian Linux und ebenfalls Kodi als Media Center-Lösung.

CROWDFUNDING

XPlotter – Desktop Plotter, Laserschneider und Gravierer



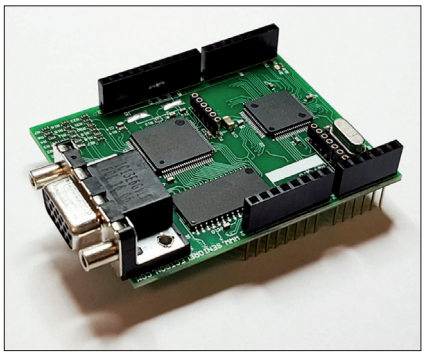
Der XPlotter definiert einen Plotter ganz neu. Durch die Integration von Plotter, Cutter und Lasergravierer in einem Gerät, wird der XPlotter zu einem vielseitigen und dennoch erschwinglichen Desktop-Tool für Künstler, Handwerker und Macher, die ihrer Phantasie nun freien Lauf lassen können. Das Gerät ist in der Lage handgemachte Zeichnungen und Schriften zu simulieren. Es kann Materialien mit großer Genauigkeit zurechtschneiden oder Gravieren. Man kann damit auch kleine Gegenstände bewegen und somit z.B. Brettspiele simulieren.

Zeit: rund 2 Monate
Finanzierungsziel: rund 4.650 €
https://www.kickstarter.com/projects/pineconerobotics/xplotter-desktop-plotter-laser-cutter-and-engraver?ref=category_recommended#

VGADuino II

Mit dieser Erweiterung soll der Nutzer in die Lage versetzt werden, ein Arduino mit jedem TV oder Monitor zu verbinden, der einen VGA-15-Pin-Anschluss besitzt. Auf der Erweiterung werden alle notwendigen Bibliotheken und Samples bereitgestellt, sodass Anwender direkt loslegen und den Fernseher oder Monitor als große Anzeige für den Arduino verwenden können. Folgende Funktionen werden unterstützt: Zeichnen von Formen, wie Linie, Rechteck, Kreis; 11 Schriftgrößen mit Standard-ASCII-Zeichen; 256 Farben, 8bit RGB-Format; Standard-VGA-Ausgang DB15; Bildschirmauflösung: 800x600 60Hz, tatsächliche Pixel: 400x300 60Hz

Zeit: über 1 Monat
Finanzierungsziel: rund 470 €
https://www.kickstarter.com/projects/67935456/vgaduino-ii-new-256-color-graphic-shield-for-ardui?ref=category_newest



Moodo – The Smart Home Fragrance Box

„OK Google, lass es hier nach Sommerregen riechen“: könnte es in Zukunft in einigen Wohnungen heißen. Dies soll mit Moodo möglich werden. Es handelt sich dabei um ein Gerät, dass Ihnen die Kontrolle über den Geruch Ihrer Umgebung ermöglichen soll. Anhand einer App für Ihr Smartphone erlaubt Moodo die Anpassung des Ambientes nach Ihrem Vorlieben. Der Geruchssinn ist ein starker Sinn und kann unsere Stimmung wirksam beeinflussen. Warum nehmen wir dann diese Beeinflussung nicht selbst in die Hand. Moodo ist die intelligente Duftdiffusionsmaschine, die aus verschiedenen Kapseln unterschiedliche Duftkombinationen erstellen kann und im Raum verteilt. Die verschiedenen Kombinationen können in einer App gespeichert und je nach Stimmung abgerufen werden.

Zeit: ca. 1 Monat · Finanzierungsziel: rund 46.800 €
https://www.indiegogo.com/projects/moodo-the-smart-home-fragrance-box--2#/

Run Android OS On Iphone 5 Models Using Our Platform

Sie lieben die Hardware von Apple wie z.B. das IPhone 5, aber bevorzugen Android als Betriebssystem? Dann ist dieses Projekt vielleicht das Richtige für Sie. Es gab bereits frühere Projekte, die aber mit der Umstellung der verwendeten Hardware verschwanden. Ziel dieses Projektes ist es, mindestens Android 4.0 stabil auf einem IPhone 5 laufen zu lassen. Anhand der Software soll sich das neue Betriebssystem spielend leicht installieren lassen.

Zeit: über 1 Monat
Finanzierungsziel: rund 140.260 €

https://www.kickstarter.com/projects/236253871/run-android-os-on-iphone-5-models-using-our-platfo?ref=category_newest

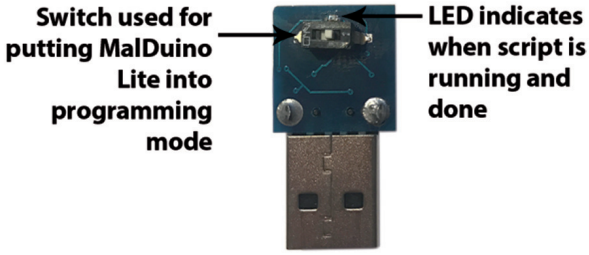
WinterCase

Der Winter ist noch nicht vorbei und gerade deshalb hat das WinterCase eine besondere Daseinsberechtigung. Denn es ist die erste Smartphone-Hülle die aktiv zum Wärmen der Hände eingesetzt werden kann. Drei Stunden lang soll das Case der kanadischen Erfinder bis zu 35°C liefern und somit die Nutzung der mobilen Alleskönner auch in der kalten Jahreszeit unterstützen. Aber auch bei warmen Temperaturen kann das WinterCase nützlich sein, da man mit dem integrierten Akku das Smartphone laden kann. Das Case ist in verschiedenen Farben (Onyx, Arctic, Crimson, and Navy Blue) für aktuelle Geräte wie iPhone 7, iPhone 6, Samsung Galaxy S7, Galaxy S7 Edge und Galaxy S6 erhältlich.

Zeit: ca. 1 Monat
Finanzierungsziel: rund 28.600 €

https://www.indiegogo.com/projects/wintercase-smartphone#/

MalDuino - BadUSB



MalDuino ist ein arduino-powered USB-Gerät mit Tastatur-Eingabe-Funktionen. Einmal eingesteckt, fungiert MalDuino als Tastatur und schreibt Befehle in übermenschlicher Geschwindigkeit. Wozu, könnte man sich fragen? Sie könnten das kleine Helferlein z.B. dazu benutzen, den Desktop-Hintergrund zu ändern oder Einstellungen anzupassen. Penetrationstestern, Bastlern und Witzbolden wird der MalDuino gut dienen! Das Gerät soll in zwei Versionen erhältlich sein. Die Lite-Version stellt 30kB Speicher zur Verfügung, was für die meisten Skripte reichen sollte. Wer mehr Speicherplatz benötigt, greift auf Elite-Version zurück, die mit einer microSD-Karte bestückt werden kann und somit genügend Speicher für alle Szenarien anbietet.

Zeit: Ende Februar · Finanzierungsziel: 600 €
https://www.indiegogo.com/projects/malduino-badusb-arduino-usb#/

EPUBReader für neue Firefox-Generation

EPUBReader ist ein Firefox Add-on, das Firefox in einen eBook-Reader verwandelt. Es ist ziemlich beliebt (Top 30 der beliebtesten Firefox-Add-ons) und wird täglich von ca. 400.000 Nutzern verwendet. Firefox hat sehr umfangreiche Änderungen der Add-on-Schnittstelle angekündigt. Das Ergebnis wird sein, dass EPUBReader in einigen Monaten nicht mehr funktionieren wird. Um dies zu vermeiden, ist es notwendig, EPUBReader fast komplett neu zu programmieren. Im Moment kann EPUBReader keine eBooks öffnen, die größer als etwa 250MB sind. Diese Einschränkung wird mit der neuen Version verschwinden. Die aktuelle Version speichert von allen eBooks Kopien und entpackt die eBooks auf der Festplatte. Die neue Version wird die eBooks im Speicher an der Stelle, an der sie gespeichert sind, öffnen. So wird kein extra Plattenplatz benötigt und die eBooks öffnen sich schneller. Die neue Version auch in

anderen Browsern laufen, die die Web Extensions-Schnittstelle unterstützen (Chrome, Opera).

Zeit: rund 1 Monat
Finanzierungsziel: 25.000 €
https://www.kickstarter.com/projects/569473473/epubreader-for-future-firefox

REGISTRY ZUGRIFF UNTER C#

By ALEXANDER SCHWAHL · Tagged: Registry

Seit .Net 1.1 ist es möglich mittels C# auf die Windows Registry zuzugreifen und viele Programme verwenden diese jeher auch als alternativen Speicherort für Informationen und Einstellungen. Allerdings gibt es wie in jedem System üblich die eine oder andere Seltsamkeit und so ist auch der Registry-Zugriff unter C# keine Ausnahme.

Mit Hilfe der Registry Klasse im Microsoft.Win32 Namespace erhält man Zugriff auf die Root Keys der lokalen Registry. Folgende Keys stehen einem damit also direkt zur Verfügung:

Schlüssel	Besonderheit
ClassesRoot	Keine Schreibrechte
CurrentConfig	Keine Schreibrechte
CurrentUser	-
DynData	Nicht länger unterstützt
LocalMachine	Keine Schreibrechte
PerformanceData	Keine Schreibrechte
Users	Keine Schreibrechte

Hier trifft man nun auf die erste Hürde: Wie man der Tabelle recht einfach entnehmen kann, hat man zunächst bei fast keinen der Root Keys schreibrechte. Dies trifft auch auf alle jeweils untergeordneten Keys zu, da die Rechte hierarchisch vererbt werden. Um in diesen Keys zu schreiben zu können, muss das entsprechende Programm mit Admin Rechten gestartet sein. Ist das nicht der Fall wird bei jedwedem Schreibzugriff eine SecurityException geworfen.

NAVIGATION

```
1: var key = Registry.CurrentUser.
OpenSubKey(@"Printers\Settings\
Wizard");
```

Mittels der OpenSubKey Methode kann man auf jeden beliebigen Key zugreifen, der sich in der Hierarchie unterhalb des gewählten Root Keys befindet. Man benötigt lediglich einen gültigen Registry-Pfad allerdings ohne zusätzliche Angabe des Root Keys. **Achtung**: Sollte der Pfad ungültig sein bekommt man null statt des gewünschten Keys zurück.

Wichtig: Man hat bei dem abgebildeten Zugriff keine Schreibrechte! Um den Schreibzugriff für den Key zu aktivieren muss man zusätzlich zum Pfad einen 2ten Parameter mit true übergeben. Sollten man dies vergessen bekommt man selbst mit Admin Rechten eine SecurityException.

WERTE LESEN

Die Registry besteht im Kern aus nur zwei Elementen: der hierarchischen Key Struktur und den Named Values von denen ein Key beliebig viele enthalten kann. Die Named Values selbst bestehen lediglich aus einem Namen, einem Typ und dem eigentlichen Wert.

Für eine Übersicht welche Named Values im aktuellen Schlüssel zur Verfügung stehen, ruft man die GetValueNames Methode auf:

```
1: var names = key.GetValueNames();
```

Mittels des zurückgelieferten String Arrays kann man nun auf die eigentlichen Werte zugreifen:

```
1: foreach (var name in names)
2: {
3:     var type = key.
       GetValueKind(name);
4:     var value = (???)key.
       GetValue(name);
5: }
```

Hier stellt die API nun die nächste Hürde auf: der Rückgabetyp von GetValue ist Object, da die Registry ihre Werte intern nicht als String sondern typisiert speichert. Um den erhaltenen Wert verwenden zu können, muss also zunächst der Typ des Wertes identifiziert werden. *GetValue Kind* gibt einem hierfür per Enum den Typen des Wertes in der Registry zurück. Anschließend kann der Wert in das korrekte .Net Äquivalent gecastet werden. Nun muss man nur noch zuordnen:

Value Kind	C# Type
Binary	byte []
DWord	int
QWord	long
String	String
ExpandString	String (mit Platzhaltern im Text)
MultiString	String []

Achtung - Sonderfall: In jedem Key darf es genau einen Wert geben der keinen Namen hat. Im Windows Registrierungseditor wird dieser Wert der Lesbarkeit halber mit "(Standart)" betitelt. Möchten man im Code auf diesen Wert zuzugreifen, muss man an die jeweiligen Zugriffs Methoden entweder null oder einen leeren String übergeben, also "" oder String.Empty.

WERTE BEARBEITEN

Um neue Keys zu erstellen ruft man die CreateSubKey Methode auf. Hier sollte darauf geachtet werden, dass der angegebene Name nicht bereits für einen anderen Subkey verwendet worden ist. Ebenfalls wichtig ist, dass Subkeys nicht umbenannt werden können. Sollte dies doch mal nötig sein, muss man leider von Hand den gesamten Inhalt in einen neuen Key transferieren und dann den Alten löschen.

Um einen Key zu löschen verwendet man entweder die DeleteSubKey oder DeleteSubKeyTree Methode. Erstere kann nur leere Keys löschen wohingegen Zweitere alle untergeordneten Keys jeweils mitlöscht.

Mittels der SetValue Methode legt man neue Named Values an oder editiert bestehende Einträge. Um hier zu verhindern das man beim editieren versehentlich neue Named Value anlegt empfiehlt es sich vorher mit GetValue noch mal zu überprüfen

ob ein entsprechender Eintrag bereits vorhanden ist. Abschließend kann man mit DeleteValue ein Named Value auch wieder vollständig aus der Registry löschen.

```
1: try
2: {
3:     using (var key = Registry.CurrentUser.
       OpenSubKey(”Pfad“, true))
4:     {
5:         using(var subkey = key.CreateSubKey(”sub“))
           // Schlüssel Anlegen
6:         {
7:             var typ = RegistryValueKind.DWord;
8:             subkey.SetValue(”Zahl“, 42, typ); //
               Wert Anlegen
9:             subkey.SetValue(”Zahl“, 84, typ); //
               Wert Editieren
10:            subkey.DeleteValue(”Zahl“); //
               Wert Löschen
11:        }
12:        key.DeleteSubKeyTree(”sub“); //
           Schlüssel Löschen
13:    }
14: }
15: catch (Exception ex)
16: {
17:     // Etwas sinnvolles im Fehlerfall tun zb. den
       Fehler loggen.
18: }
```

Vorsicht: Die Key Klasse schreibt Änderungen nicht zwingend sofort zurück in die Registry. Ähnlich wie bei relationalen Datenbanken werden Änderungen zunächst gesammelt und danach über einen separaten Befehl committet. Die Windows Registry kommt einem hier zwar entgegen indem sie in einem extern definiertem Intervall regelmäßig selbst Änderungen comittet aber auf dieses Intervall sollte man sich besser nicht verlassen.

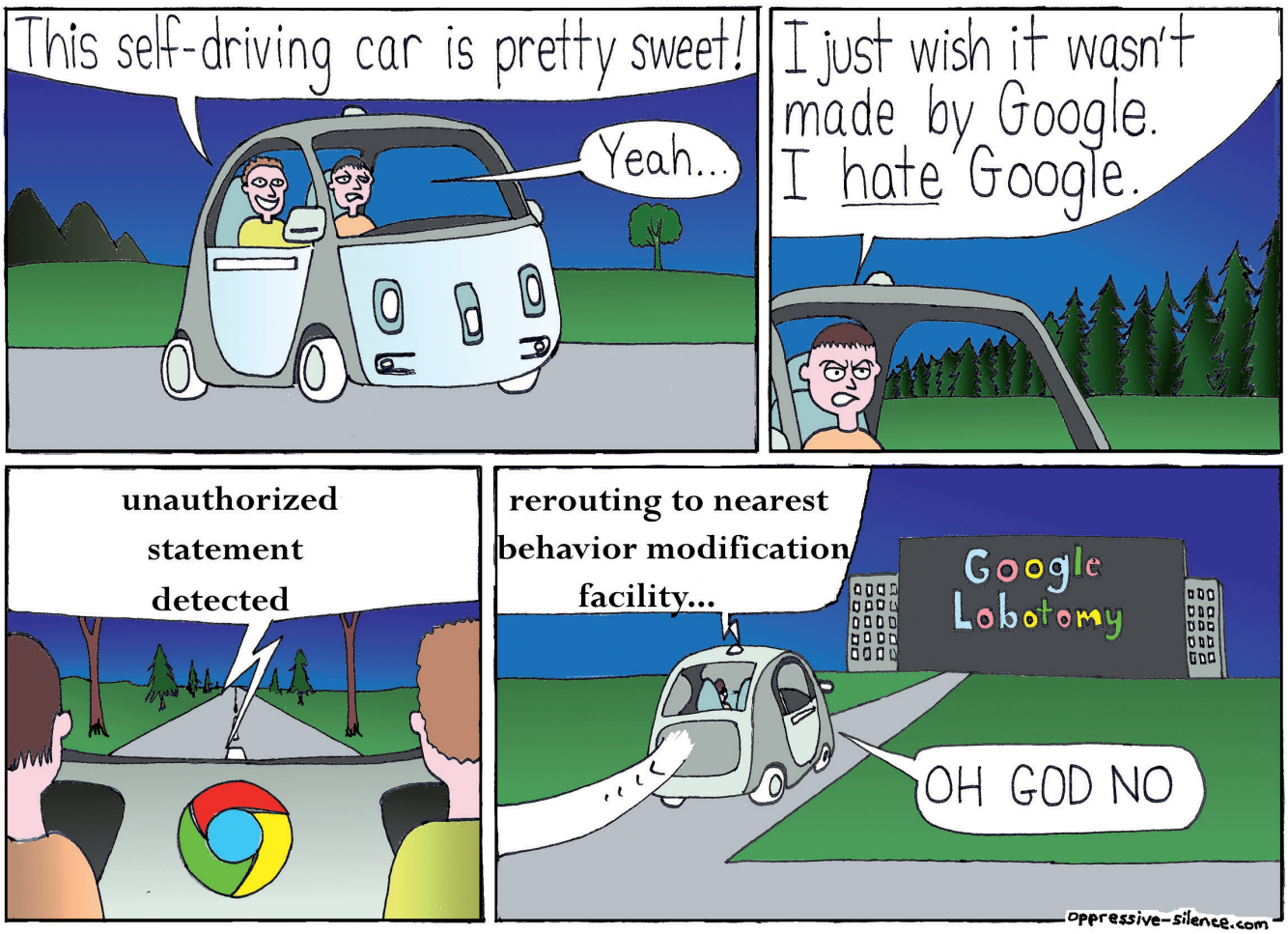
Um diese letzte Hürde zu nehmen und Änderungen manuell zu comitten, verwendet man die Flush Methode des aktuellen Keys. Diese Methode schreibt alle bisher gesammelten Änderungen sofort in die Registry. Da man das aber nicht jedes Mal von Hand machen möchte empfiehlt sich die Verwendung eines using Blocks. Dieser Block ruft implizit die Dispose Methode auf, welche unter der Haube Flush aufruft und gleich noch sauber aufräumt. Damit schlägt man zwei Fliegen mit einer Klappe und hat nun alle nötigen Werkzeuge zusammen, um die Windows Registry effektiv für sich einzusetzen.

Wer noch mehr erfahren möchte, ist mit diesem C# Kurs genau an der richtigen Adresse.

WEITERE INFOS

C# Einstieg: ppedv.de/cs

CARTOON



Mit freundlicher Genehmigung von oppressive-silence.com

ARBEITEN 4.0

© GSP architekten, Julia Vetter

Industrie 4.0 ist in aller Munde. Mal positive mal negative. Niemand weiß was uns wirklich in Zukunft erwarten wird. Mit dem Bürokonzept “Smart Workspace” für das Arbeiten 4.0 realisiert Microsoft Deutschland ein ganz anderes Konzept, bei dem der Mensch im Mittelpunkt steht.

Nach rund 2 Jahren Bauzeit wurde im letzten Jahr die neue Microsoft Deutschlandzentrale in München-Schwabing in Betrieb genommen. Auf 26.000 Quadratmetern, verteilt auf sieben Etagen können die rund 1.900 Mitarbeiterinnen und Mitarbeiter ihrem Tatendrang freien Lauf lassen.

Als eine der modernsten Arbeitsumgebungen in Deutschland, die gemeinsam mit dem Fraunhofer Institut für Arbeitsorganisation entwickelt wurde, steht die Unternehmenszentrale für die Idee einer vernetzten Arbeitswelt: Optimale Möglichkeiten für Teamwork, mehr Selbstbestimmung und persönliche Produktivität sowie ein hohes Maß an Flexibilität, ermöglicht durch moderne Technologien.

© GSP architekten, Julia Vetter

© GSP architekten, Julia Vetter

Arbeitsplatzsouveränität als Treiber von Innovation

„Silodenken und Abschottung haben bei uns ausgedient. Wir verstehen unser Büro als offene Plattform und interdisziplinäres Labor für neue Ideen. Mit der neuen Arbeitsumgebung lösen wir räumliche Trennungen auf und fördern die Zusammenarbeit auch über Abteilungs- und Hierarchiegrenzen hinweg“, erklärt Sabine Bendiek, Vorsitzende der Geschäftsführung von Microsoft Deutschland. Das Konzept des Work-Life-Flow stellt den Menschen in den Mittelpunkt, befreit von Raum und Zeit. Mit der selbstbestimmten Gestaltung des Alltags mit fließenden Übergängen zwischen Arbeit und Privatem anstelle einer starren Verteilung, versucht man auf die Anforderungen der Mitarbeiter einzugehen und ermöglicht ihnen damit mehr Flexibilität bei der Organisation des privaten und familiären Alltags.

Das klingt in der Theorie sehr gut, aber wie wird dies in der Praxis umgesetzt?

In der Walter-Gropius-Straße 5 bietet Microsoft ab sofort eine Arbeitsumgebung mit modernster Ausstattung: Neben den Arbeitsflächen stehen den Mitarbeitern u.a. elf Dachterrassen, zahlreiche Konferenzräume und Meetingflächen, Lounges und ein eigenes Fitnessstudio zur Verfügung. Die Konferenzräume sind auf höchstem technischem Niveau, u.a. mit dem Microsoft Surface Hub und Skype for Business-Lösungen ausgestattet, um den Ansprüchen der vernetzten Teamarbeit und produktiven Meetings zu entsprechen.

Im Mittelpunkt des neuen Konzeptes „Smart Workspace“, stehen die unterschiedlichen Bedürfnisse der Mitarbeiter und ihre individuellen Anforderungen an den Arbeitsplatz. So kann bei Microsoft jeder Mitarbeiter selbst entscheiden in welchen Bereichen und in welcher Art er arbeiten möchte. Dafür stehen vier verschiedene Arbeitsbereiche – von Rückzugsorten für Tätigkeiten, die eine hohe Konzentration erfordern, bis hin zu Büroflächen, die bewusst auf Teamarbeit ausgelegt sind – zur Verfügung. Eine konsequente Weiterentwicklung der Idee des neuen Arbeitens, die der Konzern bereits seit 1998 mit der Einführung des Vertrauensarbeitsortes verfolgt. Durch mehr Selbstbestimmung soll die persönliche Produktivität gesteigert werden.

Die Theorie klingt vielversprechend und sicher sehnt sich manch Arbeitnehmer nach diesen Möglichkeiten in seine Unternehmen. Ob aber eine derart hohe Flexibilität den gewünschten Effekt bringt, wird die Zeit zeigen.

Interessante Frage: Lässt Sich eine Steigerung der Produktivität messen?

Wer noch mehr Flexibilität sucht, sollte sich vielleicht als Digitaler Nomade versuchen. ■

© GSP architekten, Patrick Schuchardt

WO LEBEN DIE BESTEN HACKER?

Foto: Martin Müller/pixelio.de

HackerRank, ein Portal das unter anderem die Programmierfähigkeiten von Codern rund um die Welt getestet hat, stellte vor kurzem eine Statistik online, die die Fingerfertigkeit der Programmierer der einzelnen Länder vergleicht.

Eigenen Angaben zu Folge sind auf dem Portal rund 1,5 Millionen Coder aus allen Regionen des Globus registriert, was die Aussagekraft dieser Statistik unterstreichen soll.

Die Statistik erfüllt die meisten Erwartungen, so befinden sich zum Beispiel die russischen Softwarespezialisten in den Top 3 der Liste. Trotz veralteter Hardware gelang es diesen bereits im kalten Krieg durch effektiven Code das Maximum aus dem vorhandenen Ressourcen heraus zu kitzeln. Dies scheint sich bis heute nicht geändert zu haben, denn die russischen Entwickler sind auf Platz 2. der Rangliste zu finden.

Keine Überraschung: An der Spitze die Chinesen. Hätten Sie sich das gedacht? So oft wie man in letzter Zeit, Hacker aus Fernost für Angriffe auf die verschiedensten Einrichtungen verantwortlich macht, ist das wohl kaum ein Wunder.

Nur wer glaubt, dass wir Deutschen ganz vorn in dieser Statistik zu finden sind, der wird eine Überraschung erleben, denn es reicht gerade mal für den 14. Platz. Ebenfalls weit vor den Deutschen sind die Programmierer aus Polen, Schweiz und Ungarn sowie Frankreich und Tschechien zu finden. Unsere europäischen „Nachbarn" belegen die Plätze 3 bis 5 sowie 8 und 9.

Desillusioniert werden diejenigen, die an die vielgepriesenen Fähigkeiten der indischen Developer glauben. Denn diese erreichen nur einen mittelmäßigen 31. Platz. Nicht viel besser schneidet die USA ab und landet auf Platz 28.

Auf den letzten drei der 50 vergebenen Plätze befinden sich Nigeria (48), Sri Lanka (49) und Pakistan (50).

Für die der Statistik zugrundeliegenden Daten mussten die Programmierer in verschiedenen Bereichen und Programmiersprachen unterschiedliche Aufgabenstellungen und Probleme, wie zum Beispiel Algorithmen lösen. Die Qualität der Lösung und auch die benötigte Zeit flossen in die Bewertung ein.

http://bit.ly/2iDTTme

RADIOBUTTONGROUP IN LISTVIEWS
ASP.NET WEBFORMS

BY JULIA ZACHSKORN

Der Blogartikel ergibt sich aus der Praxis. Beim Programmieren in Asp.Net stieß der Autor auf ein größeres Problem, dass auch nicht (wirklich) irgendwo im Internet gelöst wurde. Das Problem: Radiobuttons in Listviews lassen sich nicht gruppieren und verhalten sich somit wie Checkboxen. Bei den vielen Ansätzen, löste ein Workaround das nächste Problem aus.
Hier in dem Artikel zeigt der Autor die in diesem Fall beste Lösung.

1. PROBLEM: GRUPPIEREN DER RADIOBUTTONS

```
<ItemTemplate>
    <tr>
        <td>
            <%#Eval(”Title“) %>
            <%#Eval(”Speaker“) %>
        </td>
        <td>
           <asp:RadioButton runat=“server“ GroupName= ,
           ”Grpname“ + <%#Eval(”InputBoxName“)%>‘ />
           </td>
    </tr>
</ItemTemplate>
```

Der Radiobutton wird vom Server gerendert und das GroupNamenattribut wird nur angehängt statt zugewiesen. Somit ist der Name jedes einzelnen Radiobuttons einzigartig und unabhängig von einer Gruppe.

```
<input type=“radio“ name=“…$outerListview$ctrl1$innerLi
stview$ctrl0$name = ‚Grpname1‘“/>
<input type=“radio“ name=“…$outerListview$ctrl1$innerLi
stview$ctrl1$name = ‚Grpname1‘“/>
```

WORKAROUND

Es gibt viele Workarounds, in JavaScript als auch C#, die den Gruppennamen manuell austauschen. Doch mit dem Löschen/Verändern des Namens geht auch der Status des Inputs bei den Postbacks verloren, somit weiß man nicht mehr welche Radiobuttons „true“ waren.
Da wir noch die Eingabe des Users auswerten wollen, stellen wir mit jQuery (JavaScript Framework) nur das Verhalten nach. Der gesamte Quellcode wird in document.ready gepackt um sicherzugehen, dass alle Elemente fertig geladen wurden.
Normalerweise werden beim Wählen eines Radios in einer Gruppe die Restlichen deaktiviert. Das Klicken auf den Radiobutton löst sozusagen das Event aus.
Für den nächsten Schritt betrachten wir den Namen der Inputbox nochmal genauer.

```
name=“…$outerListview$ctrl1$innerListview$ctrl1$name =
‚Grpname1‘“
```

Von dem String brauchen wir alle Zeichen ab der Stelle $name = ‚. Das letzte Hochkomma fällt auch noch weg. Was übrig bleibt ist der eigentliche Gruppenname.

```
$(document).ready(function () {
   $(”input[type=‘radio‘]“).change(function () {
      var fullName = $(this).attr(”name“);
      // Der Name als String
      var groupName = ”$name = ‚“, input = $(this);

      //herausfiltern des Namens
      var name = fullName.substr
      (fullName.indexOf(groupName)+ groupName.length,
      fullName.length-1));

      $(” input[name*=“ + name + ”]“).each(function () {
          /*Jedes Gruppenmitglied, dass nicht die Funktion aufgerufen hat wird zurück gesetzt*/
          if (input.attr(”name“)
          != $(this).attr(”name“)) {
             $(this).removeAttr(”checked“);
          }
      });
   });
});
```

2. PROBLEM

Auf die Radiobuttons selbst kann man nicht zugreifen, da sie keine Id besitzen. Selbst wenn man Ids mit einem Eval zuweisen würde, wäre es ein zu großer Aufwand bei langen Listen jeden einzelnen anzusprechen.

WORKAROUND

Aus diesem Grund macht es Sinn sich dem gemeinsamen Elternelement zuzuwenden, denn (in diesem Beispiel die Listview) die Kinderelemente beinhaltet. Die Controles sind in einem Array organisiert und mit Hilfe des Indexes kann man auf diese zugreifen.
Wir laufen im Listview alle Itemtemplates durch und in dem Itemtemplates wiederum die einzelnen Controles.

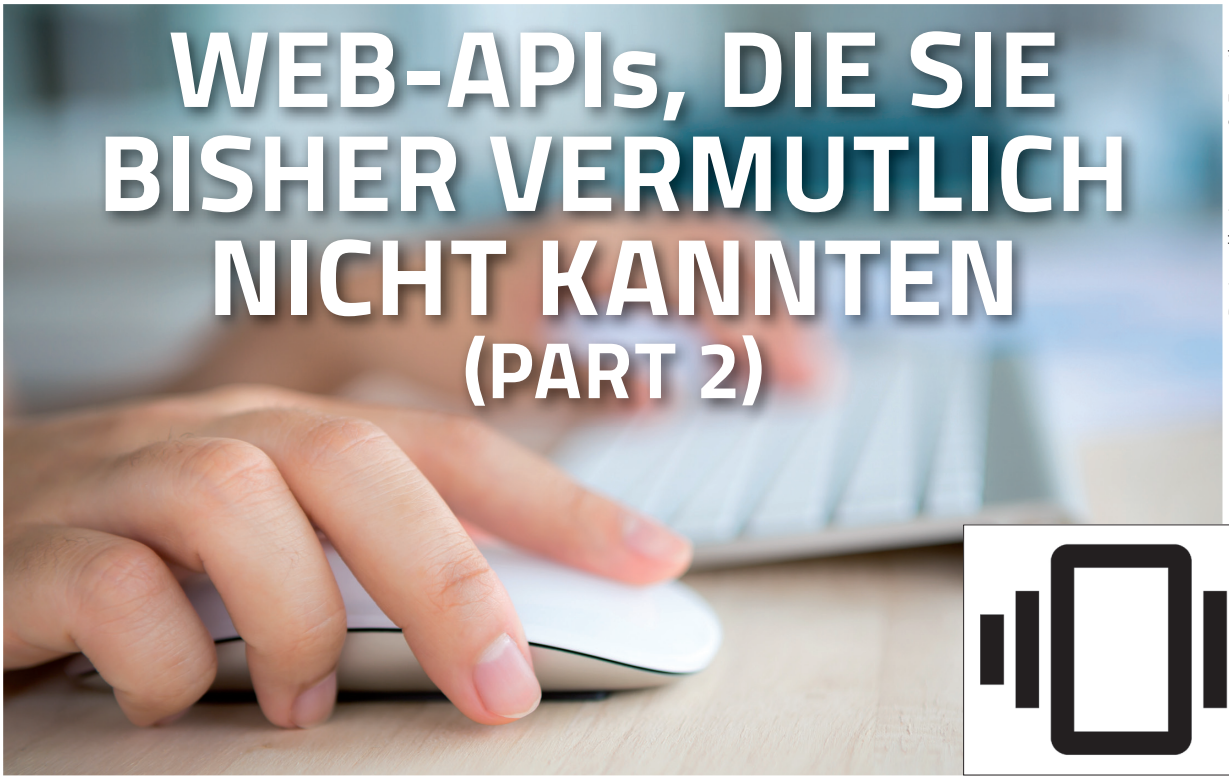
```
for (int i = 0; i <= MyListview.Items.Count - 1; i++){
    for (int col = 0; col <= MyListview.Controls[i].Con-
    trols.Count - 1; col++)    {
        if (MyListview.Controls[i].Controls[col].GetTy-
        pe().Name == ” RadioButton ”)    {
            RadioButton chkTemp = (RadioButton)MyList-
            view.Controls[i].Controls[col];
           // Speicher die Daten evtl in einer
           Datenbank
        }
    }
}
```

Es gibt die Möglichkeit Itemtemplates nach eigenen Kriterien anzusprechen. Die Klasse, dessen Objekte wiedergegeben werden, bestehen aus Properties. Aus dem Properties kann man den Datenschlüssel definieren und dann in der If-Abfrage verwenden.

```
<asp:ListView ID=“ MyListview“ runat=“server“
DataKeyNames=“Id“>
    <ItemTemplate>
  …
    </ItemTemplate>
</asp:ListView>
```

So hat man z.B. bei Sätzen aus der Datenbank gleich die richtige Id.

```
DbModel.YourTable.Find(MyListview.DataKeys[i].Value);
```



Designed by Jannoon028 - Freepik.com

Im zweiten Teil der Web APIs stellt Maximilian Schweigerdt die Vibration und PointerLock API vor.

VIBRATION-API

In den meisten modernen Mobilgeräten ist Vibration Hardware verbaut. Die Vibration API bietet Web-Anwendungen die Möglichkeit auf diese Hardware zuzugreifen, um für eine bestimmte Dauer in einem Muster zu vibrieren. Dadurch kann man den Nutzer physikalischen Feedback geben, wenn z. B. eine Form Validierung fehlschlägt.

Eine einzige Methode steuert dabei die Vibration: window.navigator.vibrate(). Die Methode erwartet als Parameter die Dauer der Vibration in Millisekunden. Ein Muster kann in ein Array abgelegt werden. Die geraden Indizes repräsentieren die Schwingungsdauer und die ungeraden Indizes die Verzögerung vor der nächsten Vibration.

```
1: if (‚vibrate‘ in window.navigator) {
2:     // Vibriert 1 Sekunde lang
3:     navigator.vibrate(1000);
4:     // Array: Muster aus Vibration und Pausen
5:     navigator.vibrate([100, 400, 500, 300, 1000]);
6: }
```

Besitzt ein Gerät die Funktionalität nicht, passiert nichts.

POINTER LOCK | MOUSE LOCK API

Die Pointer Lock API stellt Methoden und Eigenschaften zur Verfügung, um auf Deltawerte des Mauszeigers zugreifen zu können. Mehr als das: Es ist möglich, die Maus auf ein bestimmtes HTML-Element zu sperren. Wenn die Maus gesperrt ist, kann man ohne den Fokus zu verlieren, die Maus bewegen der User kann Schaltflächen bedienen und den Mauszeiger hin und her wischen, ohne sich Gedanken über den Anwendungsbereich machen zu müssen. Dadurch kann er nicht versehentlich in eine andere Anwendung klicken, die den Mausfokus von unserer Anwendung nehmen würde. Mit Pointer-Lock stoppt eine Manipulation nicht, wenn die Maus über den Browserrand hinausgeht. Dieses Verhalten ist perfekt für First-Person Games, um 3D-Objekten zu drehen oder für 360° Panoramas (http://bit.ly/2hAPWys) und vieles mehr.
Ähnlich wie mouse-capture, werden Ereignisse kontinuierlich ausgelöst, wenn die Maus bewegt wird, stoppt aber nicht, wenn der Mausbutton wieder losgelassen wird. Die API unterscheidet sich in den folgenden Punkten von mouse-capture:

- Pointer Lock ist persistent: Die API gibt die Maus erst wieder frei, wenn ein bestimmter API-Aufruf eintritt oder der Benutzer eine spezielle Geste macht.
- Die API ist nicht auf Browser oder Bildschirmgrenzen beschränkt.
- Events werden fortlaufend ausgelöst, unabhängig ob eine Maustaste gedrückt ist oder nicht.
- Der Mauszeiger wird durch Pointer Lock verborgen.
- Zugriff auf rohe Mausdaten einschließlich der relativen Mausbewegungen

1. Wir benötigen ein Canvas-Element auf den die Pointer Lock Eigenschaft anwendet wird.

```
1: var c = document.createElement(‚canvas‘),
2:     ctx = c.getContext(‚2d‘),
3:     x = c.width = window.innerWidth,
4:     y = c.height = window.innerHeight,
5:     size = 40;
6:
7: document.body.appendChild(c);
```

Die Pointer Lock API erweitert das Dokument-Interface, sowohl um eine neue Eigenschaft, als auch um eine neue Methode. Die Eigenschaft pointerLockElement() wird für den Zugriff auf die aktuell gesperrten Elemente verwendet. Die Dokument-Methode exitPointerLock() wird verwendet, um die Zeigersperre zu verlassen.

2. Cross Browser Pointer Lock Objekte erstellen.

```
8: c.requestPointerLock = c.requestPointerLock
9:                      || c.mozRequestPointerLock
10:                     || c.webkitRequestPointerLock;
11:
12: document.exitPointerLock = document.exitPointerLock
13:                          || document.
                             mozExitPointerLock
14:                          || document.
                             webkitExitPointerLock;
15:
16: document.pointerLockElement = document.
                                  pointerLockElement
17:                              || document.
                                 mozPointerLockElement
18:                              || document.webkitPoin-
                                 terLockElement;
```

3. Hinzufügen eines Klick Eventlisteners auf unser Canvas-Element, um eine Umschaltfunktion zu realisieren. Die Umschaltfunktion sperrt den Mauszeiger, wenn der User auf das Canvas-Element Klickt. Bei einem erneuten Klick des Users, wird der Mauszeiger wieder freigelassen.

```
19: c.addEventListener(‚click‘, function() {
20:     if (document.pointerLockElement === c ) {
21:         document.exitPointerLock();
22:     } else {
23:        c.requestPointerLock();
24:     }
25: }
```

4. Browserübergreifende pointerlockchange Events. Das Event wird Ausgelöst, wenn der Pointer Lock -Zustand sich ändert. Entweder durch den Aufruf von request /-exitPointerLock(), oder durch den Nutzer, wenn er z. B. die ESC Taste drückt.

```
26: document.addEventListener(‚pointerlockchange‘,
    lockChanged, false);
27: document.addEventListener(‚mozpointerlockchange‘,
    lockChanged, false);
28: document.addEventListener(‚webkitpointerlockchange‘,
    lockChanged, false);
29:
30: document.addEventListener(‚pointerlockerror‘,
    pointerLockError, false);
31: document.addEventListener(‚mozpointerlockerror‘,
    pointerLockError, false);
32: document.addEventListener(‚webkitpointerlockerror‘,
    pointerLockError, false);
```

5. Das pointerlockerror Event wird ausgelöst, wenn ein Fehler eintritt. In der Funktion wird dann eine Fehlermeldung ausgegeben.

Das pointerlockchange Event wird ausgelöst, wenn sich der Pointer Lock-Zustand ändert, danach wird die Funktion lockChanged aufgerufen. In der Funktion wird geprüft, ob ein Element gesperrt ist. Wenn ja, soll ein mousemove Event auf das Dokument gelegt werden und wenn aktuell kein Element den Zeiger sperrt, soll er das mousemove Event wieder entfernen.

```
33: function pointerLockError() {
34:   console.error(‚Pointer Lock fehlgeschlagen‘);
35: }
36:
37: function lockChanged() {
38:   if (document.pointerLockElement === c ) {
39:     console.info(‚Pointer locked‘);
40:     document.addEventListener(”mousemove“, setXY,
        false);
41:   } else {
42:     console.info(‚Pointer unlocked‘);
43:     document.removeEventListener(”mousemove“,
        setXY, false);
44:   }
45: }
```

6. Das Mausevent wird um zwei weitere Attribute - movementX und movementY erweitert. Der Wert ist die Differenz zwischen den aktuellen screenX, screenY und den der vorherigen screenX/Y Properties.

```
46: function setXY(e) {
47: // Die ”Geschwindigkeit“ zur aktuellen Position
    Hinzuaddieren
48:    var moveX = e.movementX || e.mozMovementX ||
       e.webkitMovementX || 0;
49:    var moveY = e.movementY || e.mozMovementY ||
       e.webkitMovementX || 0;
50:
51:    x += moveX;
52:    y += moveY;
53:
54:    // Werte im Canvas darstellen
55:    drawShape();
56: }
```

7. In der Funktion drawShape werden die Grenzen abgefragt und die relative Position im Canvas dargestellt.

```
57: function drawShape() {
58:   // Abfragen der Grenzen
59:   if (x > c.width + size) x = 0;
60:   if (y > c.height + size) y = 0;
61:   if (x < - size) x = c.width;
62:   if (y < - size) y = c.height;
63:
64:   with (ctx) {
65:       fillStyle = ”#333“;
66:       fillRect(0, 0, c.width, c.height);
67:       // Draw shape
68:       fillStyle = ”#0ff“;
69:       fillRect(x, y, size, size );
70:   }
71: }
```

Das Ergebnis finden Sie im ppedv-Blog unter:
http://blog.ppedv.de/max

RÜCKKEHR DES GUTEN

von TAM HANNA

Windows Phone 7 traf Entwickler wie eine 7.62mm-Kartusche. Microsoft und die Community suchten gemeinsam, dem Umstellungsschock durch ein mit diversen Steuerelementen vollgestopftes Toolkit Drall zu nehmen.

Das Universal Windows Toolkit orientiert sich insofern an seinem Vorgänger, als es den p.t. Entwicklern eine Vielzahl interessanter Widgets anbietet. Zusätzlich sind diesmal jedoch auch einige Backend-Services mit von der Partie, die das Ansprechen von Cloudservices und Co ermöglichen.

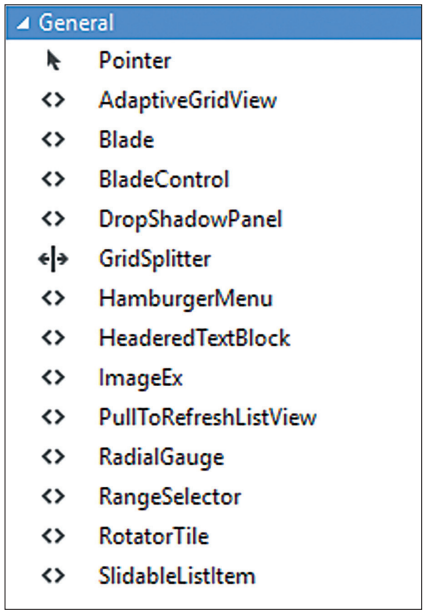


Abbildung 1: Das Universal Windows Toolkit bringt ein gutes Dutzend Widgets mit

ABER NUN LOS!

Erstellen Sie im ersten Schritt in Visual Studio 2015 – das Vorhandensein des Updates 3 ist verpflichtend – ein Projekt auf Basis der Vorlage Visual C# -> Windows -> Universal -> Blank App. Im Code zum Heft hört das Projekt auf den Namen PPEDVUWT1.

Das eigentliche Deployment des Universal Windows Toolkits erfolgt dann über NuGet – öffnen Sie den Paketmanager und suchen Sie nach dem String "Microsoft.Toolkit.UWP". Microsoft bietet das UWT in Form mehrerer Pakete an – wer den gesamten Funktionsumfang nutzen möchte, muss folgendes deployen:

- Microsoft.Toolkit.Uwp
- Microsoft.Toolkit.Notifications
- Microsoft.Toolkit.Services
- Microsoft.Toolkit.Uwp.UI
- Microsoft.Toolkit.Uwp.UI.Animations
- Microsoft.Toolkit.Uwp.UI.Controls

Wer die zum Framework gehörende Steuerelementbibliothek bequem aus dem GUI-Designer von Visual Studio heraus ansprechbar haben möchte, öffnet eine beliebige XAML-Datei und klickt die Toolbar rechts an. Wählen Sie im daraufhin erscheinenden Popup die Option Add Tab, und vergeben Sie einen Namen nach Wahl. Klicken Sie die neue Rubrik sodann abermals rechts an, klicken Sie auf Choose Items..., um im Dialog den Tab "Universal Windows Components" zu selektieren. Ein Klick auf Browse öffnet einen CommonDialog, in dem Sie in das Verzeichnis *c:\Users\%USERNAME%\.nuget\packages\Microsoft.Toolkit.Uwp.UI.Controls* navigieren. Ebenda findet sich ein zur UWT-Version passender Ordner, in dem die Datei Microsoft.Toolkit.Uwp.UI.Controls.dll liegt. Wählen Sie diese aus, um die Integration in den Designer abzuschließen – auf den Rest des Deployments gehen wir im nächsten Abschnitt ein. Nach getaner Arbeit, ist die Steuerelementliste um die in Abbildung eins gezeigten Widgets reicher.

Zur Ergänzung der angebotenen Dokumentation bietet sich das

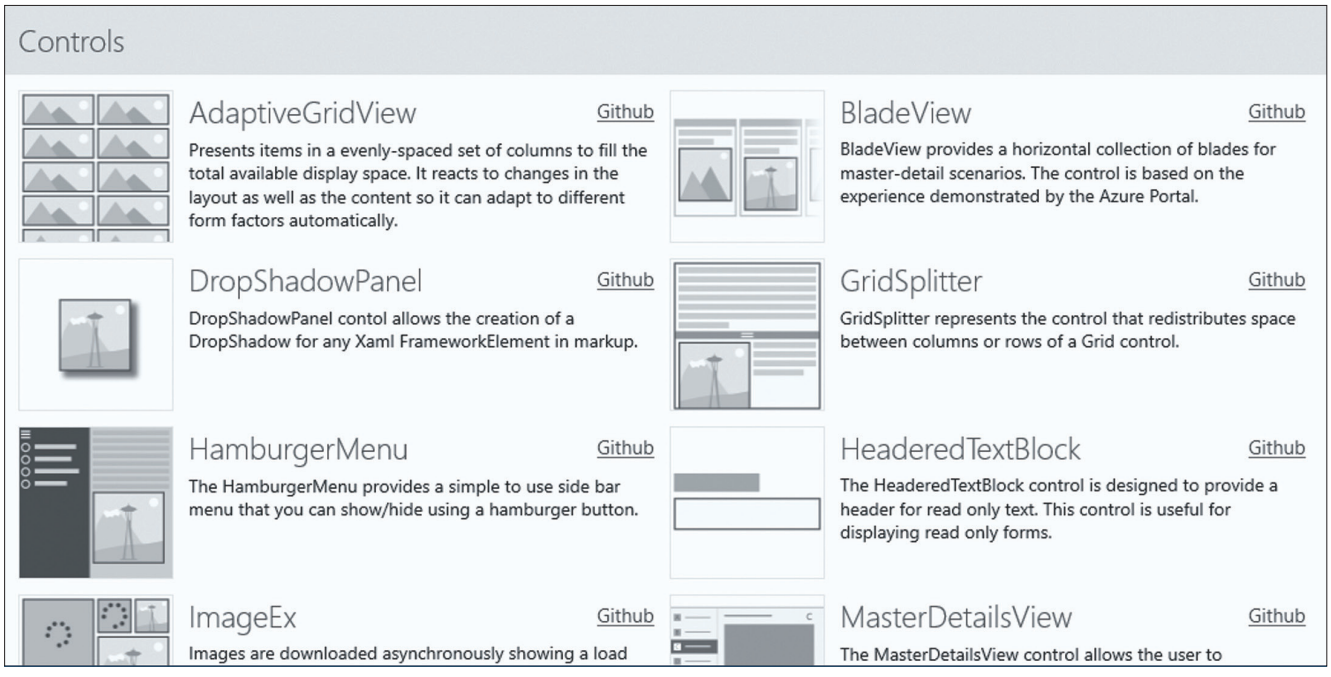


Abbildung 2: Dokumentation als App – ein durchaus interessanter Ansatz

Herunterladen der unter *https://www.microsoft.com/de-de/store/p/uwp-community-toolkit-sample-app/9nblggh4tlcq* bereitstehenden Beispielapplikation an: es handelt sich dabei um die in Abbildung zwei gezeigte App, dass die meisten in der UWP enthaltenen Features kurz anreißt.

WILLKOMMEN BEIM BURGERBRATER

Hamburger sind mittlerweile auch im Mobilmarkt Kult – und das nicht nur als Nahrung von Entwicklern. Die drei übereinanderliegenden Linien brüllen "MENÜ": mit dem UWT können Sie sie sofort in ihre Applikation integrieren.

Im ersten Schritt ist hierzu die XAML-übliche Deklaration eines Namespaces notwendig, die dem Parser das Einbinden der betreffenden Klassen ermöglicht:

```
<Page
    x:Class="PPEDVUWT1.MainPage"
    xmlns:controls="using:Microsoft.Toolkit.Uwp.
UI.Controls"
```

Öffnen Sie sodann die nach dem weiter oben besprochenen Verfahren erweiterte Toolbox, und binden Sie ein neues HamburgerMenu per Drag&Drop ein. Eine grundlegende Implementierung sieht so aus – achten Sie darauf, dass Sie den Trigger von Hand auf die obere linke Bildschirmkante bewegen müssen:

```
    <Grid Background="{ThemeResource
    ApplicationPageBackgroundThemeBrush}">
        <controls:HamburgerMenu Content="HamburgerMenu"
        HorizontalAlignment="Left" Margin="10,10,0,0"
        VerticalAlignment="Top"/>
    </Grid>
```

Daraufhin entsteht ein – weitgehend dysfunktionales – Containersteuerelement, das beim Anklicken seinen im Moment aus einer Textbox bestehenden Inhalt am Bildschirm hin- und herschiebt. Zur eigentlichen Realisierung eines Menüs müssen wir die einzelnen Items als XAML-Templatelemenete anlegen und in das Gesamtsteuerelement einschreiben.

Microsoft unterstützt Entwickler beim Deployment hauseigener Hamburger-Menüs mit zwei vorbereiteten DataTemplates, die die in Menü anzulegenden Elemente beschreiben. Sie finden sich im Beispielcode in der Datei HamburgerMainPage.xaml - um Ihnen das Herunterladen des Files zu ersparen, hier den der in den folgenden Schritten verwendete HamburgerMenuItem-Code:

```
    <Page.Resources>
        <DataTemplate x:Key="HamburgerMenuItem"
        x:DataType="controls:HamburgerMenuGlyphItem">
            <Grid Width="240" Height="48">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="48" />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>
                <FontIcon Grid.Column="0"
                          Margin="12,12,11,12"
                          FontFamily="Comic Sans"
                          Glyph="{Binding Glyph}"
                          Foreground="White" />
                <TextBlock Grid.Column="1"
                          Text="{x:Bind Label}"
                          Foreground="White"
                          FontSize="16"
                          VerticalAlignment="Center" />
            </Grid>
        </DataTemplate>
    </Page.Resources>
```

Für typografisch unerfahrene Entwickler: Wir haben es hier mit einem Element zu tun, dessen Label mit einem Kapitälchen aufgehübscht wird. Die Verwendung der FontIcon-Klasse ist dabei insofern von Vorteil, als sie sich das Anlegen von Bitmap Ressourcen sparen können - das aufwändige Rendern des Fonts erledigt die in Windows integrierte TrueType-Engine.

Sorgfältige Vergleicher bemerken an dieser Stelle eine Änderung im Bereich der FontFamily des FontIcon-Steuerelements. Microsoft nutzt dieses Item von Haus aus mit Programmsymbolen: Die vorgegebene Schriftart ist für Nutzer von textuellen Icons nicht sinnvoll, als sie keine brauchbaren Zeichen enthält und zur Laufzeit nur Platzhalter rendert.

Dieses Problem lässt sich durch Anpassen der FontFamily-Eigenschaft lösen - als alter Windows-Held nutzt der Autor hier das universal verhasste Comic Sans MS, weil das Front auf so gut wie allen Windows-Systemen verfügbar ist. Dass Sie in praktischen Applikationen eine andere Schriftart verwenden sollten, folgt aus der Logik.

Damit sind wir auch schon zum Einfügen der Inhalte bereit. Entfernen Sie im ersten Schritt die automatisch angelegte Inline-Content-Deklaration, die das Element mit einem einzelnen Menüeintrag ausstattet:

```
<controls:HamburgerMenu Content="HamburgerMenu"
```

Im nächsten Schritt bekommt das Hamburger-Menü folgende Struktur eingeschrieben, um die Renderinglogik des Steuerelements als Ganzes zu parametrieren:

```
<controls:HamburgerMenu . . .
     ItemTemplate="{StaticResource HamburgerMenuItem}"
     ItemClick="HamburgerMenu_OnItemClick">
</controls:HamburgerMenu>
```

Für die Anlieferung der Einträge ist sodann eine ItemsSource notwendig, die als Kind des HamburgerMenu-Steuerelements entsteht und folgendermaßen aussieht:

```
<controls:HamburgerMenu . . .
    <controls:HamburgerMenu.ItemsSource>
        <controls:HamburgerMenuItemCollection>
            <controls:HamburgerMenuGlyphItem
            Label="Erstes Feld" Glyph="A"/>
            <controls:HamburgerMenuGlyphItem
            Label="Zweites Feld" Glyph="B"/>
            <controls:HamburgerMenuGlyphItem
            Label="Drittes Feld" Glyph="C"/>
            <controls:HamburgerMenuGlyphItem
            Label="Viertes Feld" Glyph="D"/>
        </controls:HamburgerMenuItemCollection>
    </controls:HamburgerMenu.ItemsSource>
</controls:HamburgerMenu>
```

Wie in den meisten anderen Fällen gilt auch hier, dass das XAML zwar lang, aber nicht sonderlich kompliziert ist. Wir legen eine ItemsSource an, die im nächsten Schritt

KOMPILIERE MICH!

Wer die Solution mit dem UWP-Code öffnet, steht nach dem Laden vor einer Wand voller Compilerfehler. Diese lassen sich durch eine Rekompilation beheben – dies ist sinnvoll, weil sie den Code erst danach mit IntelliSense und Co durchforsten können.

Abbildung 3: Das Hamburger-Menü ist einsatzbereit

mit vier Elementen bevölkert wird. Dank ihrer Position als Kind des Steuerelements wird sie diesem automatisch eingeschrieben.

Zur Sicherstellung der Kompilierbarkeit müssen wir einen Eventhandler anlegen, der im Fall des Anklickens seines Menüeintrags aufgerufen wird. Hier reicht für's erste folgende, leere Methode:

```
private void HamburgerMenu_
OnItemClick(object sender, ItemCli-
ckEventArgs e){        }
```

Damit sind wir zur Programmausführung bereit. Starten Sie die Applikation, und erfreuen Sie sich an einem halbtransparenten Menü. Dies liegt daran, dass aus Beispielen entnommener XAML-Code oft unerwünschte Nebeneffekte aufweist - die Deklaration der Hintergrundfarbe erfolgt in Microsoft's Beispiel „nebenbei" bzw. an anderer Stelle, und fehlt nun nach dem Übernehmen des betreffenden Codes. Zur Behebung sind folgende Änderungen notwendig:

```
<Grid Background="{ThemeResource
ApplicationPageBackgroundTheme-
Brush}">
    <controls:HamburgerMenu
    PaneBackground="DarkGray"
```

Führen Sie die Applikation sodann nochmals aus, um sich am in Abbildung drei gezeigten Menü zu erfreuen.

ABER NUN MIT OPTIONEN

In diesem Zusammenhang muss angemerkt werden, dass das Hamburgermenü auch mit separaten Optionseinträgenm umgehen kann. Es handelt sich dabei um Elemente, die am unteren Ende des Menüs aufscheinen. Schon aus Gründen der Vollständigkeit wollen wir auch diese Vorgehensweise kurz vorführen.

Zur Nutzung von Optionen müssen wir die Attribute Options ItemTemplate und OptionsItemClick einschreiben: OptionsItemTemplate legt dabei das XAML-Steuerelement fest, das für die eigentliche Anzeige der Optionseinträge zuständig ist. Klicks auf diese Elemente werden nicht über den allgemeinen Eventhandler abgearbeitet - der dafür zuständige Code wird über ds zweite Attribut beim Betriebssystem angemeldet. Modifiziert sieht das ganze Programm dann folgendermaßen aus:

```
<controls:HamburgerMenu Height="400"
. . .
    OptionsItemTemplate="{Static
    Resource HamburgerMenuItem}"
    OptionsItemClick="HamburgerMenu_
    OnOptionsItemClick">
```

Damit müssen wir eine weitere ItemSource einschreiben - sie verhält sich im Großen und Ganzen so, wie weiter oben besprochen:

```
<controls:HamburgerMenu.
OptionsItemsSource>
    <controls:HamburgerMenuItemColl
    ection>
        <controls:HamburgerMen
        uGlyphItem Label="Über"
        Glyph="Ü"/>
    </controls:HamburgerMenuItemCol
    lection>
</controls:HamburgerMenu.
OptionsItemsSource>
```

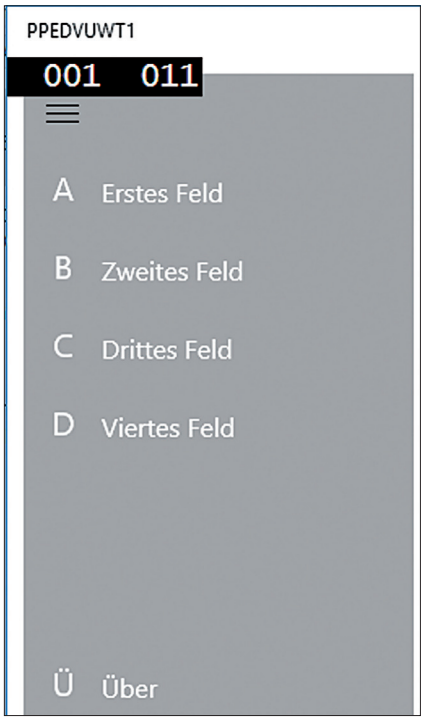


Abbildung 4: Die Optionsitems werden immer auf der Unterseite des Bildschirms dargestellt.

Nun sind noch zwei Änderungen erforderlich. Erstens müssen wir einen weiteren leeren Eventhändler anlegen, zweitens muss das Hamburger-Menü eine Höhe zugewiesen bekommen - von Haus aus ist es nämlich nur so hoch wie die Summe seiner Items. Im Fall von Optionsmenüs ist dies insofern unschön, als sie direkt unter den restlichen Einträgen kleben würden und visuell so nicht vom Rest abgetrennt erscheinen.

Damit ist auch diese Version des Programms zur Ausführung bereit. Abbildung vier zeigt, was sie sich erwarten dürfen.

SAMMLE INPUT EIN

Eines der witzigsten Steuerelemente des Silverlight Toolkits für das Windows Phone 7 war mit Sicherheit ein vom iPhone inspirierter Ein-Aus-Schalter für binäre Entscheidungen. Für die Universal Windows Platform steht ein ganzes Array interessanter Widgets bereit: Beginnen wir mit dem Deployment der Tachometern nachempfundenen RadialGauge.

Auf der Workstation des Autors kam es an dieser Stelle zu einem Fehler der Bauart "MissingMethodException: Method not found: ‚Void Windows.UI.Composition.Visual.put_Size(System.Numerics.Vector2)" - es handelt sich dabei um eine Beisserei zwischen dem Preview-Renderer von Visual Studio und dem Steuerelement. Zur Laufzeit funktioniert alles wie erwartet: Das primäre Ärgernis besteht darin, dass das Widget im vorliegenden Zustand nicht per Drag&Drop verschoben werden kann.

Ein einfacher Weg zur Lösung dieses Problems ist das manuelle Einschreiben der vier für die Positionsberechnung notwendigen Attribute nach folgendem Schema:

```
<controls:RadialGauge
Height="160" Margin="192,188,0,0"
HorizontalAlignment="Left"
VerticalAlignment="Top"
Width="370"/>
```

Ab diesem Zeitpunkt erscheint das Steuerelement wie in Abbildung fünf in der WYSIWYG-Ansicht: seine Größe und sein Aufenthalt lassen sich zwar nicht direkt verschieben, sonst ist aber alles so, wie man es erwarten würde. Platzieren Sie danach eine weitere Gauge, und weisen sie den beiden Elementen Namen zu.

Als nächstes Widget bringen wir einen RangeSelector zum Einsatz; es handelt sich dabei um eine Art "zweiwertigen Slider", der dem Nutzer das Einstellen eines Wertebereichs ermöglicht. Die eigentliche Konfiguration der Parameter ist ob der Eingabeverifikation etwas haarig – eine funktionierende Beispielimplementierung würde folgendermaßen aussehen:

```
<controls:RangeSelector  . . .
        RangeMax="55"
        Maximum="100"
        RangeMin="25"
        Minimum="0"/>
```

Maximum und Minimum beschreiben dabei den minimalen bzw. Den maximalen Wert, der auf dem jeweiligen Steuerelement eingestellt werden kann. RangeMin und RangeMax beschreiben sodann die eigentlichen Auswahlwerte, die von Seiten des Nutzers eingegeben wurden. Achten Sie darauf, dass Visual Studio keine Eingaben nach dem Schema RangeMax>Maximum oder RangeMin<Minimum akzeptiert. Ändern Sie daher immer zuerst den "kritischeren" der beiden Werte, um den anderen sodann von Hand "nachzuführen".

Damit können wir die im Steuerelement enthaltenen Werte abernten und in den beiden "Uhren" zur Anzeige bringen. Auch in der RadialGauge finden sich vier Parameter, die folgendermaßen deklariert sind:

```
<controls:RadialGauge  . . .
        MinAngle="-151"
        MaxAngle="151"
        Maximum="100"
        Minimum="0"/>
```

MaxAngle und MinAngle beschreiben hierbei, wie weit das Zeigerinstrument in die jeweilige Richtung ausschlagen darf. Maximum und Minimum ordnen diesen Zuständen numerische Werte zu – im Fall unseres Beispiels würde ein Wert von 0 einem Ausschlag von -151 Grad entsprechen.

Eine Analyse des Widgets ergibt, dass wir es mit einer mehrwertigen Funktion zu tun haben: ValueChanged wird immer dann aufgerufen, wenn sich einer der beiden Werte ändert. Da wir hier keine besonders komplexe Logik handhaben, triggern wir in diesem Fall einfach ein komplettes Update beider Instrumente:

```
private void RangeSelector_ValueChanged(object sender,
Microsoft.Toolkit.Uwp.UI.Controls.RangeChangedEventArgs e)
{
    RdGauge1.Value = (sender as RangeSelector).RangeMin;
    RdGauge2.Value = (sender as RangeSelector).RangeMax;
}
```

Damit ist auch diese Version unseres Programms einsatzbereit: Abbildung sechs demonstriert die funktionierende Korrelation, die allerdings erst nach dem erstmaligen Anklicken des Sliders beginnt. Zur Lösung des Problems könnte man entweder auf Datenbindungen setzen oder den Inhalt der beiden Gauges auch in der XAML-Datei oder im Konstruktor parametrieren.

STÄRKER ANIMIERT

Man kann Steve Jobs lieben oder hassen: Mit den Konsequenzen seiner Arbeit muss man als Handcomputertechniker auf jeden Fall leben. Erfreulicherweise

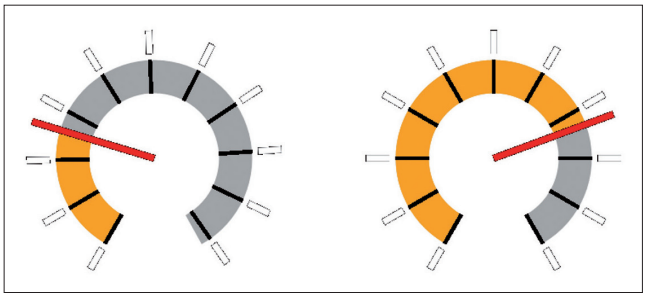


Abbildung 6: Mit RangeSelector und RadialGauge kommt unter Windows 10 echtes Cockpitfeeling auf

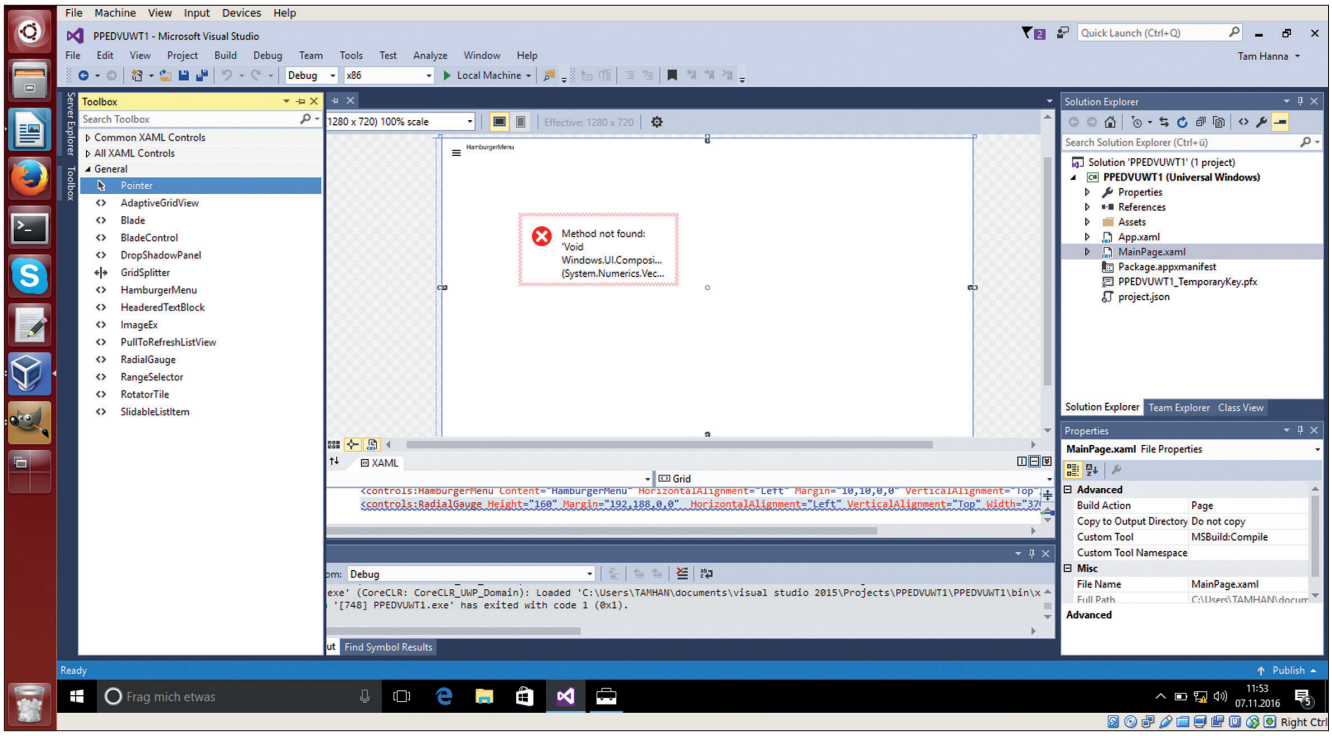


Abbildung 5: Fehler im Renderingpfad sind ärgerlich – wirken sich aber nicht sonderlich auf den GUI-Designer aus

programmieren Entwickler ihre Animationen nur selten selbst – das Keyframe-Prinzip hat sich fast vollständig durchgesetzt.

Hierzu eine kleine Erklärung: Bewegungen lassen sich durch vier Kennwerte vollwertig beschreiben. Wer den Start- und den Endpunkt, die Bewegungsfunktion und die Geschwindigkeit bestimmt, kann unter Nutzung mathematischer Mittel den Aufenthaltsort zu jedem Zeitpunkt feststellen.

Das Universal Windows Toolkit bringt eine Gruppe von sogenannten Easing Functions mit, die die Bewegung der verschiedenen Elemente beschreiben. Abbildung sechs zeigt die relevante Seite der Beispielapplikation, die sich auch als Funktionsliste zweckentfremden lässt (s. Abbildung 7).

Als kleines Beispiel für ihre Verwendung wollen wir an dieser Stelle unser Radialanzeigeelement um eine Sprunglogik erweitern, die das Widget anklicken eines Buttons zur Veränderung seiner Größe animiert. Öffnen Sie dazu die XAML-Datei, und beginnen Sie mit der Erweiterung der folgenden beiden Elemente:

```
<Page . . .
    xmlns:interactivity="using:Microsoft.Xaml.
    Interactivity"
    xmlns:behaviors="using:Microsoft.Toolkit.Uwp.
    UI.Animations.Behaviors" . . .>
```

Als erstes müssen wir die Page um einen Verweis auf den Namespace erweitern, der die diversen Bewegungslogiken enthält. Interaktive Verhaltensweisen werden in XAML prinzipiell als Kinder des zu animierenden Steuerelements angelegt. Im Fall unserer radialen Anzeige sind folgende Änderungen notwendig:

```
<Grid Background="{ThemeResource
ApplicationPageBackgroundThemeBrush}">
   . . .
      <interactivity:Interaction.Behaviors>
          <behaviors:Scale x:Name="Scale"
               ScaleX="{Binding ScaleX.Value,
               Mode=OneWay}"
               ScaleY="{Binding ScaleY.Value,
               Mode=OneWay}"
               CenterX="{Binding CenterX.Value,
               Mode=OneWay}"
               CenterY="{Binding CenterY.Value,
               Mode=OneWay}"
               Duration="{Binding Duration.Value,
               Mode=OneWay}"
               Delay="{Binding Delay.Value,
               Mode=OneWay}"
               AutomaticallyStart="{Binding Automati-
               callyStart.Value, Mode=OneWay}"/>
      </interactivity:Interaction.Behaviors>
</controls:RadialGauge> . . .
```

Dieser vergleichsweise umfangreiche Code - XAML ist mit Sicherheit keine kompakte Programmiersprache - deklariert ein Verhalten namens Scale, das über das Interaktionsverhaltenssystem von XAML aktiviert werden kann.

Unsere nächste Aufgabe besteht darin, eben dies zu tun. Dazu wird unser Formular um einen Auslöseknopf erweitert, der mit folgenden Eventhandler zu verdrahten ist:

```
private async void hoppyRunner() {
   await RdGauge1.Scale(centerX: 0.5f,
        centerY: 0.5f,
        scaleX: 2f,
        scaleY: 2f,
        duration: 200, delay: 0).StartAsync();
   await RdGauge1.Scale(centerX: 0.5f,
        centerY: 0.5f,
        scaleX: 1f,
        scaleY: 1f,
        duration: 200, delay: 0).StartAsync();
}
private void CmdHoppy_Click(object sender, RoutedEven-
tArgs e)
{  hoppyRunner(); }
```

Interessant ist an dieser Stelle die Vergabe der Parameter: mathematisch erfahrene Entwickler würden annehmen, dass das Steuerelement nach einer Skalierung um den Faktor zwei und einer darauffolgenden Skalierung um den Faktor eins doppelt so groß erscheinen würde. Dies ist falsch: die Werte beziehen sich immer auf den ursprünglichen Zustand des Steuerelements.

Das bedeutet, dass unser Steuerelementen ersten Schritt um den Faktor zwei wächst – dahinter steht die

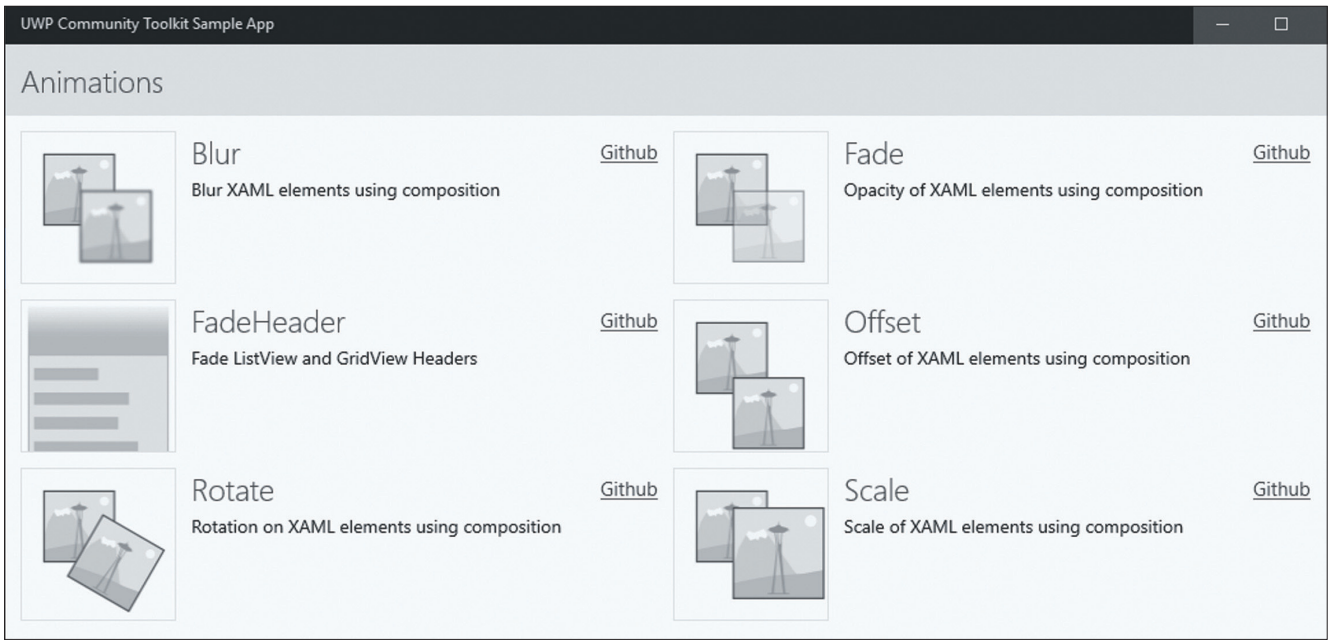


Abbildung 7: Jedes Kästchen entspricht einer Easing Function

Formel Originalgröße * 2. Die nächste Operation nutzt den Faktor eins, um zur Ursprungsgröße zurückzukehren – dies wird durch Originalgröße * 1 beschrieben, was sich zu Originalgröße abkürzen lässt. Der Duration-Parameter wird übrigens in Millisekunden abgefragt, was das genauere Timing der Bewegungen erleichtert.

Wer das vorliegende Programm zu kompilieren sucht, wird von Visual Studio mit einer Fehlermeldung der Bauart "‚RadialGauge' does not contain a definition for ‚Scale' and no extension method ‚Scale' accepting a first argument of type ‚RadialGauge' could be found" abgespeist. Dies ist insofern verwirrend, als die Fehlermeldung nicht auf die wirkliche Ursache hinweist: die Lösung des Problems ist das Fehlen folgender Deklaration:

```
using Microsoft.Toolkit.Uwp.UI.Animations;
```

TWITTER UNTER FEUER

Auch wenn es für Entwickler schwer vorstellbar ist: der Durchschnittsuser findet nichts dabei, die Feeds von Freunden und Kollegen mit mehr oder weniger belanglosen Meldungen zuzuspammen. Besonders viral wird die Chose, wenn der User für seine Spamming-Tätigkeit die eine oder andere Belohnung erwarten darf.

Der erste Schritt zur Nutzung von Twitter ist das Anmelden der Applikation beim Kurznachrichtendienst: öffnen Sie dazu das unter *https://apps.twitter.com/app/new* bereitstehende Entwicklerportal, und melden Sie ein neues Programm an. Achten Sie darauf, im Feld Callback URL eine URL anzugeben – diese muss nicht unbedingt gültig sein.

Im nächsten Schritt können wir die Initialisierung des TwitterServices anstossen. Dazu ist die statische Methode Initialize vorgesehen – da sie die anzugebenden Strings an ihre Applikation anpassen müssen, drucken wir absichtlich kein vollständiges Sample ab:

```
TwitterService.Instance.Initialize(ConsumerKey.Text, ConsumerSecret.Text, CallbackUri.Text);
```

Damit ist ihr Programm gegenüber Twitter authentifiziert. Im nächsten Schritt muss der User dazu animiert werden, seine Kontodaten einzugeben – dies erfolgt über ein asynchrones Steuerelement, das über das Toolkit folgendermaßen angefordert werden kann:

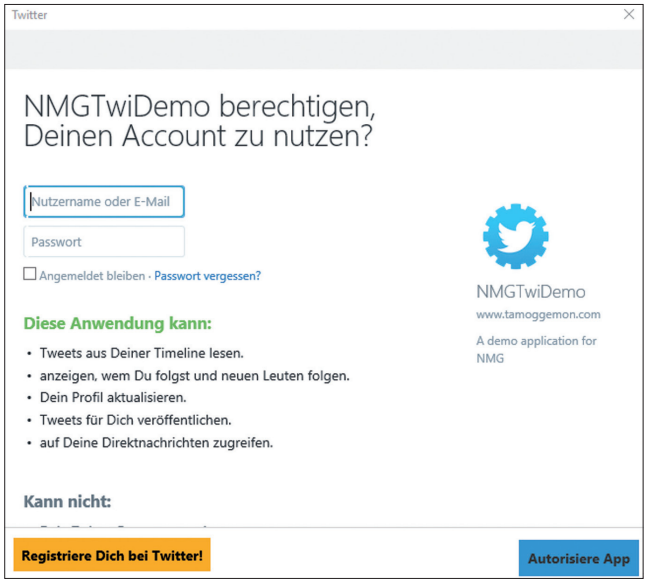


Abbildung 8: An diesem Pop-Up führt kein Weg vorbei

```
private void CmdTwitterLogin_Click(object sender, RoutedEventArgs e)        {
   loginRunner();
}
private async void loginRunner()        {
   TwitterService.Instance.Initialize(ConsumerKey.Text,
   ConsumerSecret.Text, CallbackUri.Text);
   if (!await TwitterService.Instance.LoginAsync())
   {
      return;
   }
   CmdTwitterTest.IsEnabled=true;
}
```

Wer unser Snippet in ein Beispiel einpflegt und dieses ausführt, sieht ein Popupfenster – das eigentliche Einloggen erfolgt ausschließlich über von Twitter bereitgestellte Steuerelemente (siehe Abbildung acht).

Zum Absetzen von Nachrichten ist ein weiteres asynchrones Kommando erforderlich – aus Gründen der Didaktik zeige ich hier eine komplette Routine, die das Freischalten einer Belohnung für das Versenden eines Tweets demonstriert. Beginnen wir mit dem eigentlichen Senden der Kurznachricht über folgenden Code:

```
private async void freeStuffRunner() {
    await TwitterService.Instance.
    TweetStatusAsync("Hello World from the PPEDV
    program sample!");
```

Twitter ist sich des "Spammings" durch Applikationen von Drittanbietern mittlerweile übrigens bewusst. Das mehrfache Absetzen des gleichen Tweets führt zu einer Exception – berücksichtigen Sie dies auf jeden Fall beim Design der Social Integration ihres Programms.

Ab einem gewissen "Helligkeitsgrad" der User kommt es vor, dass ein abgesetzter Tweet nach dem Erhalten der Belohnung aus dem Feed gelöscht wird. Die hier gezeigte Methode versucht, diesem Problem durch Prüfen des Vorhandenseins des Tweets einen Riegel vorzuschieben. Aus codetechnischer Sicht ist die Routine derweil nicht sonderlich interessant: nach dem Absetzen des Tweets beschaffen wir eine Liste aller Tweets, die sich im Konto des gerade angemeldeten Users befinden. Danach wird diese unter Nutzung eines Iterators durchsucht:

```
//Wait a minute
var user = await TwitterService.Instance.GetUserAsync();
IEnumerable<Tweet> myEnumerable = await TwitterService.
Instance.GetUserTimeLineAsync(user.ScreenName, 50);
IEnumerator<Tweet> myEnumerator = myEnumerable.
GetEnumerator();
bool foundSth = false;
myEnumerator.Reset();
try {
   for (int i = 0; i < 10;i++) {
      if (myEnumerator.Current != null && myEnumerator.
      Current.Text.Contains("PPEDV"))
         foundSth = true;
      }
      if (myEnumerator.MoveNext() == false) break;
    }
  }
  catch (Exception e) { }
}
private void CmdTwitterTest_Click(object sender, RoutedEventArgs e) {
  freeStuffRunner();
}
```

Achten Sie bei dieser Vorgehensweise allerdings darauf, nicht zu streng vorzugehen. Erstens können Viel-Twitterer den zurückgelieferten Bereich binnen kürzester Zeit mit Tweets füllen – zweitens kann es sein, dass die Aktualisierung der Timeline des Kontos etwas dauert. Ersteres Problem lässt sich durch Überprüfen des CreatedOn-Dates der angelieferten Tweets umgehen – bei Zweiterem hilft nur asynchrones Warten.

MIT HELPER UND NOTIFARIA

Bei der Arbeit als Entwickler gibt es immer wieder Sachen, die einen grün und blau ärgern. Das Universal Windows Toolkit begegnet diesen Problemen durch eine Vielzahl von Helferklassen, die verschiedene häufig auftretende Ärgernisse entschärfen.

Als erstes Beispiel dafür wollen wir uns dem - insbesondere in Zeiten steigender Internetkosten - lästigen Problem zuwenden, ob eine Internetverbindung Kosten pro Megabyte verursacht oder nicht. Dies lässt sich über die ConnectionHelper-Klasse begegnen, deren Nutzung folgendermaßen aussieht:

```
private void HamburgerMenu_OnItemClick(object sender,
ItemClickEventArgs e)
{
   if (ConnectionHelper.IsInternetAvailable){
       if (ConnectionHelper.IsInternetOnMeteredConnec-
       tion == false) {
         //Aber nun los!

       }
    }
}
```

in Tests des Autos funktionierte die Klasse soweit ohne Probleme- achten Sie darauf, dass die Nutzer im Fall eines WLANs selbst dafür verantwortlich sind, Microsoft darüber zu informieren, wenn bei der Nutzung dieses Funknetzwerks permanent Kosten anfallen.

Ein weiteres nettes Produkt hört auf den Namen Color Helper. Die Klasse ist darauf spezialisiert, Konversionen zwischen verschiedenen Farbbäumen vorzunehmen - das ist insbesondere dann hilfreich, wenn man mit Bilddaten hantiert und dem Nutzer die Möglichkeit geben möchte, Farben punktgenau zu beschreiben.

Eine weitere extrem interessante Klasse ist der DispatcherHelper - er wendet sich dem leidigen Problem zu, dass der GUI-Stack von Windows nur aus dem GUI-Thread heraus ansprechbar ist. Die in der statischen Klasse enthaltenen Helfermethoden nehmen Task-Instanzen entgegen, die sodann – für den Entwickler transparent - am GUI-Thread ausgeführt werden.

Da eine komplette Betrachtung der Helferklassen den Rahmen dieses Artikels sprengen würde, sei auf den Code des Frameworks verwiesen. Analysieren Sie einfach die diversen Dateien im Helpers-Unterordner: er gibt einen kompakten Überblick über den zum Zeitpunkt der Drucklegung vorhandenen Funktionsumfang. Beachten Sie, dass sowohl Dokumentation als auch Beispiel-App in diesem Bereich noch unvollständig sind.

Ein weiterer interessanter Einsatzbereich ist das Hantieren mit Notifications: die mit Windows Phone Sieben erstmals eingeführte Kachelbenutzeroberfläche bietet Entwicklern die Möglichkeit, Informationen über ihr Programm auf eine sehr kompakte Art und Weise darzustellen. Leider ist die technische Implementierung mitunter etwas haarig – das manuelle Zusammenstellen der diversen .xml-Dateien artet insbesondere ob des Fehlens von IntelliSense in Arbeit aus.

Das Universal Windows Toolkit schafft hier mit Wrappern Abhilfe, die sich leichter mit IntelliSense erforschen lassen. Ob des vergleichsweise umfangreichen Codes – die Realisierung eines Tiles ist auch mit dem UWT keine einfache Aufgabe – muss ich mich hier auf eine kurze Besprechung des in LiveTilePage.xaml und dem dazugehörenden CodeBehind-File enthaltenen Beispiels beschränken.

Dreh- und Angelpunkt ist die TileContent-Klasse, die eine Abstraktion für die diversen per XAML anzuliefernden Felder darstellt. Microsoft schreibt hierbei eine TileVisual-Klasse ein, die ihrerseits Platzhalter für die verschiedenen Tilegrößen bereitstellt:

```
public static TileContent GenerateTileContent(string
username, string avatarLogoSource){
   return new TileContent() {
      Visual = new TileVisual(){
         TileMedium = GenerateTileBindingMedium(userna
         me, avatarLogoSource),
         TileWide = GenerateTileBindingWide(username,
         avatarLogoSource),
         TileLarge = GenerateTileBindingLarge(username,
         avatarLogoSource)
```

Das eigentliche Herausschreiben der generierten Inhalte erfolgt dann über die aus dem normalen Framework bekannte SecondaryTile-Klasse, die als Payload den aus tileContent generierten XML-Content bekommt;

```
private async void PinTile()
{
   SecondaryTile tile = new SecondaryTile(DateTime.Now.
   Ticks.ToString())
   {
       DisplayName = "Xbox",
       Arguments = "args"
   };
   tile.VisualElements.Square150x150Logo = Constants.
   Square150x150Logo;
   . . .
   tile.VisualElements.ShowNameOnSquare150x150Logo =
   true;
   . . .

   if (!await tile.RequestCreateAsync())
   {
       return;
   }

   TileUpdateManager.CreateTileUpdaterForSecondaryTile(
   tile.TileId).Update(new TileNotification(_tileContent.
   GetXml()));
}
```

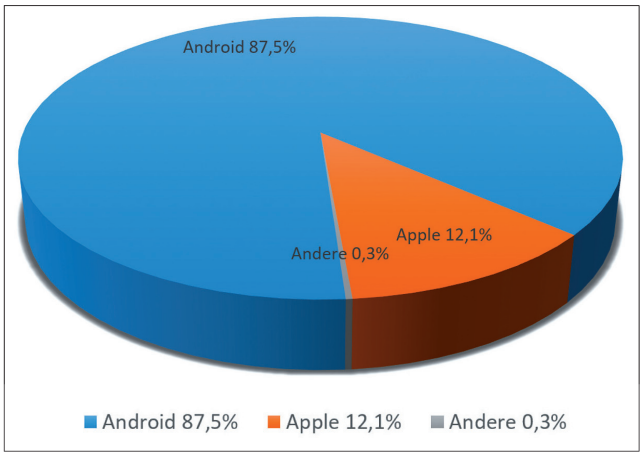
FAZIT

Ein alter Kalauer besagt, dass man einem geschenkten Gaul nicht ins Maul schaut. Im Fall des Universal Windows Toolkit wäre diese Höflichkeitsmaßnahme allerdings gar nicht notwendig: Das Produkt hält, was es verspricht. Wer die in ihm enthaltenen Features nicht kennt, erfindet das Rad neu – eine nur wenig befriedigende Aktivität...

WINDOWS PHONE

Während Microsoft mit seiner aktuellen Strategie im Tablet-Segment scheinbar alles richtigmacht und entsprechend Marktanteile gewinnt, so gilt dies nicht für die Smartphones des Unternehmens. Ganz im Gegenteil: Android und Apple-Geräte bauen ihren Vorsprung weiter aus und verdrängen die anderen Anbieter laut Strategy Analytics im Moment auf einen mageren Marktanteil von 0,3%. Letztes Jahr waren es noch 2,3 %.

Noch aussagekräftiger: Da der Verkauf der von Microsoft hergestellten Smartphones derartig zurückgegangen ist, werden diese bei Strategy Analytics nur noch unter der Kategorie „Andere“, in einer Gruppe mit BlackBerry, Ubuntu und Symbian zusammengefasst. Diese „Anderen“ verkauften anstelle von 8,2 Millionen nur 1,3 Millionen Geräte, ein Einbruch im Vergleich zum Vorjahr von 84,1%, zu dem alleine Microsoft 74% beiträgt.



Marktanteile der Smartphonehersteller

Der Markt an sich bleibt ist zwar stabil, die Markanteile zwischen Googles Android und Apple aber verschieben sich zu Gunsten von Android. Android kann so ein Plus von 10,3% verbuchen, während Apple 5,2% zum Vorjahr verliert. Insgesamt konnten laut der Statistik im 3. Quartal 2016 rund 328 Millionen Smartphones mit dem Android-Betriebssystem und rund 45 Millionen mit iOS verkauft werden.

Woody Oh (Director at Strategy Analytics) ist der Meinung, dass die Plattform für Android-Geräte durch die vielen Hersteller aktuell überfüllt ist und Googles neues Flaggschiff „Pixel“ darüber hinaus die eigenen Hardware-Partner attackiert. Die Frage stellt sich: Ist das eine Chance für Microsoft? Da Mobilfunkkonnektivität in Zukunft mit Sicherheit weiter an Bedeutung gewinnen wird, investiert man auch bei Microsoft weiter in Windows 10 Mobile. Aus Sicht der Endverbraucher und der Entwicklung macht ein dritter Konkurrent durch aus Sinn.

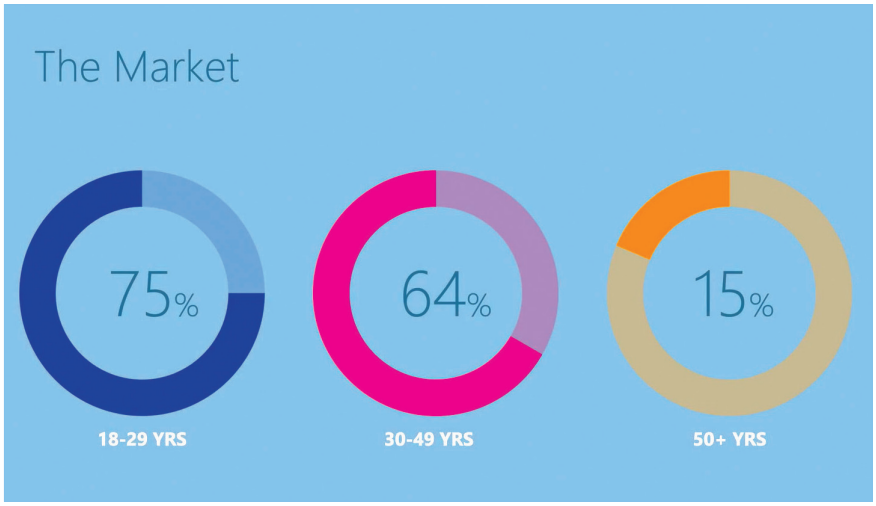
MICROSOFT MACHT SLACK KONKURRENZ

Teams sind heute in vielen Unternehmen keine festen Arbeitsgruppen, die sich regelmäßig treffen. Viel mehr sind es projektbezogene, also wechselnde Besetzungen von Mitarbeitern, die zudem räumlich als auch zeitlich verteilt agieren. Zur Unterstützung der Kollaboration gibt es bereits verschiedene Tools. Seit kurzem gibt es auch eins von Microsoft.

Nach zahlreichen Wochen der Gerüchte hat Microsoft nun offiziell den hauseigenen Slack-Konkurrenten „Microsoft Teams“ angekündigt. Als cloudbasiertes Toolset für Office ist Microsoft Teams ein wichtiges Werkzeug, welches der Zusammenarbeit fördert.

Auf Basis eines Chatraums für Unternehmen, soll es wie auch Slack, Teams die Koordination und Organisation der Arbeit erleichtern. Da ist es eigentlich schon selbstverständlich, dass die Integration zu Skype, Outlook, OneDrive, SharePoint, OneNote, Planner, Power BI und Delve vorliegt, was wiederum einen großen Pluspunkt für Microsoft Teams darstellt. Der Vorteil ist das damit verbundene Handling von Dokumenten. Die Funktionalitäten gehen dabei über die von Slack deutlich hinaus, so können z. B. Videoanrufe getätigt und Meetings organisiert werden.

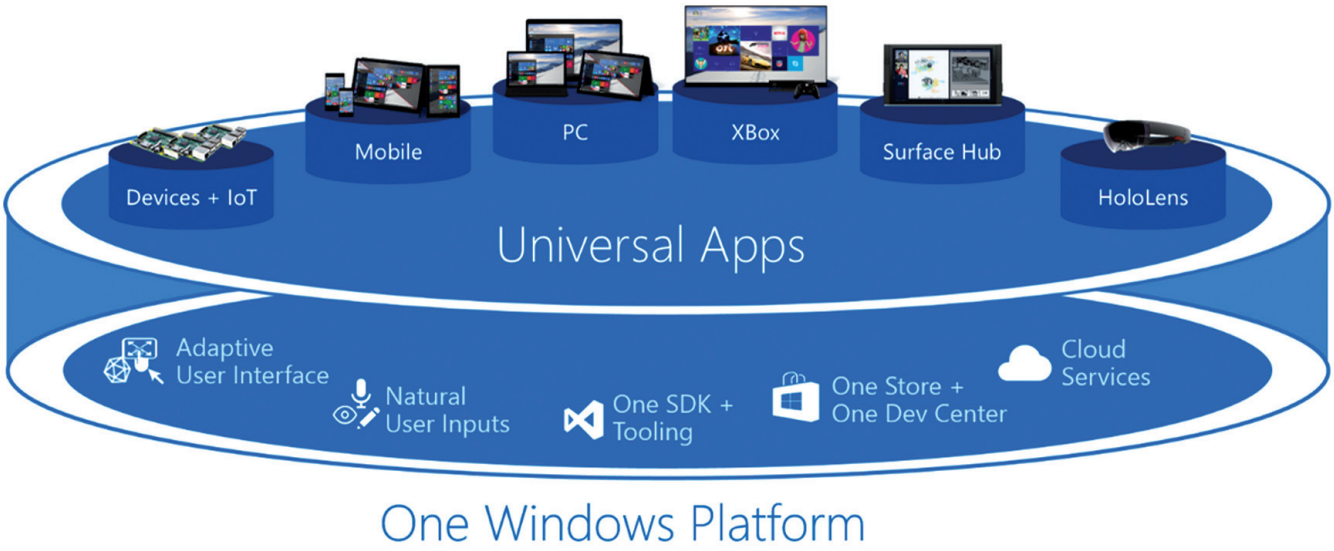
Mit Microsoft Office 365 hat man bereits jetzt schon ein gut geschnürtes Paket, das in vielen Unternehmen zum Einsatz kommt und durch die Erweiterung mit Microsoft Team geht man nun noch einen Schritt weiter. Slack wird es vermutlich sehr schwer haben, mit Microsoft Teams zu konkurrieren. Besonders dann, wenn man bedenkt, dass es bereits standardmäßig im Paket von Microsoft Office 365 enthalten ist und nicht zusätzlich bezahlt werden muss.



Das Preview der Funktionalität konnte man in den vergangenen Wochen testen. Der offizielle Start soll im ersten Quartal dieses Jahres erfolgen. Wer Microsoft Teams aktivieren möchte, muss sich einfach ins Admin Center im Office 365 Portal begeben und die Einstellungen öffnen. Hier nur noch unter Dienste & Add-ins Microsoft Teams auswählen und aktivieren.

WINDOWS DESKTOP EXTENSIONS FÜR UWP

Autor: LARS KRÄMER



Dieser Artikel befasst sich mit der Funktionalität der Windows Extensions für Desktop-Devices, sowie der Integration und Verwendung innerhalb von UWP-Apps. Dabei wird speziell auf die Verwendung in einer UWP in Bezug auf die Benutzeroberfläche eingegangen.

Zunächst erst einmal die Frage: Was sind die Windows Desktop Extensions? Und warum überhaupt Extensions? Nun, kurz und knapp: Mit Windows 10 lässt sich das Betriebssystem bekanntermaßen auf nahezu jeder Form von Gerät nutzen. Von normalen Desktop-Systemen über Tablets, Smartphones und Multimedia-Geräten wie der Xbox One bis hin zu Kleinst, und IoT-Geräte (Stichwort Internet der Dinge). Damit eine App aber wirklich nur eine Code-Basis verwenden muss, wurden Gerätespezifische Methoden und Bibliotheken in die sogenannten Windows

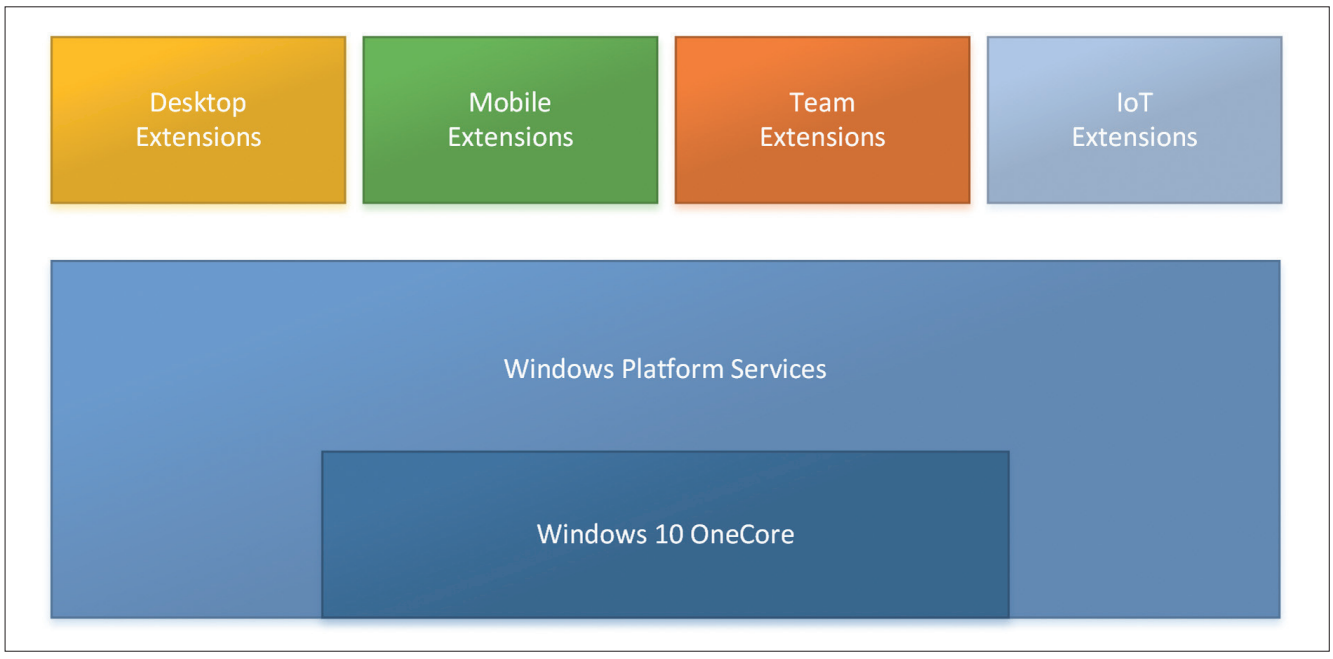


Abbildung 1: Die Grobe Struktur von Windows 10 in Bezug auf UWP

Extensions ausgegliedert (s. Abbildung 1).
So enthalten z.B. die Windows Desktop Extensions alle Klassen, welche sich nur auf einem Desktop-Windows 10 verwenden lassen. Das sind neben diversen weiteren Klassen z.B. die Methoden und Objekte, welche den Zugriff auf die Taskbar ermöglichen. In unserer App können wir damit dann unter anderem auf die JumpListItems oder Badges zugreifen (s. Abbildung 2).

Ein weiteres Beispiel: Die Windows Mobile Extensions beinhalten Objekte, welche z.B. den Zugriff auf die Phone-Statusbar und die Hardware-Buttons (Camera, Back, Search, etc.) regeln. So beinhaltet also jede Windows Extension verschiedene Methoden und Objekte, um den Zugriff auf Gerätespezifische Funktionen zu vereinfachen.

GRUNDLEGENDE VERWENDUNG

Was müssen wir eigentlich beachten? Nun, grundsätzlich nicht viel. Um eine Windows Extension dem App Projekt hinzuzufügen wählen wir in der Projektmappe unter Verweise den Menüpunkt ‚Verweis hinzufügen. Der sich nun öffnende Verweis-Manager sollte bereits bekannt sein. Für UWP-Projekte gibt es nun auf der linken Seite neben Assemblys und Projekte, etc. noch einen weiteren Punkt „Universal Windows“. Dieser beinhaltet sämtliche Extensions und SDK's für ein UWP-Projekt. Unter dem Punkt „Erweiterungen“ werden dann alle verfügbaren Extensions aufgeführt. Neben den Windows Extensions findet man hier auch Extensions für Advertising und den Zugriff auf den App-Store.
Kommen wir direkt zu den Windows Extensions für Desktop. In der Auflistung werden uns zum aktuellen Zeitpunkt nicht nur eine sondern gleich 3 Einträge angezeigt, welche man anhand der Version am Ende der Zeile unterscheiden kann. Der Inhalt der jeweiligen Extension passt zu genau der Windows Version, welche diese Versionsnummer trägt. Grundsätzlich sollte man die aktuellste Version der Extension wählen, abhängig von der Ziel-Windows Version (z.B. Build 10240, 10586 oder 14393) kann man natürlich auch eine bestimmte verwenden.

Abbildung 2: Taskbar-Badges und eine JumpList aus UWP-Apps

Kommen in künftigen Windows 10-Versionen weitere Gerätespezifische Funktionen hinzu, findet man den Zugriff auch nur in der Extension mit der entsprechenden Version (s. Abbildung 3).

Je nachdem auf welcher Version von Windows unsere App nun läuft, kann es vorkommen, dass wir in unserem Code einen Methodenaufruf verwenden, welchen es in einer älteren Version noch nicht gegeben hat. Oder wir verwenden die App gerade auf einem Gerät, welches diese Funktion nicht unterstützt. Es gibt dafür eine einfache Methode zur Überprüfung:

```
if(Windows.Foundation.Metadata.ApiInformation.
IsTypePresent(”Windows.Phone.UI.Input.HardwareButtons“))
{
    Windows.Phone.UI.Input.HardwareButtons.BackPressed
+= HardwareButtons_BackPressed;
}
```

In diesem Fall prüfen wir mit der IsTypePresent()-Methode, ob das aktuelle Gerät die Eigenschaft *„Windows.Phone.UI.Input.HardwareButtons“* besitzt und auf diese zugreifen kann. Ist dies der Fall, schreiben wir uns in das Event ein und unsere Methode wird aufgerufen, wenn die Hardware-Buttons des Gerätes betätigt werden. Bevor man auf Extensions zugreift, sollte in jedem Fall so vorher geprüft werden, ob die jeweilige Funktionalität auch wirklich auf dem aktuellen Gerät vorhanden ist.

WINDOWS DESKTOP EXTENSIONS

Nachdem wir nun wissen wie man die Gerätespezifischen Methoden nutzt, wollen wir einmal in eine App eine Funktionalität aus der Desktop Extension einbauen.

JUMPLIST

Die JumpList werden einige bereits aus dem Windows 7 SDK kennen. Sie ist quasi eine Art Kontextmenü und bietet Verknüpfungen zu bestimmten Funktionen der App oder zu Dateien welche mit der App verknüpft wurden. Innerhalb von UWP-Apps heißt die entsprechende Klasse *„JumpList“* und lässt sich im Namespace *„Windows.UI.StartScreen“* finden.

Zunächst müssen wir prüfen, ob das aktuelle Gerät auf welchem die App ausgeführt wird auch die JumpLists unterstützt. Dazu gibt es eine einfache Methode, welche sich direkt aus der Klasser heraus aufrufen lässt:

```
if (JumpList.IsSupported())
{
    . . .
}
```

Innerhalb dieser if-Abfrage können wir nun gefahrlos unseren Code ausführen. Zunächst laden wir die Instanz der JumpList und entfernen mit der Methode Clear() bereits vorhandene Einträge:

```
JumpList jumpList = await JumpList.LoadCurrentAsync();
jumpList.Items.Clear();
```

Der eigentliche Aufruf zum Erstellen eines JumpListItems ist wie folgt:

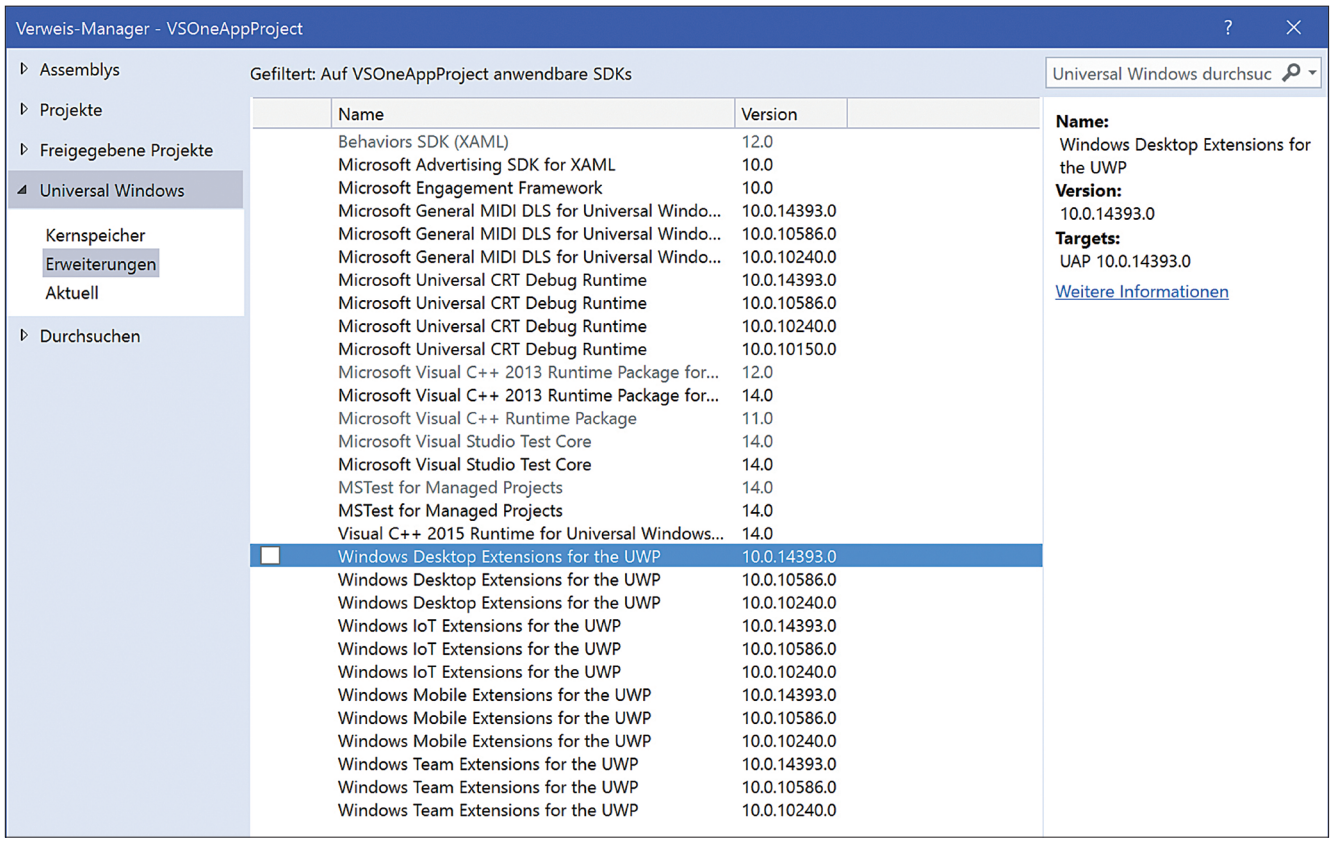


Abbildung 3: Der Verweis-Manager mit den Windows Extensions

```
JumpListItem userItem = JumpListItem.
CreateWithArguments("jumplist_users_overview", "Benut-
zer anzeigen");
userItem.Logo = new Uri("ms-appx:///Assets/Square150x-
150Logo.png");
jumpList.Items.Add(userItem);
```

Mit dem Aufruf JumpListItem.CreateWithArguments(); legen wir eine neue Instanz an und geben an erster Stelle den Wert mit, welche unsere App erhält, wenn ein Benutzer in der JumpList den Eintrag auswählt. An zweiter Stelle wird der Text angegeben der angezeigt wird. Die Klasse bietet außerdem verschiedene weitere Eigenschaften an. Mit diesen können wir die einzelnen Einträge z. B. einer speziellen Gruppe zuweisen oder dem Eintrag eine Grafik hinzufügen, welche neben dem Text angezeigt wird. Zweites verwenden wir bei der obigen Deklaration. Über diese Eigenschaft Logo legen wir den Pfad zu einer beliebigen Grafik fest.

ZUM ABSCHLUSS

Es gibt in den jeweiligen Extensions viel zu entdecken und diese werden in Zukunft mit Sicherheit um das ein oder andere nette und wichtige Feature erweitert. Richtig eingesetzt können die Funktionen einer Extension in vielen Bereichen die App sinnvoll erweitern. Eine Übersicht sämtlicher Möglichkeiten der Universal Windows Platform (Auch außerhalb der Windows Extensions) finden Sie unter [Link01]. Eine Auflistung des gesamten Inhalts der Windows Desktop Extensions finden Sie unter [Link02]. Dort finden Sie nebenbei auch die weiteren Extensions (Mobile, IoT, etc.)

LINKS

- Link01: *https://github.com/Microsoft/Windows-universal-samples* - UWP-App Beispiele
- Link02: *https://msdn.microsoft.com/en-us/library/windows/apps/dn706136.aspx* - UWP Desktop Device Family

NUTZER, WER BIST DU?

Foto: Rainer Sturm/pixelio.de

Wer waren doch die Nutzer? Wenige oder umfassende IT-Kenntnisse? Zwischen 20 und 30 oder doch im Durschnitt über 60 Jahre? Welche Interessen haben Sie? Wird das Angebot eher von Frauen genutzt oder ist die Quote ausgeglichen? Egal? Nicht wirklich! Die Zielgruppe zu kennen ist wichtig! Nur so kann man passgenaue Angebote erstellen. Die Website kann die korrekte Ansprache wählen und das Design der App richtet sich nach den Gewohnheiten und dem Sprachgebrauch ihrer Nutzer. Mit derartigen Maßnahmen kann der Erfolg des Produktes gesteigert werden. Der Beitrag gibt einen kompakten Überblick über die Fragestellungen dieses Themas.

Autoren: Dr. VEIKKO KRYPCZYK und OLENA BOCHKOR

Als IT-Dienstleister werden wir mit der Erstellung von Webseiten und Online-Shops und mit der Programmierung von Apps für die Kunden unserer Auftraggeber beauftragt. Um derartige Aufträge erfolgreich auszuführen, muss man die richtige Kundenansprache wählen. Texte und Design müssen zu den Erwartungen der künftigen Nutzer passen, nur dann wird das digitale Produkt akzeptiert und es gibt eine Change auf Erfolg. In diesem Artikel geht es genau um diese Fragen. Im Einzelnen:

- Welche Kunden interessieren sich für das Angebot?
- Kann man das Produkt einer bestimmten Kategorie zuordnen und daraus bereits erste Schlussfolgerungen für das Design und die textuelle Ansprache ableiten?
- In wie weit lassen sich die allgemeinen Erkenntnisse aus der Marktforschung zur Erstellung von Kundensegmenten, wie zum Beispiel die bekannten Sinus-Milieus, auf den digitalen Bereich übertragen?

Die daraus gewonnenen Erkenntnisse lassen sich grundsätzlich auf die Gestaltung aller digitalen Produkte anwenden. Insbesondere gelten die Aussagen für die

Erstellung von Websites und die Produktion von Apps für mobile Geräte. Beide Produktgruppen vermischen sich an vielen Stellen, z.B. in Form von Web-Apps.

BEISPIELE

Zunächst ein Beispiel aus dem Bereich Webdesign: Über einen Webshop sollen Produkte aus der Kosmetikbrache vertrieben werden. Bisher wurden diese Produkte nur über den stationären Handel angeboten. Wie ist die Seite zu gestalten? Welche Ansprache wählt man? Eher formal oder kann man die Kunden/Nutzer auch persönlich ansprechen? Welches Feeling vermittelt man über das Design, die Farben und das Bildmaterial? Für die Erstellung von mobilen Apps gilt das ebenso, wie das folgende Beispiel zeigt: Eine App soll das digitale Lesen eines Magazins erleichtern. Wie ist die App zu gestalten? Welche Farben und welche Aufmachung wählt man? Kann man bei den Lesern Erfahrungen im Umgang mit Technik voraussetzen, wenn es beispielsweise darum geht, Inhalte für das Offline-Lesen abzulegen?

Diese und ähnliche Fragen laufen alle darauf hinaus, dass man sich vor der Produktentwicklung exakt mit den künftigen Nutzern beschäftigt. Um die Erwartungen besser zu erfüllen, muss man die Nutzer segmentieren und die entstanden Kundengruppen analysieren. Letztendlich können daraus Schlussfolgerungen abgeleitet werden, um die digitalen Produkte genau anzupassen. Auch die moderne Technik der mobilen Geräte kann dabei hilfreich sein. Smartphones verfügen über eine Vielzahl von Sensoren, die auch dazu genutzt werden können, Produkte und Services zu verbessern. Ein oft genutztes Feature ist Geo-Targeting. Wenn ich weiß, wo sich der Nutzer aufhält, dann kann ich diesen mit ortsbezogenen Angeboten überzeugen. Auch hier wieder ein Beispiel: In einer Banking-App gibt es die Option nach Geldautomaten zu suchen. Ist der Kunde jedoch im Ausland, so endet die Suche mit dem wenig hilfreichen Hinweis, dass kein Angebot gefunden wurde. Das ist zunächst einmal korrekt, denn im Ausland sind alle Bargeldabhebungen i.d.R. kostenpflichtig. Mit Geo-Targeting könnte man die Position des Nutzers feststellen und dann alternative kostenpflichtige Angebote vorschlagen.

Name	Anzahl der Nutzer	Kurzbeschreibung
Web Mainstream	18,51 Mio. oder 42%	• eher weiblich und älter • niedrige bis mittlere Bildung • seltene Internet-Nutzung angepasst
Info Seeker	12,59 Mio. oder 29%	• eher älter • höhere Bildung • gehobenes Einkommen • vielseitig interessiert • informationsstarke Internet-Nutzung
Generation Fun	5,01 Mio. oder 11%	• eher männlich und jung • niedrige Bildung • häufige Internet-Nutzung • entertainmentorientiert
Lifestyle Network	3,88 Mio. oder 9%	• eher weiblich und jünger • mittlere Bildung • durchschnittliche Internet-Nutzung • trendbewusst • modeaffin
Web to Go	1,60 Mio. oder 4%	• eher jünger und männlich • häufige Internet-Nutzung • lifestyle- und entertainmentorientiert • vielseitige Computernutzung
Money Community	1,26 Mio. oder 3%	• eher männlich • mittleres bis höheres Alter • hohe Bildung • Einkommensstark • hohe Internet-Nutzung
Web Experts	1,21 Mio. oder 3%	• eher jung und männlich • vielseitige Internet-Nutzung • aktiv • technikaffin

Tabelle 1: Webtypen für Deutschland nach der Studie von Focus Mediale im Überblick [1].

STRATEGIE

Bevor man an sich an die Umsetzung von Maßnahmen macht, sollte über die Strategie nachdenken. Handeln ohne Strategie macht auch in diesem Fall wenig Sinn. Mit Strategie ist vor allem die strategische Kundenausrichtung gemeint. Wie gut kennt man seine eigenen Kunden? Welche Methoden gibt es, um die Kunden besser kennenlernen? Hilfreich ist die Segmentierung in Zielgruppen. Besonders wichtig ist die Absicht des Kunden zur Nutzung der Software zu kennen. Warum und in welcher Situation wird die App gestartet? Welche Informationen möchte der Nutzer erhalten? Wieviel Zeit steht zur Verfügung? Handelt man typischerweise unter Zeitdruck und unter Ablenkung oder kann man eher davon ausgehen, dass der Nutzer in Ruhe vor seinem PC/Notebook sitzt. Für beides haben wir soeben Beispiele genannt. Der Nutzer der Banking-App, welcher einen Geldautomaten sucht, wird dazu i.d.R. wenig Zeit haben und das Smartphone auf der Straße unter allerlei Ablenkungen nutzen. Er ist nur an der Entfernung und dem Weg zum nächsten Automaten interessiert. Auf zusätzliche Informationen verzichtet er gerne. Das andere Extrem ist die potenzielle Kundin, welche in Ruhe sich für ein neues Kosmetikprodukt entscheidet. Hier kann mit zusätzlichen Informationen, einer visuelle Kundenansprache und beispielsweise mit einem Kundenforum gearbeitet werden.

KUNDE – WER BIST DU?

Indem man die eben gestellten Fragen beantwortet, ergibt sich bereits eine grobe Klassifizierung in Prototypen von Zielgruppen. Daraus können anschließend echte Zielgruppen gebildet werden. Die Nutzer werden somit aufgrund ihrer Eigenschaften in so genannte Webtypen eingeteilt. Die Studie Communication Networks von Fokus Medialine definiert sieben Webtypen (siehe Tabelle 1).

Welche dieser Webtypen möchte man mit seiner Webseite ansprechen? Anschließend kann der Kundenwert für jeden Nutzer (eng. Customer Lifetime Value) berechnet werden. Darunter versteht man einen

monetären Wert, welchen der Nutzer der Webseite hat. Bei Online-Shops kann dieser Wert sehr genau bestimmt werden, indem man die getätigten Käufe auswertet. Für andere Webseiten und für Apps ist die Bestimmung dieses Wertes schon deutlich schwieriger, mit hohen Ungenauigkeiten behaftet oder scheinbar nicht möglich. Es gilt: Eine Schätzung bzw. ungefähre Berechnung ist besser als kein Wert.

Eine weitere Möglichkeit der Kundensegmentierung ist die ABC-Kundenanalyse. Dabei werden die Besucher einer Webseite in drei Gruppen aufgeteilt:

- A-Kunden: Diese sind die oberen 20% der Besucher nach dem Kundenwert.
- B-Kunden: Es handelt sich um die unteren 20% der Besucher nach dem Kundenwert.
- C-Kunden: Sind alle anderen Besucher, die nicht der A- und B-Kategorie gehören.

Zu beachten ist, dass die mobile Nutzung des Internets in der letzten Zeit immer mehr zunimmt, d.h. stationäre und mobile Nutzer vermischen sich zunehmend.

PRODUKT-KATEGORIEN – ODER WER BIN ICH?

Auch das eigene Produkt bzw. die Zugehörigkeit zu einer bestimmten Produktgruppe ist für das Erstellen eines passenden Angebotes entscheidend. Am Beispiel einer klassischen Webseite zeigen wir eine Klassifikation. Die kann i.d.R. eine der folgenden Kategorien zugeordnet werden [1]:

- *Persönliche Homepage*: Liefert Informationen über eine Person, dient der Imagepflege und dem Aufbau einer Community.
- *Internetpräsenz für ein Unternehmen:* Marken- und Produktpräsentation, Neukundengewinnung, Nutzer- und Presseinformation.
- *Produktverkauf, zum Beispiel ein Online-Shop:* Umsatzsteigerung, Kundenbindung, Neukundengewinnung.
- *Produktpräsentation:* Marken- und Produktpräsentation, Nutzerinformation und Kundenbindung.
- *Informations- und Nachrichtenportale:* Steigerung der Besucherzahlen, Umsatzsteigerung durch Werbung und kostenpflichtige Inhalte.
- *Social-Media-Websites:* Nutzergenerierung, Aktivitätssteigerung, Vermarktung über Werbeeinnahmen.
- *Entertainment:* Umsatzsteigerung z.B. durch Mitgliedschaft, Vermarktung über Werbeeinnahmen.
- *Webanwendungen:* Nutzerzufriedenheit, Nutzerinformation, Vermarktung über Werbeeinnahmen.

Die zentrale Frage lautet: Welches Ziel wird mit der Webpräsenz primär verfolgt? Es ist wichtig, bereits am Anfang das Ziel klar zu definieren und daraus lang-, mittel- und kurzfristige Maßnahmen abzuleiten. Apps für Smartphones und Tablets können zum Beispiel anhand der App-Store-Kategorien eingeteilt werden (Abbildung 1). Aus der Kombination der Produktkategorie und Kundengruppe ist eine geeignete zielgruppengerechte Ansprache der Kunden zu erarbeiten.

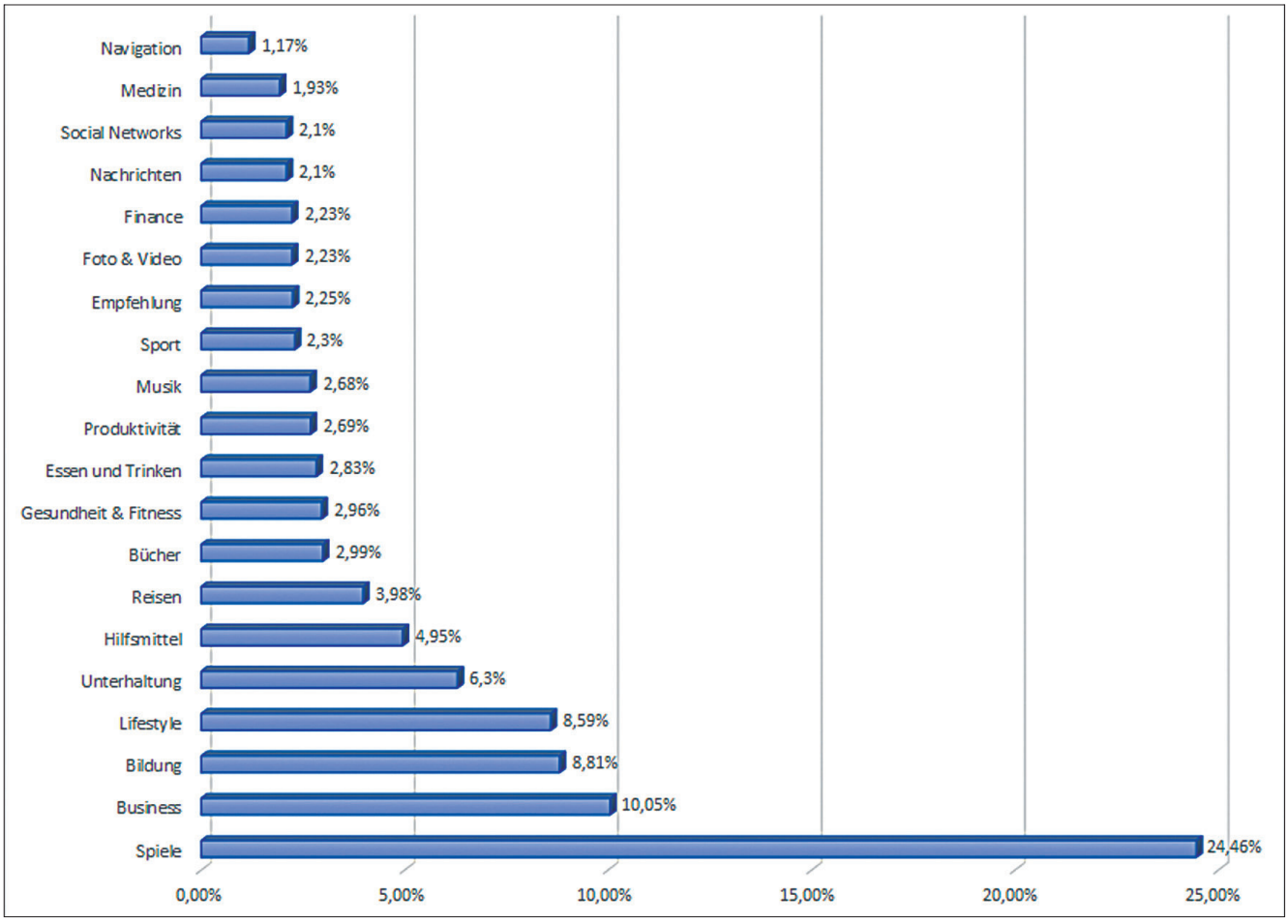


Abbildung 1: Ranking der Top-20-Kategorien im App Store im Oktober 2016 [2]

CONTENT – ZIELGRUPPENGERECHTE AUSRICHTUNG

„Content ist nicht alles – aber ohne Content ist alles nichts“ zitieren wir an dieser Stelle Arthur Schopenhauer. Unter Content werden alle Inhalte einer Webseite verstanden. Der Begriff kann danach unterschieden werden, an welchem Ort, in welcher Form und zu welchem Zweck die Inhalte im Internet erscheinen. Die Unterteilung in folgenden Gruppen ist denkbar:

- *Navigationsinhalte*: z.B. interne Kategorien, Verlinkung
- *Serviceinhalte:* z.B. Kontakt- oder FAQ-Seite
- *Redaktionelle Inhalte:* z.B. Blogbeiträge, Ratgeber
- *Marketing-Content:* z.B. Anzeigetexte, Newsletter
- *Inhalte zum Angebot:* Produkttexte, Kaufempfehlungen
- *User-generated Content:* Bewertungen, Kommentare
- *Social-Media-Inhalte:* Tweets, Facebook-Meldungen
- *SEO-Inhalte:* z.B. Metainformationen wie Page Titel
- *Funktionale Inhalte:* z.B. Fehlermeldungen, Hilfetexte zur Benutzerführung.

Welche Inhalte auf einer Webseite überhaupt und welche Inhalte im Fokus der Darstellung stehen, ergibt sich in erster Linie aus der Produktkategorie. Konkretisieren wir dieses wieder für unsere Beispiele. Bei der Website für den Online-Vertrieb sind neben der reinen Shop-Funktion weitere Inhalte ausdrücklich erwünscht. So können Blog-Beiträge dazu beitragen, die „beste“ Verwendung der „altersvermindernden“ Mittel den potenziellen Kunden näher zu bringen. Eine FAQ-Seite kann den Kaufinteressenten die wichtigsten Fragen zur Zahlung und Lieferung beantworten. Die Banking-App kann ebenso mit einem weitergehenden Informationsangebot aufgewertet werden. Beispielsweise könnte ein integrierter Währungsrechner automatisch die aktuellen Kurse aus dem Internet laden und anhand des aktuellen Standorts des Nutzers (Geo-Targeting) die Umrechnung von Bargeldabhebungen unterstützen. Ebenso sind die Inhalte von der jeweiligen Nutzergruppe abhängig. Eine Social-Media-Integration macht nur dann Sinn, wenn ein Großteil der Nutzer auch darüber erreichbar ist.

ZIELGRUPPEN ERREICHEN

Es ist nur dann möglich eine gewünschte Zielgruppe zu erreichen, wenn man diese bereits definiert und das entsprechende Wissen gesammelt hat. Wie lernt man eine Zielgruppe kennen? Wie sammelt man aussagekräftige und authentische Informationen? Nachfolgend geben wir ein paar Beispiele (Abbildung 2):

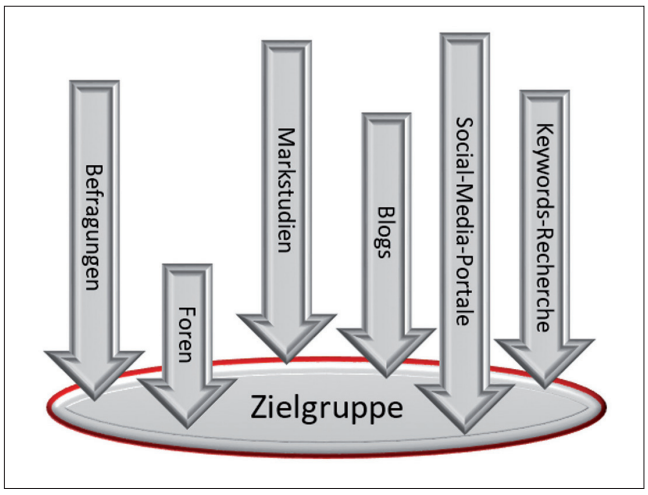


Abbildung 2: Wissen über die Zielgruppe sammeln [1].

- *Marktstudie:* In regelmäßigen Abständen werden von zahlreichen Marktforschungsinstituten Studien zu diesem Thema veröffentlicht. Achtung: Publikationen von Wettbewerbern können ebenso hilfreiche Informationen beinhalten. Also die Augen immer offenhalten!
- *Blogs, Foren und Social-Media-Portale:* Werden Sie in den Blogs und Communities aktiv und finden Sie auf diese Weise heraus, welche Probleme und Themen zum jetzigen Zeitpunkt aktuell sind.
- *Keywords-Recherche:* Google und Co. liefern reichhaltige Informationen über Ihre Zielgruppe.
- *Befragungen:* Eine weitere Option ist es, die Zielgruppe durch Befragungen direkt ansprechen. Auch Online-Befragungen sind dazu geeignet.

Nach dem Sammeln der Informationen ist man noch nicht fertig. Diese sind in kompakter Form aufzubereiten. Im Idealfall erschafft man nach einem solchen Prozess ein klares Bild von der Zielgruppe. Ein erster Ansatzpunkt ist das Nutzungsverhalten des Internets. Wer ist online? Kann ich meine Kunden darüber erreichen? Ein paar Zahlen zeigt der nebenstehende Infokasten „Die Altersstruktur der deutschen Internet-Nutzer“. Eine solch grobe Struktur genügt jedoch noch nicht. Die weitergehende Frage lautet, welche Angebote werden von welcher Nutzergruppe in Anspruch genommen? Wieder ein Bezug zum Beispiel: Zwar sind auch ältere Personen zunehmend im Internet unterwegs, jedoch unterscheidet sich ihr Nutzungsverhalten erheblich. Social Media und Online-Banking werden von dieser Personengruppe weniger genutzt, als von der jüngeren Bevölkerung. Die gewonnenen Ergebnisse sind in Zusammenhang mit dem eigenen digitalen Produkt zu bringen.

ALTERSSTRUKTUR DER DEUTSCHEN INTERNET-NUTZER

Die soziale Struktur der Internet-Nutzer gleicht sich immer mehr der Bevölkerungsstruktur an. Die aktuellen Zahlen [3]:

- 14-19-jährige 93,7%
- 20-29-jährige 87,9%
- 30-39-jährige 79,7%
- 40-49-jährige 69,7%.

Fazit: Die Jugend ist fast komplett im Internet unterwegs. Aber auch ältere Personen sind zunehmend vertreten.

SINUS-MILIEUS

Spricht man über Zielgruppen und eine daraus abgeleitete Produktentwicklung, muss man das Konzept der Sinus-Milieus thematisieren. „Die Sinus-Milieus sind ein Gesellschafts- und Zielgruppenmodell, das Menschen nach ihren Lebensstilen und Werthaltungen gruppiert“ [4]. Im Ergebnis wird ein wirklichkeitstreues Bild der soziokulturellen Vielfalt der Gesellschaft erstellt. Dieses gibt Auskunft über die Befindlichkeiten und Orientierungen der Menschen, über ihre Werte, Lebenseinstellungen und Ziele. Sinus-Milieus helfen die Menschen zu verstehen und herauszufinden was sie bewegt. „Nur wer versteht,

was die Menschen bewegt, kann sie auch bewegen“ [4]. Die Sinus-Milieus betrachten den ganzen Menschen, deren Wertvorstellungen und gruppieren diese nach ihrer Grundhaltung und Lebensweise. Die klassischen Segmentierungsansätze basieren auf sozioökonomischen, demografischen und geografischen Kriterien Abbildung 3 zeigt die Sinus-Milieus Deutschlands für 2016. Es gibt eine Weiterentwicklung der klassischen Sinus-Milieus für unsere Zwecke. Die digitalen Sinus-Milieus beschäftigen sich mit der Ermittlung von Zielgruppen im Internet. Sie sind im digitalen Marketing vielfaltig einsetzbar:

- Traffic-Analysen für Webseiten
- Planung und Optimierung von digitalen Content
- Gezielte Markenbildung im digitalen Umfeld.

Heutzutage ist „online“ ein integrierter Bestandteil der ganzheitlichen Lebenswelt und kann nicht mehr vom restlichen Tagesablauf der Menschen losgelöst betrachtet werden. Mit Sinus-Milieus können die strategischen Gruppen über alle Kanäle verstanden, angesprochen und aktiviert werden. Eine Gegenüberstellung von klassischer Segmentierung und Sinus-Milieu zeigt Abbildung 4. Der Ansatz des Sinus-Institutes (ein deutsches Sozial- und Marktforschungsinstitut) betont die ganzheitliche Betrachtungsweise. Diese muss uns bei der Erstellung digitalen Contents stets bewusst sein.

FEHLER VERMEIDEN

Dieser Abschnitt beschreibt die am häufigsten auftretenden Fehler, die man beispielsweise auf dem Weg zu einer erfolgreichen Webseite machen kann. Die Folge: Die Benutzer finden sich auf der Webseite nicht zurecht und die Zahl der Nutzer sinkt.

- *Fehlende Kenntnisse der Zielgruppe:* Wie bereits erwähnt, hat jede Zielgruppe spezielle Erwartungen an die Webseite. Ein umfassendes Wissen über die Zielgruppe ist deswegen sehr wichtig.
- *Schwierige Schreibweise der Domain:* Man sollte beim Domain-Namen auf eine komplizierte Schreibweise verzichten. An lange Namen erinnert man sich nicht. Ebenso kann das Eintippen des Domain-Namens zum Problem werden. Abkürzungen machen Sinn. Beispielsweise, wenn man www.db.de aufruft, wird man automatisch auf die Webseite der Deutschen Bahn weitergeleitet.
- *Ungeschicktes Navigationskonzept:* Das Auffinden von Informationen fällt den Usern sehr schwer, wenn die Webseite keine eindeutige Struktur hat. Insbesondere bei einem großen Informations- und Produktangebot ist eine klare Einordnung wichtig. Wie man eine Webseite am sinnvollsten strukturiert, ist jedoch sehr individuell. Grundlegend gilt: Eine klare Benennung der Kategorien und keine exotischen Worte bzw. unbekannten Wortkreationen! Holen Sie Feedback von anderen Personen ein!
- *Schlechte Suchfunktion:* Viele Webseiten bieten eine interne Suchfunktion an. Schlechte Suchfunktionen verstehen nicht die Anfrage, liefern keine Ergebnisse oder strukturieren diese nicht. Als Vorbild kann die Funktionsweise von Suchmaschinen gelten.
- *Unverständliche und lange Formulare:* Keiner mag Formulare ausfühlen! Die Regel lautet: Kurze Formulare mit nachvollziehbaren Abfragen. Wichtig ist das Einhalten des Datenschutzes.
- *Suchmaschinenunfreundlichkeit:* Viele Webseite-Betreiber setzten sich zum Ziel die Besucher über Suchmaschinen zu generieren. Wichtig ist es bereits bei der Programmierung auf eine suchmaschinenfreundliche Gestaltung zu achten.
- *Ineffiziente Suchmaschinenwerbung:* Darüber können Unternehmen Werbung auf Ergebnisseiten schalten. Dieses ist schnell und einfach möglich, z.B. Google Ad-Words. Zu vermeiden ist eine unübersichtliche Kampagnenstruktur. Auch hier muss man sich an der

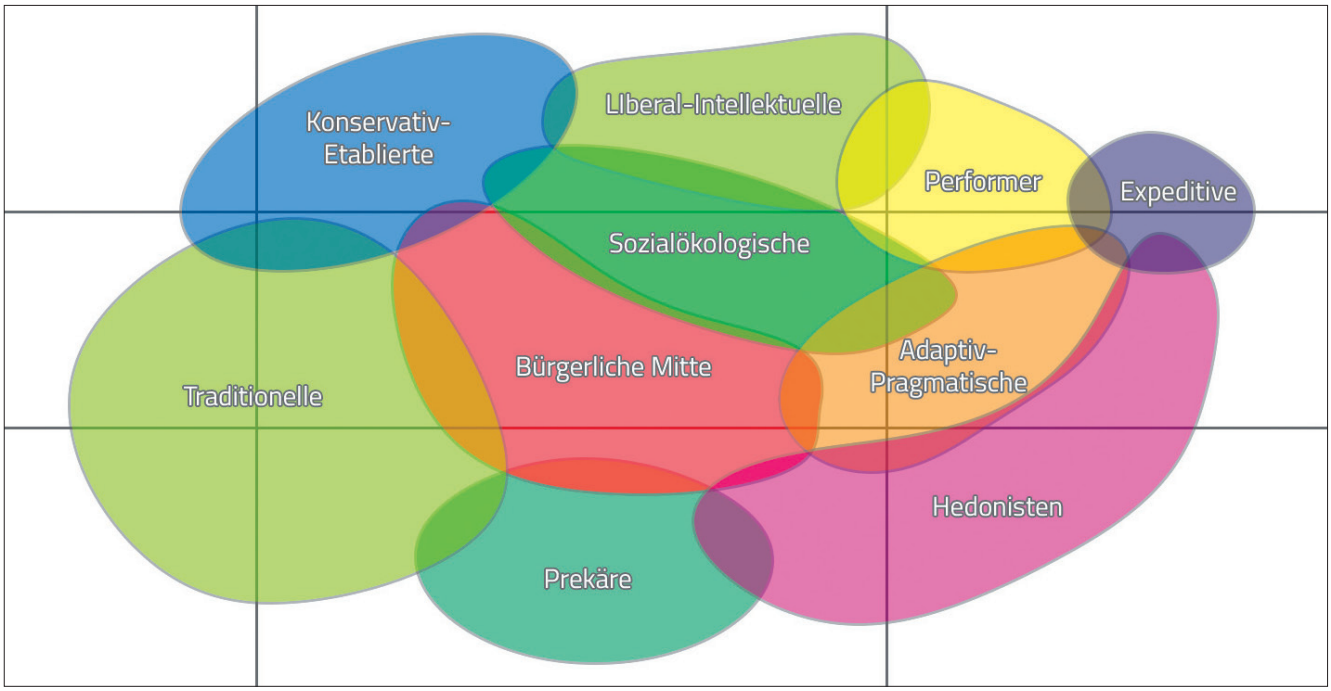


Abbildung 3: Veranschaulichung analog der Sinus-Milieus Zielgruppen-Typologie

relevanten Zielgruppe ausrichten. Zu vermeiden sind sehr viele, sehr wenige und irrelevante Keywords, welche zu nicht passender Werbung führen würden.

- *Fehlerhaftes Bannermarketing:* Das Bannerset muss zum Produktangebot passen. Zu vermeiden sind aufpoppende und blinkenden Werbemittel, sowie die Banner die sich nicht schließen lassen und beim Scrollen mitwandern.
- *Unwirksames E-Mail-Marketing:* Dieses ist einer der meistgenutzten Werbeformen im Internet. Folgende Fehler schrecken den Kunden ab: Newsletter ohne Impressum, kein eindeutiger Absender, keine Angaben in der Betreffzeile, keine Personalisierung und eine unpassende Versandfrequenz.

WEBSEITEN UNTER DER LUPE

Wie erfolgreich bin ich? Anhand von Kennzahlen kann der Erfolg einer Webseite quantifiziert werden. Was sind die wichtigsten Kennzahlen? Page Impression: Seitenaufrufe; Visits: Besuche einer Webseite; Visitors: identifizierte Besucher innerhalb eines bestimmten Zeitraums; Bounce-Rate: Absprungrate; Conversion-Rate: Verhältnis zwischen Besuchern und getätigten Transaktionen und die Click-Through-Rate: die Anzahl der Klicks auf Werbebanner oder Sponsorenlinks im Verhältnis zu den gesamten Impressionen. Web-Analytics-Methoden und -Werkzeuge sind eine große Hilfe. Für diesen Bereich gibt es spezialisierte Software-Anbieter. Das bekannteste Tool ist Google Analytics. Dieses ist kostenlos und unkompliziert in der Bedienung. Wegen Datenschutzbedenken steht dieses Tool in Deutschland jedoch unter Kritik. Vor allem für große Websites wird oft Adobe Analytics genutzt. Als ein Teil der Adobe Marketing Cloud bietet es die Möglichkeit von komplexen Analysen. Diese können im Rahmen eines integrierten Marketing-Automation Systems mit anderen Daten, z.B. der Kampagnenplanung verknüpft werden. Viele Deutsche Websites und Online-Shops setzen auf ein Werbeanalyseunternehmen aus Berlin, Webtrekk. Die Software positioniert sich als User-Centric Analytics-Tool. Das Nutzerverhalten kann damit detailliert analysiert und bewertet werden. Das Web-Analytics-System Piwik ist Google Analytics sehr ähnlich. Es wurde als Open-Source-Projekt aufgebaut.

FAZIT UND AUSBLICK

Einfach ein digitales Produkt zu konzipieren und zu hoffen, dass es bei den Nutzern ankommt, das kann funktionieren, aber wahrscheinlich nicht. Passende Angebote setzen voraus, dass man den Kunden kennt und sich mit ihm beschäftigt. Auf der anderen Seite muss auch das eigene Produkt in den Wettbewerb eingeordnet werden. Die Informatik bedient sich hier dem Wissen der Markt- und Sozialforschung. Einmal mehr ist die technische Umsetzung erst der zweite Schritt. Zuvor gilt es zu beobachten, zu fragen, manchmal auch zu raten und zu analysieren.

LITERATUR UND LINKS

[1] Keßler, E., Rabsch S., Mandic, M.: Erfolgreiche Websites. SEO, SEM, Online-Marketing, Usability, Rheinwerk Computing, 2015

[2] Statista, 2016: https://de.statista.com/statistik/daten/studie/166976/umfrage/beliebteste-kategorien-im-app-store/

[3] http://www.selbstaendig-im-netz.de/2007/04/19/kundengewinnung/alle-zielgruppen-im-internet/

[4] http://www.sinus-institut.de/sinus-loesungen/sinus-milieus-deutschland/

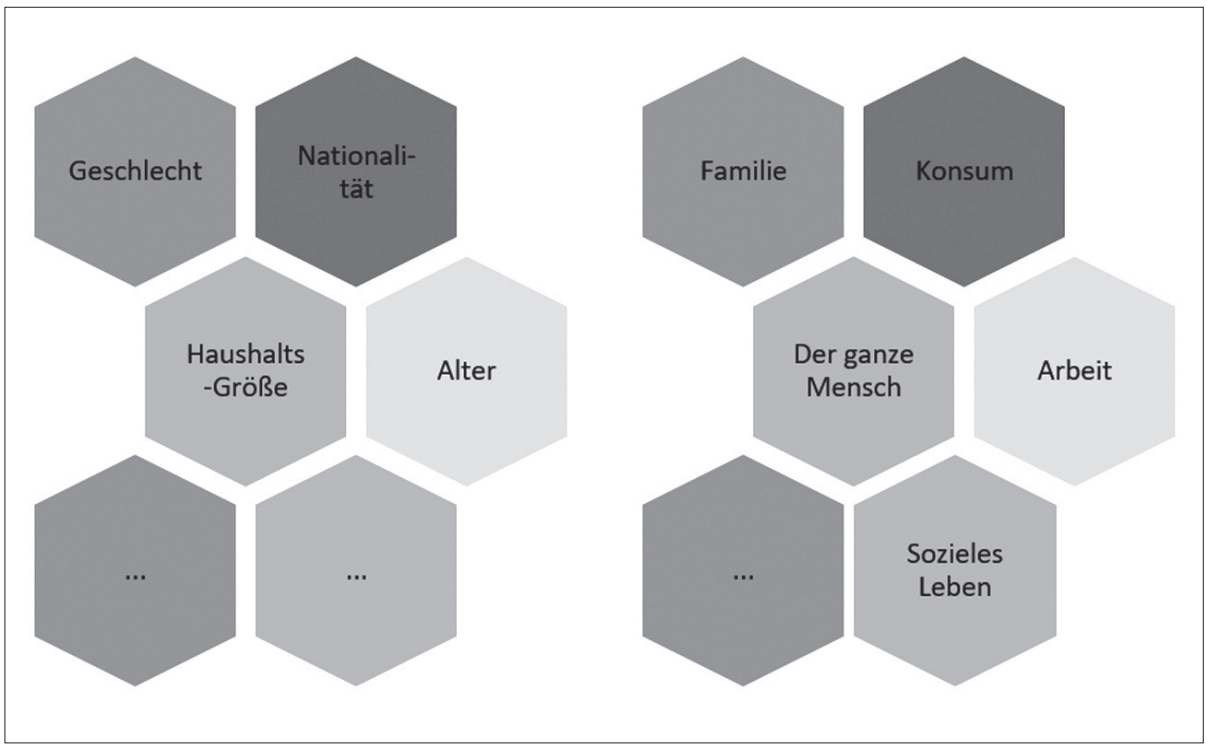


Abbildung 4: Klassische Segmentierungsansätze vs. Sinus-Milieus Betrachtung [4].

BEWEGTE BILDER IM GIF-FORMAT

Kurze Animationen als Unterstützung für schriftliche Erklärungen

Autor: FABIAN DEITELHOFF

Animationen lassen sich hervorragend dazu nutzen, bestimmte Abläufe oder Sachverhalte klar darzustellen. Davon profitieren textuelle Beschreibungen aller Art, wie zum Beispiel Dokumentationen oder FAQ-Bereiche. Mit ScreenToGif können im Handumdrehen kleinere Animationen im Sinne eines Screencasts aufgenommen und im GIF-Format abgespeichert werden. Dadurch ergeben sich zahlreiche Möglichkeiten, sonst langweilig anmutende Textwände aufzulockern.

Text ist im Allgemeinen ein ziemlich gutes Medium, um etwas zu erklären. Es ist sicherlich noch immer die vorherrschende Methode, um Informationen zu übertragen. Selbst in Zeiten von immer besserer Audio- und Video-Übertragungen ist das sicherlich keine Übertreibung.

Wird Text allerdings nicht gedruckt, sondern liegt in digitaler Form vor, bieten sich noch weitere Möglichkeiten, Informationen in einen Beitrag einzubetten. Natürlich stehen die bereits angesprochenen Audio- und Video-Daten wieder zur Verfügung. Allerdings ist das nicht immer der beste Weg. Zum Beispiel bei Online-Artikeln und Blogposts, bei denen Videos häufig als störend empfunden werden. Nicht zu vergessen der Einsatz als Internet Meme [1], um einen Sachverhalt durch eine kurze Animation zusammenzufassen oder zu erklären, häufig auf eine pointierte Art und Weise.

VOM PROBLEM ZUR LÖSUNG

Ein Problem bei eingebetteten Videos ist nämlich, dass sie den Lesefluss und den Informationsfluss unterbrechen. Stellen Sie sich vor, der Artikel würde an dieser Stelle enden, die nächsten 1-2 Seiten wären als Video vorhanden und dann ginge der Text weiter. Eine sehr gruselige Vorstellung. Mit Audio-Daten besteht das nahezu identische Problem.

Es gibt allerdings ein Zwischenformat, dass zwar wie ein Video daherkommt, allerdings nur ein animiertes Bild ist. So lassen sich Teilinformationen, zum Beispiel in einem Tutorial, auf animierte Weise darstellen, ohne dass sich Leser und Leserinnen erst ein minutenlanges Video anschauen müssen. Gerade bei kurzen Beispielen hilft diese Art der Kombination von Informationen durch eine kurze Animation und den zugehörigen Text enorm beim Verständnis.

Programme, mit denen der Bildschirm aufgenommen und in einer GIF-Datei gespeichert wird, erfreuen sich in letzter Zeit immer größerer Beliebtheit. Insbesondere in Blogposts sind sie äußerst beliebt. Der Vorteil ist, dass für moderne Browser diese animierten GIF-Dateien keinerlei Probleme mehr darstellen. Ebenfalls von Vorteil ist der geringe Speicherbedarf und damit die geringere Belastung für die Bandbreite, verglichen mit einem vollwertigen Video.

Der Artikel stellt im Folgenden das sehr beliebte Tool ScreenToGif vor. Am Ende befindet sich zudem eine Übersicht über weitere Programme, die einen ähnlichen Funktionsumfang haben. Die Fülle der Anwendungen macht es leider unmöglich, diese alle vollständig in einem Artikel vorzustellen.

WAS IST SCREENTOGIF?

Die Anwendung ScreenToGif [2] trägt einen sehr markanten Namen, der den Sinn und den Funktionsumfang des Tools fast vollständig beschreibt. Primär ist es damit möglich, den Bildschirm, das Bild einer Webcam oder einen im Programm eingebetteten Zeichenbereich aufzuzeichnen und im zum Beispiel GIF-Format abzuspeichern (siehe Abbildung 1). Als Bonus ist ein Editor integriert, mit dem sich das aufgezeichnete Material direkt in der Anwendung bearbeiten lässt.

ScreenToGif ist auf der einen Seite aufgrund seiner zahlreichen Features - dazu später mehr - so beliebt. Auf der anderen Seite aber auch deshalb, weil die Anwendung komplett kostenfrei ist und als Open Source Projekt im Quelltext zur Verfügung steht. Das Projekt wird auf GitHub in einem Git Repository [3] gehostet und steht unter der MS-PL Lizenz. Neben dem Repository auf GitHub gibt es zusätzlich eins auf CodePlex [4]. Allerdings sind die Beschreibungen dort mit vielen Hinweisen versehen, dass das Projekt nach GitHub umgezogen ist und die Informationen auf CodePlex damit veraltet sind. Entwickelt wird ScreenToGif hauptsächlich von Nicke Manarin [5].

INSTALLATION

Die Installation von ScreenToGif ist nicht der Rede wert, da es keine dedizierte Installation gibt. Heruntergeladen wird das Programm als zip-Archiv in dem sich eine ausführbare Datei befindet. Getestet wurde im Übrigen mit der Version 2.3.2 von Ende November 2016. Das Projekt ist erfreulich aktiv und es erscheinen regelmäßig Updates.

Die so heruntergeladene Datei kann einfach gestartet werden. Voraussetzung ist das .NET Framework 4.6.1. Weitere Abhängigkeiten gibt

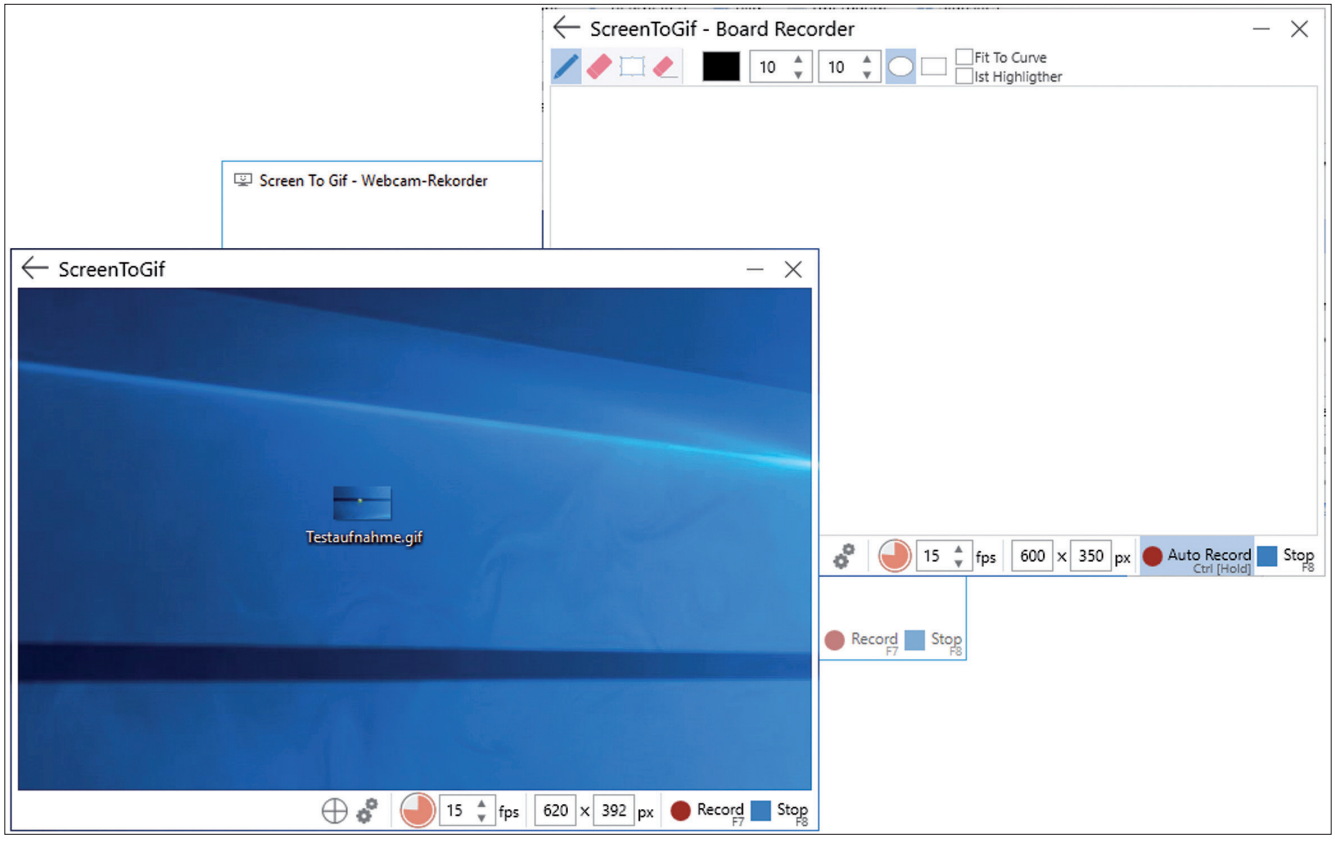


Abbildung 1: Die verschiedenen Hauptfunktionen von ScreenToGif.

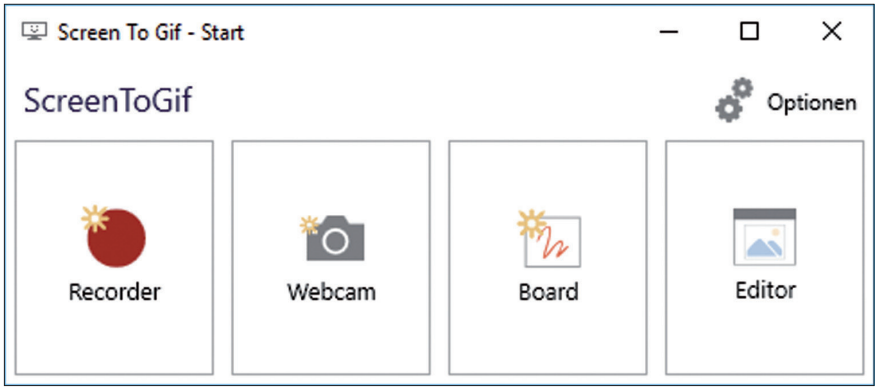


Abbildung 2: Das Startfenster von ScreenToGif.

es nicht. Nach einigen Sekunden erscheint das Startfenster von ScreenToGif, das in Abbildung 2 zu sehen ist. Darüber kann die Aufzeichnung des Bildschirms, der Webcam, des Zeichenbereichs oder der Editor gestartet werden, ebenso wie die Einstellungen der Anwendung.

AUFNEHMEN, BEARBEITEN UND SPEICHERN

Die Hauptfeatures der Anwendung konzentrieren sich auf die bereits genannten Komponenten des Rekorders, der Webcam-Aufzeichnung, des Zeichenbereichs und des Editors. Dieser Artikel konzentriert sich auf den Rekorder als Hauptfunktion. Einmal im Startfenster ausgewählt, ergibt sich das Bild aus Abbildung 3. Die Abbildung zeigt keine optische Täuschung oder etwas in der Art. Der Hauptbereich des Fensters ist durchsichtig, so dass die darunterliegende Anwendung zu sehen ist. Ganz genau betrachtet handelt es sich beim Rekorder-Fenster um den Rahmen, da der mittlere Bereich nicht nur durchsichtig ist, sondern alle Aktionen zum darunterliegenden Fenster durchlässt. Eine sehr interessante und gut durchdachte Funktion, da es so kinderleicht ist, den Bereich auszuwählen, der von ScreenToGif aufgezeichnet werden soll.

Die weitere Handlung zum Aufnehmen ist dann denkbar einfach. Am unteren Bildschirmrand befinden sich häufig notwendige Optionen wie die Frames pro Sekunde, die Größe des aufgenommenen Bereichs und Schaltflächen beziehungsweise deren Tastaturkürzel zum Aufnehmen (F7) und Stoppen (F8) einer Aufnahme.

Läuft eine Aufnahme, ist das ganz deutlich an den durchlaufenden Frames in der Titelleiste von ScreenToGif zu erkennen. Wird die Aufnahme gestoppt, schließt sich das Rekorder-Fenster und der Editor wird mit der gerade durchgeführten Aufnahme geöffnet (siehe Abbildung 4). Das Editorfenster bietet zahlreiche Optionen, um die Aufnahme zu bearbeiten. Ganz konkret können die einzelnen Frames, die am unteren Bildschirmrand zu sehen sind, durchlaufen und nach Belieben verändert werden.

Die im Rahmen dieses Artikels durchgeführte Beispielaufnahme ist zum Beispiel deutlich zu lang. Aufgenommen wurde, wie eine leere Textdatei mit Namen Testdatei.txt über das Kontextmenü vom Desktop eines Windows 10 Systems gelöscht wird. Von den aufgenommenen 162 Frames sind aber etliche redundant, weil sie keine Veränderung im Bild darstellen. Für animierte Bilder, die in Beiträge wie zum Beispiel Tutorials eingebunden werden sollen, ist es allerdings sehr relevant, dass die Animation genau den Kern des Pudels zeigt. Steht die Animation auch

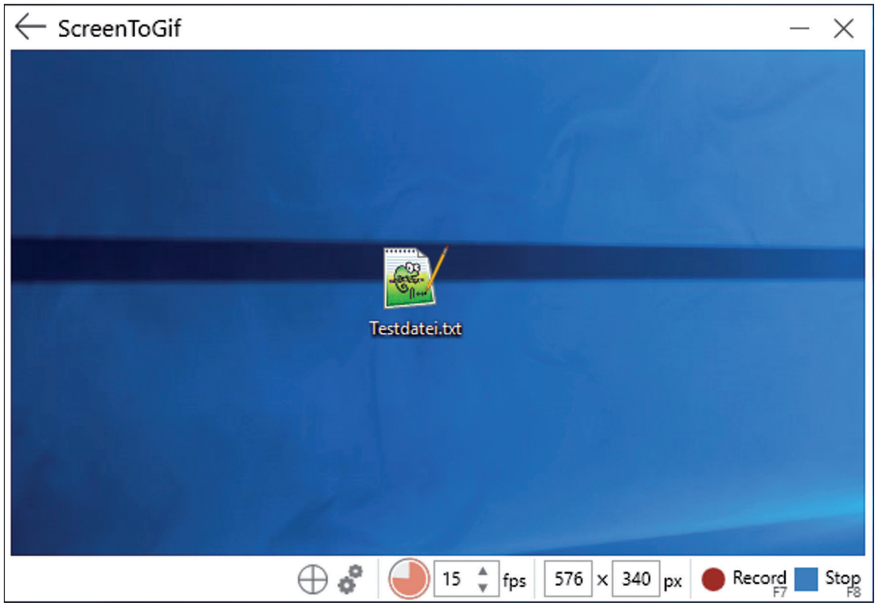


Abbildung 3: Der Rekorder mit dem relevanten Bildausschnitt in der Mitte.

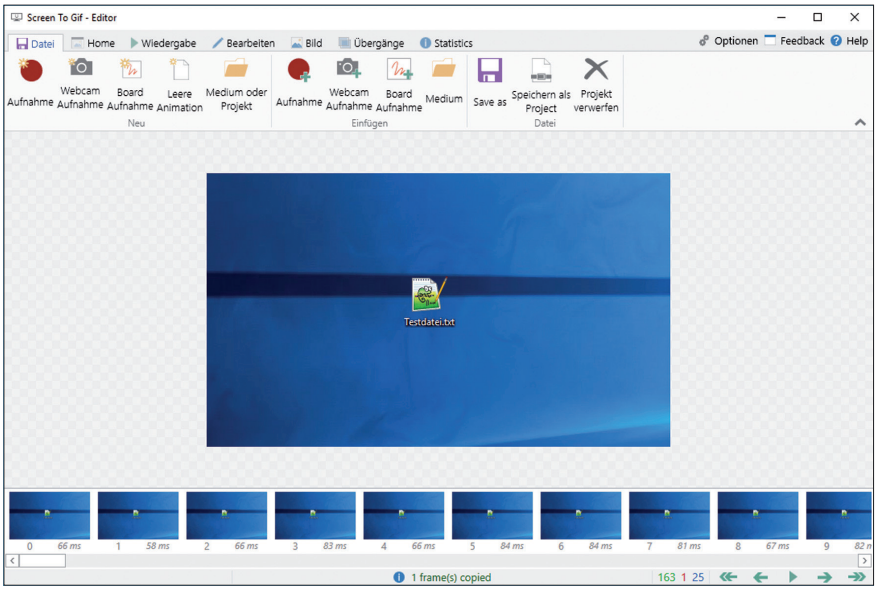


Abbildung 4: Die Testaufzeichnung im Editor.

nur etwas länger still als sie muss, wundert sich der Betrachter eventuell ob etwas nicht richtig ist. Zudem stehen Animationen ganz selten für sich alleine. In der Regel sollen sie eine Textpassage inhaltlich unterstützen. Hierfür ist ebenfalls eine zeitlich kurze Animation von Vorteil, die nur das Wesentliche zeigt. Frames lassen sich ganz einfach über die Auswahl am unteren Bildschirmrand löschen. Frame oder Framebereich auswählen, entfernen drücken und weg sind sie. Auf diese Weise konnte die Beispielaufnahme auf 52 Frames gekürzt werden.

Sind die Bearbeitungen abgeschlossen, kann die Aufnahme als ScreenToGif-Projekt oder als GIF- beziehungsweise Video-Datei abgespeichert werden (siehe Abbildung 5). Dabei stehen unterschiedliche Kodierungsverfahren zur Verfügung. Mit dem Legacy-Encoder ist das Resultat 176 Kilobyte groß und lässt sich mit den üblichen Bildbetrachtern anzeigen.

Weitere Features des Editors sind unter anderem die Möglichkeit, Frames in der Größe zu verändern, zu stutzen oder zu spiegeln. Eigener Text ist ebenso möglich wie freies Zeichnen, das Einfügen von Wasserzeichen oder Fortschrittsbalken. Gerade letztes kann sich auszahlen, da Betrachter der Animationen sofort wissen, wie lange ebendiese Animation noch dauert.

DER BLICK INS REPOSITORY

ScreenToGif ist nahezu vollständig in C# geschrieben. Als Oberflächen-Technologie kommt die Windows Presentation Foundation (WPF) zum Einsatz. Wenn es allerdings darum geht mit Screenshots zu arbeiten, eine Aufzeichnung findet als Reihe von sequentiellen Screenshots statt, wird direkt auf Windows API-Funktionen zurückgegriffen. Wer sich für diese Implementierung interessiert, wird im Bereich des GifRecorders der Anwendung fündig. Genauer gesagt in der Klasse ScreenCapture in der Datei Screenshot.cs [6]. Leider driften Dateinamen und Klassennamen auseinander, was die Suche und Navigation im Repository etwas erschwert. Die in der Klasse vorhandene Methode CaptureWindow erstellt ein Image Objekt von einem spezifischen Fenster. Durch zahlreiche User32 und GDI32 API-Aufrufe wird zunächst ein Zeiger auf ein Bild erzeugt, um das abschließend mit C# Methoden in die verwaltete .NET Welt zu bekommen.

Einen weiteren Blick wert sind die Implementierungen zu den Decodern und Encodern, die sich im Verzeichnis ImageUtil [7] befinden.

WEITERE TOOLS IM ÜBERBLICK

Am Anfang des Artikels wurde es bereits angekündigt, dass ein einzelner Beitrag nicht ausreicht, um alle Programme vorzustellen. ScreenToGif ist nämlich beileibe nicht die einzige Anwendung, die den Bildschirm aufzeichnen und im GIF-Format speichern kann.

Eine weitere Anwendung ist Recordit [8], die ziemlich anders an das Thema herangeht. Der Download ist schnell erledigt und die Installation geht ebenfalls flott über die Bühne. Nach dem Start der Anwendung ist nur ein Symbol in der Info-Leiste von Windows zu sehen. Mit einem Klick wird eine Selektion gestartet, um den Bildschirmbereich auszuwählen, der aufgenommen werden

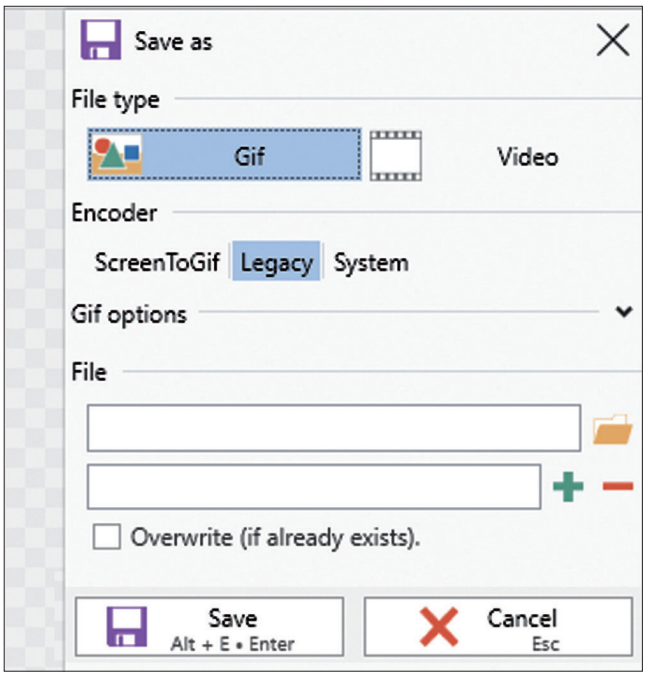


Abbildung 5: Die verschiedenen Möglichkeiten, das aufgezeichnete Material abzuspeichern.

soll. Nach der Aufnahme öffnet sich nicht etwa ein Editor oder eine sonstige Möglichkeit zur Bearbeitung, sondern eine Webseite mit einer eindeutigen URL [9]. Das aufgezeichnete Material wird nämlich direkt hochgeladen und steht dort denjenigen zur Verfügung, die von der URL Kenntnis haben. Auf der einen Seite ist das schön puristisch, auf der anderen Seite muss jeder für sich überlegen, ob der automatische Upload für die Aufnahme überhaupt in Frage kommt.

GifCam [10] ist ein weiteres beliebtes Tool, das sich bei der Handhabung stark an ScreenToGif orientiert. Es kommt als einzelne ausführbare Datei und bringt einen rudimentären Editor mit. Für kleinere Aufgaben ist es gut einsetzbar. Wer ein paar mehr Features benötigt, gerade was den Editor betrifft, sollte direkt zu ScreenToGif greifen.

Und zu guter Letzt noch eine Auflistung all derer Anwendungen, die aus Platzgründen bisher komplett außen vor bleiben mussten:

- Capture Gif
- GIF Brewery 3
- Gif Screen Recorder
- GYAZO
- LICEcap
- Camstudio
- Icecream Screen Recorder
- Gif Recorder

Die Auflistung ließe sich beliebig fortsetzen. Interessant ist, dass es zahlreiche Anwendungen nur für Mac OS X gibt. Ob das daran liegt, dass Mac User häufiger Bedarf an GIF-Animationen haben oder ob es durch den App Store einfacher ist, Anwendungen an den Mann und die Frau zu bringen, ist schwer zu sagen. Glücklicherweise hat sich das in den letzten Jahren geändert, so dass es mittlerweile eine ebenfalls stattliche Zahl von Screen Rekordern für die Windows Plattform gibt.

FAZIT

Ich persönlich bin ein großer Fan von ScreenToGif, was einer der Gründe ist, warum das Tool in diesem Artikel

genauer unter die Lupe genommen wurde. Allerdings ist es auch nur ein Beispiel unter vielen, da zahlreiche andere Anwendungen einen ebenso großen Funktionsumfang bieten und ohne Probleme als Ersatz dienen können.

Allgemein hat sich der Einsatz von kurzen Animationen im GIF-Format mindestens gefühlt drastisch erhöht. Nicht nur als Meme, um einen Sachverhalt auf den Punkt zu bringen. Auch in Blogposts beziehungsweise Online-Texten aller Art kommen die Animationen immer häufiger vor. Wenn sie gut gemacht sind, können sie die Informationen im Text unterstützen und anreichern.

LINKS UND QUELLEN

[1]: Der Begriff „Internet Meme“ in der englischsprachigen Wikipedia: https://en.wikipedia.org/wiki/Internet_meme

[2]: Webseite zu ScreenToGif: http://www.screentogif.com/

[3]: Das Git Repository auf GitHub zu ScreenToGif: https://github.com/NickeManarin/ScreenToGif

[4]: Veraltetes Repository auf CodePlex zu ScreenToGif: https://screentogif.codeplex.com/

[5]: Twitter-Profil von Nicke Manarin: https://twitter.com/NickeManarin/ScreenToGif

[6]: Die Screenshot-Funktionen im Repository zu ScreenToGif: https://github.com/NickeManarin/ScreenToGif/blob/master/GifRecorder/Capture/Screenshot.cs

[7]: Die ImageUtil-Funktionen im Repository zu ScreenToGif: https://github.com/NickeManarin/ScreenToGif/tree/master/ScreenToGif/ImageUtil

[8]: Webseite zu Recordit: http://recordit.co/

[9]: Beispiel-Aufzeichnung in der Cloud von Recordit: http://recordit.co/s4VBP4Dyye

[10]: Webseite zu GifCam: http://blog.bahraniapps.com/gifcam/

NACHHALTIGE SOFTWARE-ENTWICKLUNG NICHT DER REDE WERT

Autor: RALF WESTPHAL

Clean Code im Fachbuchhandel ist kein Thema. Nachhaltige Softwareentwicklung scheint niemanden zu interessieren. Als symptomatischen Beleg dafür nimmt Ralf Westphal die Regalmeter einer Fachbuchhandlung in Hamburg.

Die Buchhandlung wird sich mit dem Sortiment nach dem richten, was nachgefragt wird. Also ist der Inhalt der Regale ein Spiegel des Interesses in der Softwarebranche.

Zu sehen sind um die 2.200 Bücher. Deren Themen reichen von Linux über Office, Minecraft und Hacking bis zu C++, Agilität und Clean Code. Da ist für jeden etwas dabei, sollte man meinen.

Aber wo sind denn die Bücher über Clean Code als ausdrückliche Disziplin für nachhaltige Softwareentwicklung, also solche, die die Anforderung Wandelbarkeit erfüllt? Finden Sie diese? Tipp: Die Bücher sind in Abbildung 1 mit einem grünen Rahmen versehen.

Es sind fünf Bücher links im zweiten Regal ganz oben. Das sind sage und schreibe knapp 2‰ des Gesamtangebots.

Fasst man das Thema etwas weiter und nimmt auch noch Bücher hinzu (rote Kästen), die sich mit einzelnen Praktiken des Clean Code Development auseinandersetzen (z. B. Continuous Integration) oder die Modularisierung als Grundlage für Wandelbarkeit behandeln (z.B. Entwurfsmuster, Softwarearchitektur, Domain Driven Design), dann kommen vielleicht nochmal 20 Bücher dazu. Insgesamt wären es also 25 von 2.200 oder 1,1%. Nur 1% der Nachfrage dreht sich um Nachhaltigkeit!

Und wenn es 2% wären, würde das auch keinen großen Unterschied machen. 98% würden sich immer noch nicht dafür interessieren.

Was sagt das über unsere Branche aus?

Ralf Westphal nimmt an, dass mittlerweile jedem klar sein sollte, dass es ein Nachhaltigkeitsproblem gibt. Der Begriffe dafür gibt es ja viele: brownfield, legacy code, big ball of mud...

Doch guidance sucht man für den Weg aus dieser Situation anscheinend nicht in der Literatur. Bei Minecraft hingegen (12 Bücher) oder Raspberry Pi (13 Bücher) glaubt man, Hilfe zu gebrauchen. Dabei ist beides im Verhältnis zur Nachhaltigkeit trivial. Denn bei beidem ist das Feedback, ob man es richtig oder falsch gemacht hat, unmittelbar. Clean Code hingegen oder eine zukunftsfähige Softwarearchitektur zeigen sich nicht einfach so auf den ersten Blick. Diese herzustellen bedarf auch anderer Gewohnheiten – sonst wäre das Drama ja nicht so groß – und daraus folgend einiger mehr Anleitung und Übung.

Ralf Westphal müsste eigentlich erschüttert sein, aber er hatte es schon geahnt als er in den Laden ging. Das Angebot an Büchern zur Nachhaltigkeit ist mager bis zur Vernachlässigbarkeit. Nicht einmal das vollständige Angebot zu Clean Code oder auch – etwas hipper – Domain Driven Design oder Microservices ist vertreten. Es gibt mehr und auch lesenswerte(re) Bücher dazu. Dito bei der Softwarearchitektur.

Es wird also kaum nachgefragt, so dass sich ein Angebot im Regal nicht lohnt. Noch findet eine Kuratierung der verfügbaren Titel statt. Das Personal im Laden entspricht dem zur Schau gestellten Programm. Fachkompetenz bei den drei müden Gestalten hinterm Tresen gleich Null. Warum es den Laden noch gibt, ist Ralf Westphal ein Rätsel. „Früher war das mal anders. Da gab es noch engagierte Mitarbeiter, die sich um eine gute Auswahl bemüht haben. Wann war das zuletzt? Hm... vor 10 Jahren?“: resümiert Ralf Westphal. Er mag zwar Buchläden sehr gern, aber in dieser Form findet er sie eher gruselig. Doch viel wichtiger ist aber, was die Auswahl über unsere Branche aussagt. Das real existierende Interesse liegt immer noch (oder schon wieder?) nicht bei der Nachhaltigkeit. Technology rulez!

Westphal hat nicht die Hoffnung, dass sich die vorherrschende Praxis hin zu sauberer(er) Entwicklung alsbald ändern wird. „Microservices & Docker sind keine Rettungsboote für die Flucht aus dem legacy code für die Masse. Saubere Modularisierung, sauberer Entwurf diesseits aufwändiger Technologien jedoch, die könnten helfen. Nur muss man das wollen. Man muss sich darauf einlassen, Denken und Gewohnheiten zu verändern. Das kostet mehr Zeit. Aber es lohnt auch langfristig mehr, als die schnelle Fahrt in das nächste Komplexitätsdesaster“: erklärt Ralph Westphal.

Abbildung 1: Bücherregal in einer Buchhandlung

Foto: Nadine Papenfuß

VERWIRRENDE VIELFALT

Der Kunde wünscht eine Anwendung für den PC oder eine App zur mobilen Nutzung. So weit so gut und dennoch nicht ganz eindeutig. In allen Bereichen hat die Vielfalt zugenommen. Neben unterschiedlichen Betriebssystemen stehen auch mehrere Arten von Software zur Auswahl. Windows kennt Desktop-Applikationen und Universal Apps. Im mobilen Bereich dominieren dagegen Android und iOS. Jedes System verlangt nach einem unterschiedlichen Programmieransatz. Wir finden es ist Zeit für einen kompakten Überblick, um im Chaos die Übersicht zu behalten.

Autoren: Dr. VEIKKO KRYPCZYK und OLENA BOCHKOR

Früher war die Welt aus Sicht eines Entwicklers noch weitgehend in Ordnung. Professionelle Software im Unternehmensumfeld wurde fast ausschließlich für Versionen des Betriebssystems Microsoft Windows programmiert. Natürlich gab es auch schon Mac OS und Linux-Distributionen, aber Anwendungen für Windows haben den Markt dominiert. Auch im Jahr 2016 laufen auf vielen Rechnern Betriebssysteme aus dem Hause Microsoft (siehe Abbildung 1). Interessant ist jedoch der Umstand, dass neben dem aktuellen System Windows 10 auch noch Windows 7 mit erheblichen Marktanteilen präsent ist. Sogar Windows XP, dessen Support bereits seit einiger Zeit endete, hält Unternehmen und Privatanwender offenbar nicht von ab, es weiterhin einzusetzen. Auch Mac OS ist mit nennenswerten Anteilen vertreten. Die Anteile der restlichen Systeme entfallen u.a. auf Linux-Distributionen. Oft hört man Kritik an dieser doch recht einseitigen Verteilung. Mangelende Konkurrenz führt langfristig zu ausbleibenden Innovationen. Im Bereich des Mobile Computing sieht die Verteilung anders aus. Android und iOS dominieren den Markt. Unterschiede gibt es zwischen den USA und Europa (Abbildung 2). Windows Mobile (Windows Phone) und andere Systeme liegen inzwischen weit abgeschlagen dahinter.

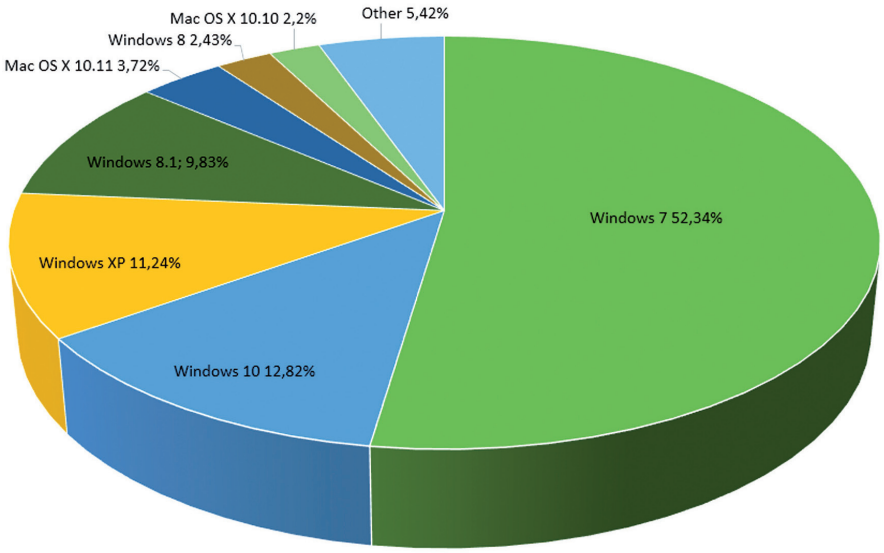


Abbildung 1: Marktanteile der Desktop-Betriebssysteme [1].

Diese Trennung zwischen Mobile und Desktop ist aber nicht ausreichend für eine Klassifizierung. Gerade Microsoft Windows trägt aktuell dazu bei, dass eine Vermischung und damit einhergehend gewisse Verwirrung stattfindet. Windows 10 wird damit beworben, auf nahezu allen Geräten und Geräteklassen lauffähig zu sein. Was am Anfang zunächst wie ein Marketingversprechen aussah, wird mit der Zeit langsam Wirklichkeit. Windows 10 ist nicht nur der Nachfolger von Windows 7 (8.*), sondern im Mobile-Bereich auch von Windows Phone. Auf Smartphones wird es als Windows Mobile 10 bezeichnet. Auf PCs und Notebooks sind primär klassische Desktop-Applikationen im Einsatz. Die Gerätevielfalt reicht jedoch vom Smartphone bis zum IoT-Device auf der einen Seite und von der XBox bis zu den HoloLens (Augmented-Reality-Brille) auf der anderen Seite (Abbildung 3). Möglich macht es die so genannte Universal Windows Plattform (UWP). Sie stellt eine Weiterentwicklung der mit Windows 8 eingeführten Windows Runtime dar. Basierend auf der UWP ist es möglich, eine Anwendung mit ein- und derselben Codebasis für unterschiedliche Klassen von Zielgeräten zu entwickeln. Diese universellen Anwendungen werden als Windows Universal Apps (UWA) bezeichnet. Interessanterweise laufen diese auch auf speziellen Version von Windows 10 für Kleinstrechner der Typen Raspberry Pi und Co. Die zugehörige Betriebssystemversion wird als Windows 10 IoT (Internet of Things) bezeichnet. Sie steht kostenfrei zur Verfügung, hat keine eigene grafische Benutzeroberfläche, wie man es von den anderen Windows-Versionen her kennt. Auf Windows 10 IoT kann man jeweils nur eine UWA im Vollbildmodus ausführen.

Diese Vielfalt ist aus Anwendersicht durchaus positiv und sorgt für eine Integration der Plattformen. Der Nutzer hat zum eine Auswahl zwischen den Systemen und zum anderen einen Wiederkennungswert. Eine App auf dem Desktop bietet eine vergleichbare Funktionalität und ein ähnliches User-Interface, wie auf dem Smartphone. Betrachten Sie dazu beispielsweise den Internetbrowser Edge von Windows 10 als App auf einem Desktop-PC und unter Windows Mobile 10. Leugnen kann man jedoch nicht die Tatsache eines Nebeneinanders von UWA und klassischer Desktop-Software. Der Großteil der Anwendungen liegt traditionell noch in Form klassischer Desktop-Applikationen vor. Einige Programme gibt es in zwei Ausführungen. Wie sinnvoll es ist, diese Doppelstruktur auf Dauer zu pflegen, wird sich zeigen. Bei der Neukonzeption eines Programms ist die Auswahlentscheidung zwischen UWP und klassischer Desktop-Oberfläche nicht leicht. Das Argument, dass die UWP zu stark in Bezug auf die notwendigen Systemressourcen eingeschränkt sind, löst sich mit der stetigen Weiterentwicklung zunehmend auf.

Ist diese Vielfalt für die Nutzer schon ziemlich komplex, wenn man Entscheidungen für ein künftiges Softwaresystem treffen soll, so ist die Komplexität aus Entwickler-Perspektive noch größer. Für jedes Betriebssystem und jeden Anwendungstyp stehen ein eigenes Entwicklungskonzept zur Verfügung. Zum Einsatz kommen unterschiedliche Programmieransätze, Vorgehensweisen bei der Erstellung der Benutzeroberflächen, unterschiedliche Frameworks und Klassenbibliotheken. Ebenso kommen unterschiedliche integrierte Entwicklungsumgebungen (IDEs) und Tools zum Einsatz. Einen Ausweg aus diesem Dilemma verspricht die plattformübergreifende Programmentwicklung. Brauchbare Ansätze kommen gerade auf den Markt, aber ausgreift sind sie noch nicht. In den

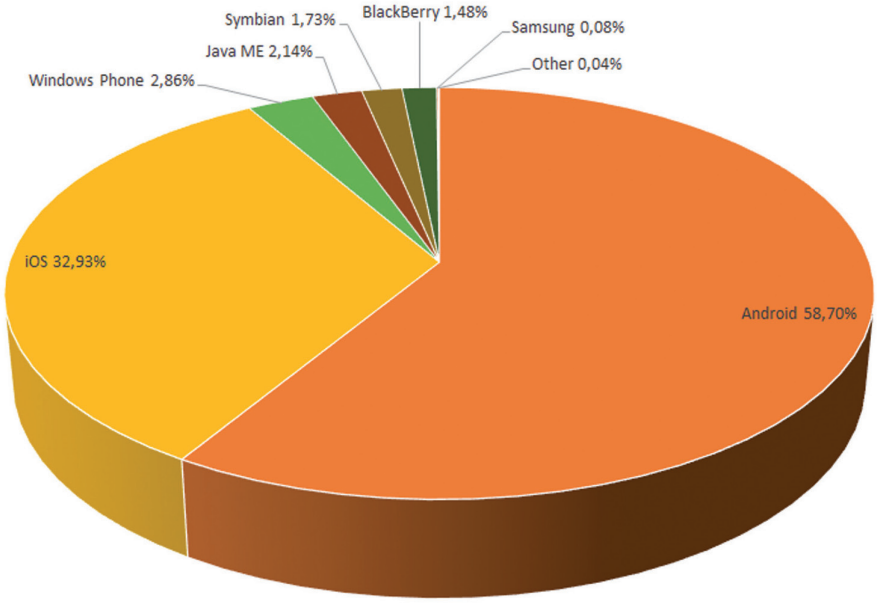


Abbildung 2: Marktanteile der Betriebssysteme für mobile Geräte [2].

Abbildung 3: Windows agiert als Multi-Device-Betriebssystem [3].

kommenden Textabschnitten systematisieren wir die Optionen der Softwareentwicklung, sortiert nach Plattform und Anwendungstyp. Es folgt ein Abschnitt über die plattformübergreifende Programmierung am Beispiel von Apps für mobile Geräte. Am Ende wagen wir eine Prognose, um als Entwickler im Dschungel der Möglichkeiten und Varianten nicht ganz den Halt zu verlieren.

APPLICATIONS FOR WINDOWS

Betrachten wir die Optionen zur Programmierung von Software, welche auf einem stationären Windows 10-Rechner (Desktop/Notebook) läuft. Als Leitfaden für die nachfolgende Beschreibung dient Abbildung 4. Desktop-Applikationen, welche auch auf älteren Versionen von Windows lauffähig sind. Diese werden auf der Basis des .Net-Frameworks programmiert. Aktuell ist .Net 4.6. Als Programmiersprache kommen C# oder alternativ Visual Basic for .Net zum Einsatz. Für spezielle Aufgaben, zum Beispiel für umfangreiche Berechnungen, kann auch die primär funktionale Sprache F# eingesetzt werden. Das UI wird deklarativ mit Hilfe der Beschreibungssprache XAML erstellt. Der UI-Aufsatz heißt Windows Presentation Foundation (WPF). Windows Forms steht auch noch zur Verfügung, hat aber nur noch für die Pflege von Altprojekten seine Berechtigung. Auch mit Hilfe von C++ und einer Klassenbibliothek können Windows-Anwendungen erstellt werde. Für Business-Applikationen ist dieser Weg aber nicht mehr die bevorzugte Option.

Apps der UWP laufen auch auf den „großen" Windows 10-Rechnern. Diese basieren technisch auf .Net Core und haben im Gegensatz zu den eben beschriebenen Desktop-Applikationen keine verwaltetet Codebasis, d.h. sie greifen auf einem nativen Kern des Betriebssystems zurück. C# ist auch hier die primäre Programmiersprache. Das UI wird ebenfalls in XAML definiert. Dieser XAML-Dialekt unterscheidet sich jedoch an einigen Stellen vom XAML für WPF. Der Umstieg von Windows-Desktop-Anwendungen auf Apps für die UWP erfordert nur geringere Anpassungen. Die Vorgehensweisen sind die gleichen, Unterschiede gibt es bei den Namensgebungen der Klassen und Eigenschaften. Das eine oder andere Konzept stehen nur auf der .Net bzw. .Net Core zur Verfügung.

ASP.Net und ASP.Net Core sind die jeweiligen Optionen zum Bau von Web-Anwendungen auf Basis der .Net-Technologie. Web-Anwendungen sind hier jedoch nicht Gegenstand der Betrachtungen, da sich diese in einem anderen Umfeld bewegen. Dazu müsste man das gesamte Spektrum der Open Source-Technologien dynamischer Websprachen, wie beispielsweise PHP, einbeziehen. Ebenso ist hinzuzufügen, dass die eben beschriebenen Optionen zur Erstellung der Anwendungen und Apps nicht abschließend sind. Es wurden die primär angewendeten Varianten vorgestellt. Für Spiele wird beispielsweise weiterhin gegen die Schnittstelle Direct X programmiert.

APPS FÜR MOBILE

Gemeint sind native Apps, welche sich durch ihre direkte Systemintegration und durch einen uneingeschränkten Hardwarezugriff von Web-Apps bzw. hybriden-Apps unterscheiden. Jeder Hersteller gibt dabei den Entwicklern einen eignen Weg zur Entwicklung vor:

- *Android:* Android basiert auf Linux, ist Open Source und die Apps können daher unter allen gängigen Betriebssystemen programmiert werden. Als Programmiersprache wird Java eingesetzt. Die Benutzeroberfläche wird teils deklarativ in XML und teils in Java erstellt. Als IDE kommt Android Studio zum Einsatz, welches Eclipse mit zusätzlichem Plug-In, abgelöst hat.
- *iOS:* Apps für die mobilen Geräte von Apple können letztendlich nur auf einem Mac-Computer erstellt werden, d.h. für das finale Build muss die IDE Xcode benutzt werden. Auch ein Emulator läuft nur auf einem Mac-PC. Die aktuelle Programmiersprache ist Swift, welche die schon etwas in die Jahre gekommene Sprache Objective-C abgelöst hat. Um vorhanden Quellcode zu warten können jedoch beide Sprachen eingesetzt werden.
- *Windows-Apps:* Wir sprechen hier von den Apps für die UWP, welche auch unter Windows 10 Mobile laufen. Diese Version hat auf den Smartphone Windows Phone abgelöst. Entwickelt wird in der Regel mit C# als Programmiersprache. Das UI wird ebenso XAML definiert.

Die IDE ist Visual Studio. Die Apps können nur auf einem Windows 10-Rechner erstellt werden.

Dieses ist sind die typischen Vorgehensweisen für die Erstellung von nativen Apps für die gängigen Betriebssysteme der mobilen Geräte. An der Aufzählung wird bereits deutlich, dass zwischen den Systemen deutliche Unterschiede vorliegen. Zwar kann grundsätzlich mit allen drei Programmiersprachen (Java, Swift, C#) objektorientiert programmiert werden, aber Syntax und Vorgehensweise unterscheiden sich in vielen Punkten. Beim UI gibt es nicht nur Unterschiede bei der technischen Umsetzung, sondern auch die eigentliche Gestaltung der Oberfläche weicht ab. Während Nutzer sich relativ schnell auf allen Systemen zurechtfinden, ist es für den Entwickler eine echte Herausforderung, Benutzeroberflächen für alle drei Systeme zu entwerfen. Sie sollen zum einem einen Wiedererkennungseffekt für die App bieten, aber auch gleichzeitig das typische Feeling des betreffenden Systems umsetzen. Typische Schwierigkeiten entstehen dadurch, dass manche UI-Konzepte nur unter einem System existieren.

Will man native Apps für alle o.g. Betriebssysteme der mobilen Geräte realisieren, muss man auf die von Herstellern vorgesehenen Entwicklungsvorgaben zurückgreifen. Jede App ist eigenständig zu entwickeln. Natürlich kann die Logik der Businessschicht nach Transformation zwischen den Programmiersprachen weiterverwendet werden, aber das UI und dessen Anbindung an die Businesslogik müssen vollständig neu umgesetzt werden. Zusammengefasst: Eine App für zwei oder drei unterschiedliche Betriebssysteme erzeugt einen fast doppelten bzw. dreifachen Aufwand. Hinzu kommt der Umstand, dass man als Entwickler i.d.R. nicht simultan auf allen Systemen gleichermaßen fit ist, d.h. man muss sich immer wieder neu einarbeiten. Bis man eine produktive Arbeitsweise auf dem jeweiligen System erreicht hat, dauert es.

Besserung versprechen plattformübergreifende Ansätze. Lange war hier kein wirklich umfassendes Konzept in Sicht. Nunmehr scheint sich langsam eine Lösung aufzutun. Auf der Basis des Mono-Projektes können Apps für die Systeme Android, iOS und Windows Mobile/Phone nahezu simultan erstellt werden. Als Programmiersprache fungiert C#. Das Unternehmen Xamarin wurde von Microsoft gekauft und das Toolset unter Windows in Visual Studio integriert. Für iOS steht mit Xamarin-Studio eine eigene IDE zur Verfügung. Eine Beta von Visual Studio für MacOS ist auch schon verfügbar. Xamarin bietet zwei Optionen plattformübergreifende Apps zu erstellen. Im ersten Ansatz wird die Fachlogik der Anwendung übergreifend in C# erstellt und das UI basiert auf dem jeweiligen System. Damit benötigt man für die Programmierung weiterhin recht umfassende Kenntnisse über alle Zielsysteme und man muss das UI jeweils separat anpassen. Dennoch: Für die Fachlogik verkürzt es den Programmieraufwand erheblich. Der zweite Ansatz basiert auf der Verwendung von Xamarin-Forms, welches eine übergreifende UI-Entwicklung für alle Systeme mit Hilfe eines XAML-Dialektes erlaubt. Wichtig dabei: Es werden native Apps erstellt. Dazu generiert Xamarin aus dem XAML-Code zur Beschreibung des UI, das jeweilige native Interface für das betreffende System. Dennoch gibt es natürlich auch Grenzen dieses Ansatzes:

- *Systemspezifik:* Sehr spezifische Funktionen müssen weiterhin direkt für jedes System programmiert werden.
- *Technische Voraussetzungen:* Für die Entwicklung von iOS-Apps wird ein Mac benötigt. Auch Windows-Apps können nur unter Windows programmiert werden.
- *Einarbeitung:* Eine Einarbeitung in alle drei Systeme ist trotz plattformübergreifender Programmierung zwingend.

Fazit: Der Weg von Xamarin, heute Microsoft, ist richtig. Es besteht die Change, dass man als Entwickler mit vertretbaren Aufwand Apps für alle großen Systeme simultan bauen kann. Die noch vorhandenen Unstimmigkeiten dieses Entwicklungsansatzes werden mit jeder Version weniger. Das macht uns Hoffnung. Als traditionelle Windows-Entwickler stimmt uns gerade der Umstand

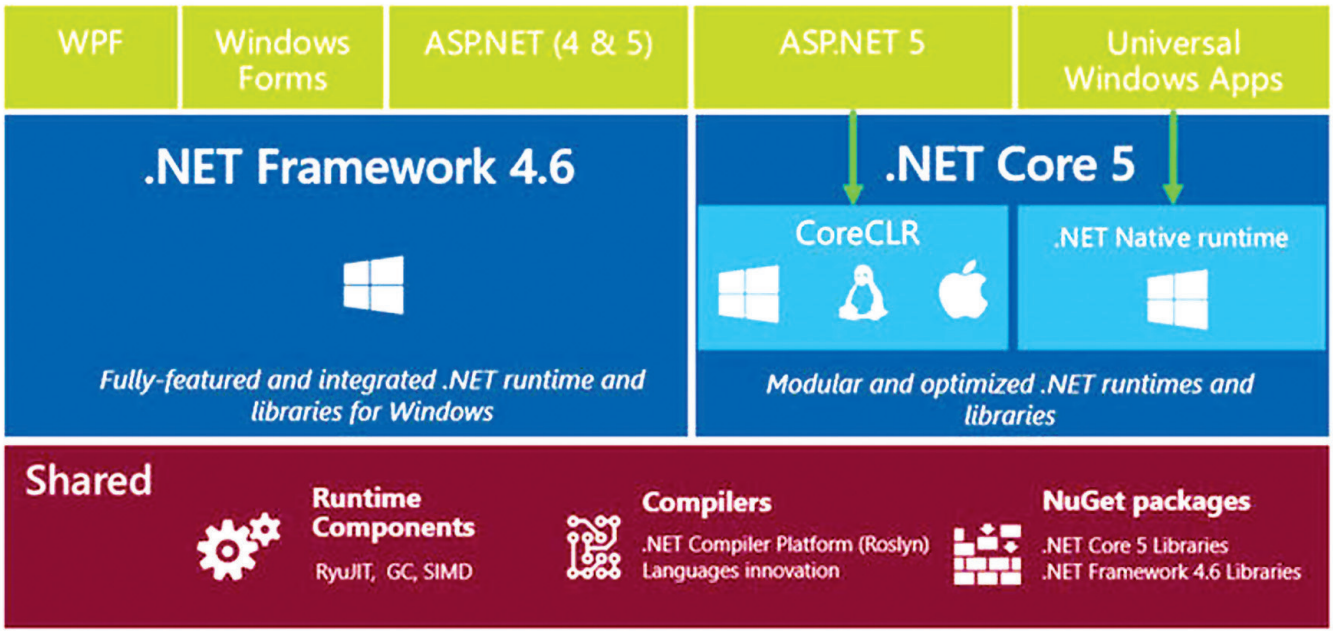


Abbildung 4: Windows 10 – Platform and Tools [4]

hoffungsvoll, dass der Ansatz auf der Basis uns bekannter Technologien basiert. Waren wir doch von der Entwicklung mobiler Apps schon etwas abgehängt, so verspüren wir nun wieder deutlichen Aufwind!

ANDERE SYSTEME

Damit sind hier GUI-Applikationen für MacOS und Linux-Distributionen gemeint. Für MacOS tragen auch diese Desktop-Anwendungen den Namen Apps und können über den systemeigenen Store publiziert werden. Sie werden ebenfalls in Swift mittels Xcode erstellt. Die Erstellung von Apps für MacOS und iOS ist einander ähnlich. Unter Linux wird primär C++ eingesetzt. Dabei wird gegen spezielle UI-Bibliotheken programmiert. Typische IDEs sind KDevelop (KDE) und Builder (Gnome). Plattformübergreifende GUI-Bibliotheken wie QT und Gtk sind unter Linux üblich. Sowohl unter MacOS, als auch unter Linux können natürlich plattformübergreifende Programmieransätze, zum Beispiel mit Hilfe von Java realisiert werden.

WERTUNG

Auf welche Plattform soll man sich nun konzentrieren? Hier muss man u.E. heute noch klar zwischen Desktop und Mobile trennen. Auch künftig werden Business-Applikationen einen Desktop-Rechner bzw. ein Notebook erfordern. Windows wird vorerst in diesem Segment das am meisten verwendete Betriebssystem bleiben. Es bleibt die Frage nach UWP oder Desktop-Anwendungen. Im Moment wird man sich bei neuen Business-Anwendungen meist für den Desktop und damit für eine Entwicklung auf der Basis mit WPF entscheiden. Die Technologie ist ausgreift, es stehen für alle erdenklichen Zwecke externe Komponenten und Bibliotheken zur Verfügung und der Zugriff auf das System ist ohne Einschränkungen. Windows Forms sollte für neue Projekte nicht mehr verwendet werden. Bestehende Projekte können jedoch noch problemlos gepflegt werden. Ob eine Migration zu einer WPF-Anwendung oder zur UWP sinnvoll ist, muss individuell entschieden werden. Der Einsatz der UWP für neue Software ist zu prüfen. Technisch ist der Weg einer WPF-Anwendung zu einer UWA nicht lang. Beide Konzepte basieren auf .Net-Technologien. Für die Businesslogik kann auf C# und für das UI auf XAML gesetzt werden. Dass unter der Haube zwischen WPF und UWP beachtliche Unterschiede liegen, spielt für den Entwickler und Anwender letztendlich keine große Rolle. Windows 10 hat beide Anwendungsplattformen zusammengeführt. Müssen im betrieblichen Umfeld ältere Windows-Versionen unterstützt werden, so ist die UWP keine Option.

Für den Mobile-Bereich sieht die Situation anders aus. Wünscht meine gewisse Marktabdeckung muss man zum heutigen Zeitpunkt Android und iOS ansprechen. An einer jeweiligen nativen Entwicklung führt kein Weg vorbei, wenn die Apps alle Systemfunktionen nutzen. Web-Apps sind nicht immer eine Alternative, da sie eine stetige Internetverbindung voraussetzen und den Zugriff auf Systemfunktionen stark beschränken. Hybride Technologien erlauben mehr systemnähe, kommen jedoch an die Möglichkeiten der nativen Apps nicht heran. Erfolgreich scheint der Weg der plattformübergreifenden Programmierung. In aller Munde ist der Ansatz von Xamarin, welcher echte native Apps erzeugt. Auch auf diesem Weg sind noch Unstimmigkeiten zu beseitigen. Ihn zu beobachten, ist aber auf jeden Fall sinnvoll.

FAZIT

Konkurrenz und Vielfalt ist gut, weil es den Markt belebt und für Innovationen sorgt. Die Hersteller konkurrieren um die Nutzer mit ihren Konzepten. Im Mobil-Bereich gibt es mit Android, iOS und Windows Mobile eine größere Produktvielfalt als im Desktop-Bereich, wo wir seit Jahren eine Dominanz von Windows haben. Für Entwickler stellt sich diese Vielfalt oft komplex dar, denn alle Systeme haben unterschiedliche Programmieransätze, verwenden andere Sprachen und die Konzepte zur UI-Gestaltung unterscheiden sich. Um weiterhin zeitgemäße Anwendungen erstellen zu können, bleibt einen nicht anders übrig, als neue Technologien zu erlernen. Eindimensionale Fähigkeiten führen langfristig ins fachliche Abseits. In diesem Sinne bleiben Sie neugierig!

LITERATUR UND LINKS

[1] http://winfuture.de/news,91244.html

[2] https://netmarketshare.com/

[3] https://msdn.microsoft.com/en-us/magazine/dn973012.aspx

[4] http://xamldeveloper.net/category/visual-studio-tools/

„DREH DIE KAPSEL BITTE WEITER, HAL!"

Neuronale Netze, maschinelles Lernen, Deep-Learning & Co. sind in aller Munde. Doch was genau ist eigentlich ein neuronales Netz und wie funktioniert es? Golo Roden zeigt, dass gängige Schulmathematik ausreicht, um das Konzept zu verstehen.

Autor: GOLO RODEN

Der klassische Weg, Computern neue Fähigkeiten beizubringen, besteht darin, sie zu programmieren. Eine Alternative könnte darin bestehen, Computern das Lernen beizubringen, so dass sie eigenständig Probleme lösen können. Doch stellt sich die Frage, wie das funktioniert: Wie lässt sich Lernen lernen?

Bemerkenswert an der Frage ist, dass jedes Lebewesen vom ersten Augenblick an lernt und diese Fähigkeit „einfach so" mitbringt. Computer übertreffen zwar sämtliche Lebewesen im Hinblick auf ihre Rechengeschwindigkeit, aber lernen können sie nicht von Haus aus. Neuronale Netze sind ein Ansatz, Computern das Lernen an sich beizubringen.

Allerdings sollte man keine Wunder erwarten: Bis zur Leistung einer künstlichen Intelligenz wie dem Computer HAL 9000 aus dem Film „2001: Odyssee im Weltraum" ist es noch ein weiter Weg. Das gleiche gilt für die regelmäßig beschriebene Dystopie, in der eine künstliche Intelligenz mit eigenem Bewusstsein der Menschheit den Garaus macht.

Ein neuronales Netz ist keine allmächtige Wundertechnologie, sondern lediglich eine andere Art, einen Algorithmus zu formulieren, der ein spezifisches Problem löst. Neuronale Netze sind in ihrer Grundform zwar universell einsetzbar, sie müssen aber auf konkrete Probleme geprägt beziehungsweise trainiert werden.

NEURONEN UND DIE SIGMOID-FUNKTION

Die Grundidee für neuronale Netze ist ähnliche wie bei genetischen Algorithmen an die Biologie angelehnt. In der Bologie ist ein Neuron eine Nervenzelle, die über Dendriten und ein Axon die Signale anderer Neuronen empfangen und eigene Signale an diese weiterleiten kann.

Überträgt man das Konzept auf die Informatik, lässt sich ein Neuron als Funktion beschreiben, die mehrere Parameter und einen Rückgabewert hat. In Abhängigkeit der Eingangssignale berechnet sie ein Ausgangssignal. Damit man Neuronen miteinander kommunizieren lassen kann, muss die Kommunikation zwischen ihnen standardisiert sein.

Dazu wird der Wertebereich auf Werte zwischen 0 und 1 begrenzt. Man benötigt also zunächst eine Funk-

tion, die einen beliebigen numerischen Wert auf diesen Bereich reduziert. Dabei ist die Mitte, das heißt der Bereich um die 0.5, interessanter als die Ränder: Das liegt daran, dass eine lediglich geringe Abweichung von einem gewünschten Wert große Auswirkungen haben kann, es aber in der Regel gleichgültig ist, um wie viel eine große Abweichung danebenliegt.

Das lässt sich an einem konkreten Beispiel gut veranschaulichen: Sind auf einer Straße als Höchstgeschwindigkeit 130 km/h zugelassen, spielt es durchaus eine Rolle, ob man mit 129, 130 oder 131 km/h unterwegs war. Ob man sich hingegen mit 220 oder 250 km/h fortbewegt hat, ist verhältnismäßig egal: Beides ist deutlich zu hoch.

Genau das leistet die sogenannte [Sigmoid-Funktion](https://de.wikipedia.org/wiki/Sigmoidfunktion), die sich beispielsweise in JavaScript wie folgt berechnen lässt:

```
const sigmoid = function (t) {
  return 1 / (1 + Math.pow(Math.E, -t));
};
```

Um die Geschwindigkeit zu normieren, muss man nun zunächst 130 abziehen. Dann ergibt eine Eingabe von 130 km/h einen Wert von `0.5`, da gilt:

```
const speed = 130;

const getNormalized = function (speed) {
  return speed - 130;
};

console.log(sigmoid(getNormalized(speed)));
// => 0.5
```

Eine Abweichung von lediglich einem Stundenkilometer nach oben ergibt `0.73`, eine Abweichung nach unten ergibt `0.26`. Wie man sieht, sind die Schwankungen nah um den Nullpunkt sehr groß. Eine Abweichung um 70 (bei 200 km/h) oder 120 (bei 250 km/h) nach oben ergibt aber jeweils `1`. Die Abweichung ist in dem Fall dermaßen groß, dass es keine Rolle mehr spielt, wie groß sie ist.

Geht man nun von der (hypothetischen) Annahme aus, dass zu schnelles Fahren bei Regen noch weitaus schlimmer ist als bei Sonnenschein, könnte man auf die Idee kommen, eine Funktion zu schreiben, die neben der Geschwindigkeit auch das Wetter als Eingabe erwartet und einen gemeinsamen Wert berechnet. Dazu gilt es aber zunächst, das Wetter numerisch zu codieren.

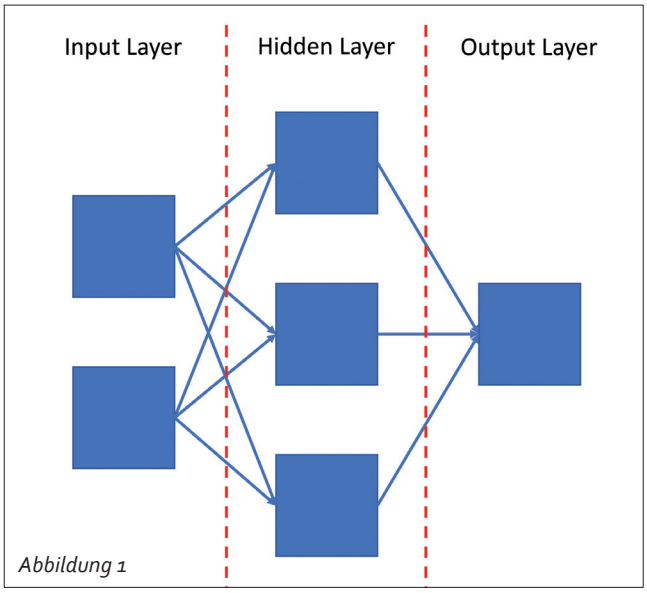


Abbildung 1

Der Einfachheit halber könnte man beispielsweise festlegen, dass Regen dem Wert `1` entspricht, und fehlender Regen dem Wert `-1`. Das lässt sich in dem Fall so einfach codieren, da in dem Beispiel eine rein duale Logik zu Grunde liegt: Entweder es regnet oder es regnet nicht. In der Praxis ist das Codieren von Werten deutlich aufwändiger und erfordert mehr Nachdenken.

Aus den Überlegungen ergibt sich nun eine Funktion, die die beiden Eingaben entgegennimmt, aufsummiert und sie anschließend mit Hilfe der Sigmoid-Funktion in eine normierte Ausgabe überführt:

```
const score = function (speed, isRaining) {
  const normalizedSpeed = getNormalized(speed),
        normalizedWeather = isRaining ? 1 : -1;

  const sum = normalizedSpeed + normalizedWeather;

  return sigmoid(sum);
};
```

Ruft man die Funktion nun mit verschiedenen Werten auf, lässt sich beobachten, wie sich die Ausgabe verändert:

```
score(130, true);  // => 0.73
score(130, false); // => 0.26
score(131, true);  // => 0.88
score(131, false); // => 0.5
```

DIE EINGABEN GEWICHTEN

Wie man sieht, steigt der Wert bei steigender Geschwindigkeit. Außerdem führt Regen zu einem höheren Wert. So weit, so gut. Allerdings weist die Berechnung einen gravierenden Makel auf: Die Geschwindigkeit und das Wetter fließen zu gleichen Teilen in das Ergebnis ein.

Besser wäre es, die Anteile gezielt gewichten zu können. Das ist kein Problem, denn dazu muss lediglich der jeweilige Parameter noch mit einem frei wählbaren Faktor multipliziert werden. Soll beispielsweise die Geschwindigkeit eine doppelt so große Rolle wie das Wetter spielen, könnte man als Faktoren zum Beispiel die Zahlen `1` und `2` wählen:

```
const score = function (speed, isRaining) {
  const normalizedSpeed = getNormalized(speed),
        normalizedWeather = isRaining ? 1 : -1;

  const weightedSum =
    2 * normalizedSpeed +
    1 * normalizedWeather;

  return sigmoid(weightedSum);
};
```

Allgemein formuliert ergibt sich auf dem Weg eine Funktion, die wie folgt aussieht. Die Parameter `weights` und `inputs` sind dabei jeweils Arrays und müssen die gleiche Länge aufweisen. Außerdem müssen die Eingaben zuvor bereits normiert worden sein:

```
const f = function (weights, inputs) {
  let weightedSum = 0;

  for (let i = 0; i < weights.length; i++) {
    weightedSum += weights[i] * inputs[i];
  }

  const output = sigmoid(weightedSum);

  return output;
};
```

VON NEURONEN ZU NEURONALEN NETZEN

Eine solche Funktion ist ein Neuron. Schaltet man nun mehrere dieser Funktionen zu einem Netz zusammen, ergibt sich ein sogenanntes neuronales Netz. Neuronale Netze werden aus Schichten aufgebaut, wobei jede Schicht wiederum aus mehreren Neuronen bestehen kann.

Die Idee dabei ist, dass jedes Neuron einer Schicht als Eingaben die Ausgaben sämtlicher Neuronen der Vorgängerschicht erhält. Seine eigene Ausgabe wird wiederum an alle Neuronen der Nachfolgeschicht weitergereicht.

Klassische neuronale Netze bestehen aus drei Schichten: Die erste Schicht ist der sogenannte I*nput Layer*, die dritte der sogenannte *Output Layer*. Die Schicht dazwischen wird als *Hidden Layer* bezeichnet (siehe Abbildung 1). Verfügt ein neuronales Netz über mehr als einen Hidden Layer, spricht man von *Deep Learning*.

Das Kunststück besteht nun darin, die Gewichte der einzelnen Neuronen in den verschiedenen Schichten so zu wählen, dass sich das gewünschte Ergebnis einstellt. Wie man sich leicht vorstellen kann, ist das bereits bei einfachen Netzen eine mühselige Arbeit, die von Hand kaum zu bewerkstelligen ist.

Daher benötigt man einen anderen Ansatz, um das gewünschte Ergebnis zu erzielen. Den Vorgang, ein neuronales Netz zu konfigurieren, bezeichnet man als Training. Um ein neuronales Netz trainieren zu können, müssen Trainingsdaten zur Verfügung stehen, die das gewünschte Ergebnis beschreiben.

Da alle Geschwindigkeiten kleiner als 130 km/h unproblematisch sind, lässt sich für sie ein einheitlich niedriger Wert festlegen. Bei höheren Geschwindigkeiten wird die Ausgabe dann allerdings interessant:

Speed	Raining	Result
50	false	0.10
50	true	0.10
120	false	0.10
120	true	0.10
130	false	0.10
130	true	0.10
135	false	0.60
135	true	0.70
140	false	0.66
140	true	0.80
200	false	0.90
200	true	0.99

Prinzipiell ließe sich einwenden, dass dafür kein neuronales Netz erforderlich ist, schließlich ist die Kombination der möglichen Eingabevarianten endlich. Der Einwand ist durchaus berechtigt, jedoch übersteigt die Anzahl der möglichen Kombinationen, gerade wenn noch weitere Parameter ins Spiel kommen, rasch die Möglichkeit, die zur Berechnung des Ergebnisses erforderliche Funktion von Hand zu entwickeln.

EIN NEURONALES NETZ TRAINIEREN

Das Training eines neuronalen Netzes läuft nun folgendermaßen ab: Zunächst initialisiert man alle Gewichte mit beliebigen Werten. Dabei spielt es keine Rolle, ob man sie willkürlich wählt oder sie von einem Zufallsgenerator bestimmen lässt. Anschließend übergibt man den ersten Datensatz der Trainingsdaten an das neuronale Netz und lässt es das Ergebnis berechnen.

Das Ergebnis wird höchst wahrscheinlich nicht dem gewünschten Ergebnis entsprechen. Also gilt es nun die Gewichte zu justieren, um dem gewünschten Ergebnis näher zu kommen. Das neuronale Netz „lernt" auf dem Weg mehr über das von ihm erwartete Verhalten. Es wird gewissermaßen konditioniert.

Anschließend übergibt man dem neuronalen Netz den nächsten Datensatz aus den Trainingsdaten und berechnet erneut das Ergebnis, vergleicht es wieder mit dem gewünschten Ergebnis und passt die Gewichte erneut so an, dass das tatsächliche Ergebnis dem erwarteten näher kommt - ohne sich dabei aber wiederum zu weit vom ersten Datensatz zu entfernen.

Der Vorgang wird anschließend vielfach wiederholt. Mit jeder Iteration passt sich das neuronale Netz besser an die Trainingsdaten an. Das tatsächlich berechnete Ergebnis verbessert sich dabei zusehends, und nach einer Weile ist das Netz dann auch in der Lage, realistische Ergebnisse für noch nicht bekannte Werte zu berechnen.

FEHLER VERMEIDEN

Gelegentlich kommt es zu einem Effekt, der als *Overfitting* bezeichnet wird. In dem Fall lernt das neuronale Netz die Trainingsdaten zu gut kennen und passt sich exakt auf sie an. Dabei verliert es die Fähigkeit, aus den Trainingsdaten zu abstrahieren, um auch unbekannte Eingaben sinnvoll verarbeiten zu können.

Daher ist es empfehlenswert, die initial gegebene Menge von Trainingsdaten in zwei Hälften zu teilen, und nur die eine für das Training zu verwenden. Mit der anderen lässt sich die Qualität des Ergebnisses unabhängig kontrollieren.

Wichtig ist auch die Frage, wie man den Fehler überhaupt bewertet. Angenommen, die Trainingsdaten geben als gewünschtes Ergebnis eine 0.8 vor, das neuronale Netz berechnet jedoch lediglich eine 0.6. Wie groß ist dann der Fehler? Auf den ersten Blick könnte man sagen, er sei 0.2, da man zum berechneten Wert von 0.6 noch 0.2 hinzuzählen muss, um zu 0.8 zu gelangen.

Analog wäre der Fehler, der bei einer 0.8 eine 1.0 berechnet, -0.2. Warum das problematisch ist, zeigt sich, wenn man den mittleren Fehler ermitteln will: Das wäre in dem Fall nämlich 0, weil sich 0.2 und -0.2 gegenseitig aufheben! Doch selbstverständlich ist ein Fehler von 0.2 genauso schlecht wie ein Fehler von -0.2. Daher wird der Fehler üblicherweise quadriert, was diesen Effekt verhindert:

```
 0.2 *  0.2 = 0.04
-0.2 * -0.2 = 0.04
```

Außerdem werden durch das Quadrieren größere Fehler stärker gewichtet als kleinere, was das Lernen beschleunigt: Man weiß, dass man in diesem Fall einen größeren Schritt in die richtige Richtung machen kann als bei einem nur sehr kleinen Fehler.

Einen Schritt in die richtige Richtung heißt in dem Zusammenhang, den Fehler zu minimieren. Das funktioniert über das Konzept der mathematischen Ableitung, womit sich die Steigung einer Funktion berechnen lässt.

Nun zeigt sich auch, warum die Sigmoid-Funktion so eine häufig und gern verwendete Funktion im Zusammenhang mit neuronalen Netzen ist: Sie ist stetig und differenzierbar, das heißt, ihre Ableitung lässt sich problemlos berechnen. Prinzipiell lassen sich aber auch andere Funktionen verwenden, solange sie ähnliche Eigenschaften aufweisen.

Nachdem man den Fehler des Ergebnisses berechnet hat, gilt es, die Gewichte für den Output Layer zu berechnen und den Fehler auf dessen Gewichte anteilsmäßig zu verteilen. Anschließend wendet man auch das Vorgehen wieder iterativ an und passt die Gewichte des Hidden Layers an, und schließlich jene des Input Layers. Das Vorgehen wird als Back Propagation bezeichnet. Je mehr Layer ein Netz enthält und je mehr Neuronen beteiligt sind, desto aufwändiger wird diese Berechnung.

DIE BERECHNUNG BESCHLEUNIGEN

Eine spannende Frage ist, wie sich das beschleunigen lässt. Dazu gibt es zwei Überlegungen: Zum einen lassen sich die Berechnungen an sich beschleunigen. Da sich die Berechnungen als Operationen der Vektor- und Matrizenrechnung abbilden lassen, kann hierbei auf entsprechend optimierte Bibliotheken zurückgegriffen werden.

Auf dem Weg lassen sich viele unnötige Schleifen einsparen. Das Ersetzen von schleifenbasiertem und prozeduralem Code durch optimierte Berechnungen mit Vektoren und Matrizen wird als Vektorisierung bezeichnet.

Die zweite Überlegung ist, wie viele Neuronen und Layer überhaupt verwendet werden sollten. Die Anzahl der Neuronen im Input Layer und Output Layer lässt sich leicht festlegen: Die Anzahl der Variablen entspricht der Anzahl der Neuronen im Input Layer, die Anzahl der gewünschten Ergebnisvariablen entspricht der Anzahl der Neuronen im Output Layer.

Im vorliegenden Beispiel, wo die Geschwindigkeit und das Wetter auf einen einzigen numerischen Wert abgebildet werden sollen, bedeutet das, dass das neuronale Netz über zwei Neuronen im Input Layer und ein Neuron im Output Layer verfügt.
Doch was ist mit dem Hidden Layer beziehungsweise den Hidden Layers? Wie viele Schichten sollte man verwenden, und wie viele Neuronen sollte jede einzelne Schicht aufweisen? So unbefriedigend das klingen mag, ist der beste Ansatz, verschiedene Varianten auszuprobieren und die einzelnen Ergebnisse zu vergleichen.

Dann lässt sich auf Grund der Qualität der Ergebnisse und der benötigten Rechendauer bestimmen, welchen Pfad man weiterverfolgen sollte und wo sich weiteres Training lohnt.

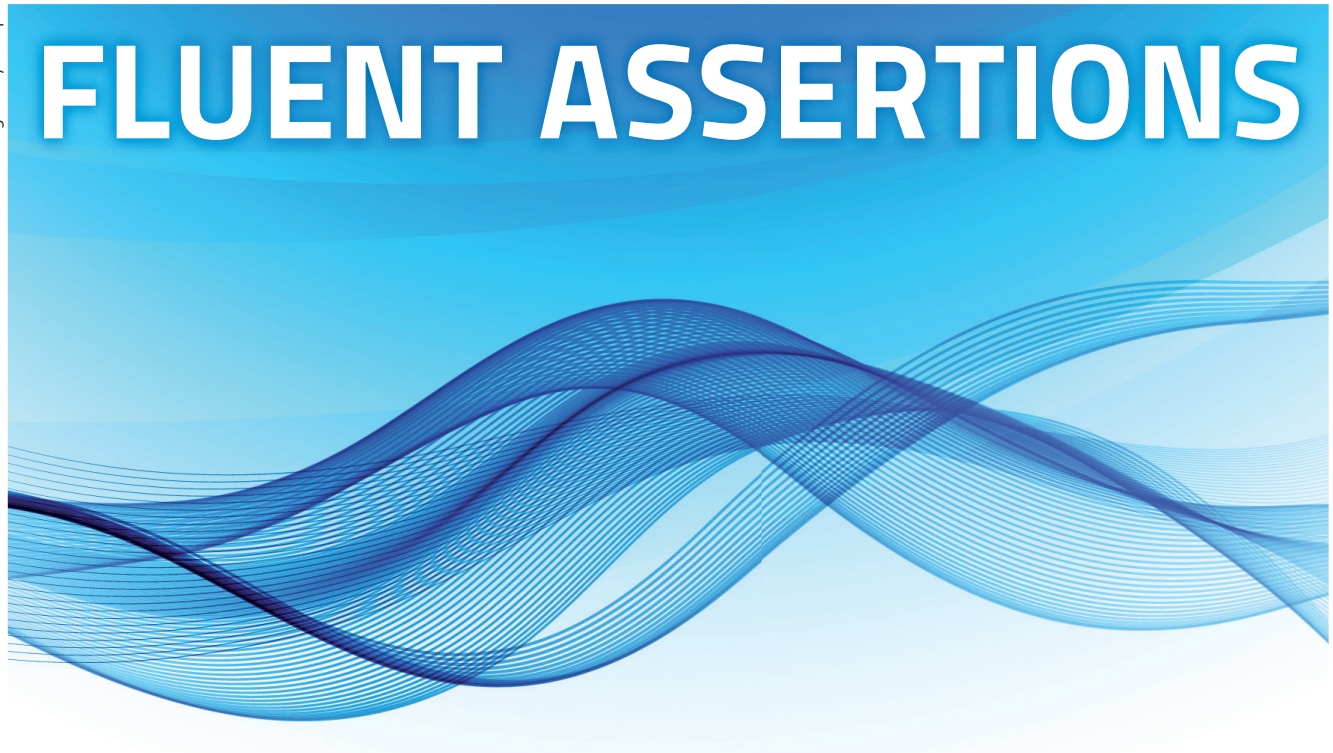
FAZIT

Wie leicht zu sehen ist, sind neuronale Netze keine schwarze Magie. Sie sind lediglich eine Reihe von für sich genommen sehr einfachen Funktionen, die parallel und in Reihe geschaltet werden. Insbesondere haben neuronale Netze kein Bewusstsein und haben auch keine eigenen Wünsche, Bedürfnisse oder Ideen. Daher ist die Angst vor einer Machtübernahme durch künstliche Intelligenz unbegründet, zumindest im Hinblick auf neuronale Netze.

Trotzdem können diese Systeme beeindruckendes leisten. Das Hauptproblem besteht meistens allerdings darin, die Eingangsdaten überhaupt erst einmal in ein passendes Format zu transformieren. Für das gezeigte Beispiel war das einfach, doch wie geht man mit Bildern oder Audiodaten um? Das sind im praktischen Gebrauch die eigentlich spannenden Fragen, bei denen es sich häufig lohnen kann, einen Datenexperten zu Rate zu ziehen, der Erfahrung mit maschinellem Lernen hat.

Wer kein neuronales Netz von Grund auf selbst entwickeln will, kann auf eins der zahllosen Systeme am Markt zurückgreifen. Dabei muss es je nach Anwendungsfall nicht immer um ein hochkomplexes System wie [TensorFlow](https://www.tensorflow.org/) handeln, es finden sich durchaus auch kleine leistungsfähige Module in der Programmiersprache der Wahl.

Designed by Freepik



Endlich Unabhängigkeit vom Test Framework

Autor: STEFAN LIESER

Es existieren auf der .NET Plattform inzwischen eine ganze Reihe von Test Frameworks. Die am meisten verwendeten sind *MSTest* [1], *NUnit* [2] und *xUnit* [3]. Im Grunde leisten diese Frameworks alle das Gleiche: Sie ermöglichen die Automatisierung von Tests. Die Funktionalität der Test-Frameworks besteht im Wesentlichen aus zwei Bereichen:

- Da sind zum einen die *Attribute*, mit deren Hilfe der Testrunner die Testmethoden einsammelt. Die Grundfunktionalität ist bei allen Frameworks ähnlich. Der größte Unterschied liegt im Bereich datengetriebener Tests.
- Der zweite Bereich umfasst die *Assert* Methoden. Hier sind die Unterschiede schon deutlicher. Während MSTest noch die ursprüngliche und stark eingeschränkte *Assert.AreEqual* Syntax verwendet, haben NUnit und xUnit sich in dem Bereich stark weiterentwickelt.

Persönlicher Favorit des Autors ist nach wie vor NUnit. Es verfügt über einen sehr reichhaltigen Satz an *Assert* Methoden. Ferner sind die datengetriebenen Tests mittels Attributen sehr leistungsfähig. xUnit ist ähnlich stark, jedoch fehlt das *Explicit* Attribut, mit dem ein Test markiert wird, der explizit gestartet werden muss und andernfalls nicht mitläuft. Das Attribut ist sehr nützlich für langlaufende Tests, UI Tests oder Integrationstests mit Ressourcenzugriffen ins Internet.

Nun gibt es allerdings Projekte, in denen die Wahl bereits getroffen wurde. Häufig wird dabei auf MSTest gesetzt, weil es „out of the box" funktioniert. Nach der Installation von Visual Studio stehen entsprechende Projekt Templates zur Verfügung, der Visual Studio Testrunner findet die Tests, die Integration in den TFS und den Buildprozess ist leicht, etc. Das sind für viele Teams Gründe, bei MSTest zu bleiben, obschon die *Assert* Syntax nicht die Beste ist. Zwar ist eine Referenz auf NUnit oder xUnit mit NuGet schnell hergestellt, doch es scheint einen Hang zur Bequemlichkeit zu geben, sodass viele Teams erst bei Microsoft nach einer Lösung schauen und Alternativen häufig nicht in Betracht ziehen.

Egal welches Unit Test Framework im Projekt gesetzt wird: die *Assert* Syntax kann vereinheitlicht und deutlich ausdrucksstärker gestaltet werden. Mit *Fluent Assertions* [4] steht eine Open Source Bibliothek zur Verfügung, die mit allen gängigen Unit Test Frameworks zusammenarbeitet. Die Grundidee: Man verwendet die Attribute und den Test Runner des gewohnten Frameworks. Lediglich die Assertions werden mit dem Fluent Assertions Framework in anderer Weise geschrieben.

Fluent Assertions setzt auf ein sogenanntes „fluent interface". Damit werden APIs bezeichnet, die eine fast natürlichsprachliche Formulierung ermöglichen. Insbesondere werden hier Methodenaufrufe von links nach rechts, also im Lesefluss, hintereinander verkettet. Dazu ein Beispiel:

```
var i = 42;
i.Should().Be(42);
```

Die Annahme würde auf englisch formuliert lauten „The value of i should be 42.". Mit ein wenig syntaktischer Ergänzung von Klammern und Punkten wird daraus ein gültiges C# Konstrukt.

Die Syntax der Fluent Assertions Bibliothek wird technisch möglich, durch sogenannte Extension Methods. Diese wurden mit dem

Sprachstandard C# 3.0 eingeführt. Mit *Extension Methods* ist es möglich, eine statische Methode so zu formulieren, dass sie auf einem vorhandenen Typ angewandt werden kann. Im Ergebnis sieht es so aus, als wäre die Methode auf dem Typ selbst implementiert:

```
public static class DoubleExtensions
{
public static string ToCurrency(this
double euro) {
return euro.ToString("C");
}
}
```

Die statische Methode *DoubleExtensions.ToCurrency* kann nun direkt auf double Werte angewandt werden. Syntaktisch ist es notwendig, die Methode statisch zu definieren. Ferner muss sie in einer statischen Klasse definiert sein. Der Unterschied zu einer normalen statischen Methode entsteht dadurch, dass man den ersten Parameter zusätzlich mit dem Schlüsselwort this kennzeichnet. Durch diese Deklaration kann die Methode nun wie folgt angewandt werden:

```
string result = 3.42.ToCurrency();
```

Der Compiler formt dieses Statement um:

```
string result = DoubleExtensions.
ToCurrency(3.42);
```

Durch die Umformung des Compilers musste bei der Einführung der Syntax keine Veränderung an der CLR, der Common Language Runtime, vorgenommen werden. Es handelt sich lediglich um "compiler magic".

Die Fluent Assertion Bibliothek macht sich diese Möglichkeiten der C# Syntax zunutze, um die Assertions so flüssig wie möglich formulieren zu können. Folgendes Listing zeigt eine Testklasse, die für MSTest erstellt wurde, aber Fluent Assertions enthält, anstelle von *Assert.AreEqual:*

```
[TestClass]
public class WürfelbecherTests
{
private Würfelbecher sut;

[TestInitialize]
public void Setup() {
sut = new Würfelbecher(new
Random(42));
}

[TestMethod]
public void Wurf_besteht_aus_n_Wür-
feln() {
int[] result = sut.Würfeln(5);
result.Should().HaveCount(5);
}
}
```

Die Attribute *TestClass*, *TestInitialize* und *TestMethod* stammen von MSTest. Auf diese Weise ist die Testklasse mit dem in Visual Studio integrierten Unit Test Runner ausführbar. Es wurde lediglich per NuGet eine Referenz auf das Projekt „FluentAssertions" ergänzt. Daher können die Annahmen mit der *Should*-Schreibweise formuliert werden. Das Beispiel zeigt bereits eine der umfangreichen Assertions für Aufzählungen. Die Variable *result* ist ein Integer Array. Daher steht die Methode *HaveCount* zur Verfügung, um auszudrücken, dass die Aufzählung 5 Elemente enthalten muss. Die Annahme lautet hier also „result should have count 5". Für Aufzählungen gibt es zahlreiche nützliche Assertions, die Tests gut lesbar machen. Folgendes Listing zeigt einige Beispiele.

```
var x = new[] { 1, 2, 3, 4, 5 };
x.Should().HaveCount(5);
x.Should().Equal(1, 2, 3, 4, 5);
x.Should().BeEquivalentTo(5, 1, 3,
4, 2);
x.Should().BeInAscendingOrder();
x.Should().NotBeEmpty();
```

HaveCount kennen wir bereits von weiter oben. Es drückt aus, dass die Aufzählung die angegebene Anzahl von Elementen haben muss. Die *Equal* Methode verwendet *params int[]* in der Signatur, so dass die einzelnen Elemente direkt angegeben werden können, statt sie als Array zu schreiben. Alle Elemente müssen in der gegebenen Reihenfolge in der Aufzählung vorhanden sein. Die *BeEquivalentTo* Variante prüft dagegen, ob alle Elemente vorhanden sind, ohne Berücksichtigung der Reihenfolge. Mit *BeInAscendingOrder* wird ausgedrückt, dass die Elemente aufsteigend sortiert sein müssen. *NotBeEmpty* fordert eine nicht-leere Aufzählung. Es gibt noch viele weitere Assertions auf Aufzählungen. Es lohnt sich, einen Blick darauf zu werfen.

Auch für Strings hat die Bibliothek einiges zu bieten.

```
var s = "Hello world";
s.Should().Contain("lo");
s.Should().StartWith("Hell");
s.Should().EndWith("world").And.Con-
tain(" ");
```

Besonders interessant: Man kann generell, nicht nur bei Strings, mehrere Annahmen mit *.And.* verbinden. Wirklich spannend wird es dann beim Thema Events. Folgendes Listing zeigt eine Klasse *WithEvent*, die in ihrer *Fire* Methode einen Event auslöst.

```
using System;
using FluentAssertions;
using NUnit.Framework;

namespace fluentassertions
{
[TestFixture]
public class EventsTests
{
[Test]
public void Event_is_raised() {
var sut = new WithEvent();
sut.MonitorEvents();

sut.Fire();

sut.ShouldRaise("MyEvent");
}
}

public class WithEvent
{
public event Action MyEvent;

public void Fire() {
MyEvent();
}
    }
}
```

Interessant ist, wie nun mit Fluent Assertions überprüft werden kann, ob die *Fire* Methode den Event *MyEvent* auslöst. Dazu muss die zu prüfende Instanz in der *Arrange* Phase durch den Aufruf *MonitorEvents* instrumentiert werden. Diese Methode sorgt dafür, dass hinter den Kulissen ein Handler an alle Events gebunden wird. In der Assert Phase des Tests kann dann mit *ShouldRaise* überprüft werden, ob der angegebene Event tatsächlich ausgelöst wurde. Leider muss der Event hier als String angegeben werden, so dass man beim Umbenennen des Events aufpassen muss, dass auch dieser String mit umbenannt wird.

Einen ähnlichen Mechanismus gibt es für Viewmodels. Diese müssen das *INotifyPropertyChanged* Interface implementieren. Um nun zu prüfen, ob eine Eigenschaft den erforderlichen Event auslöst, bietet Fluent Assertions dafür eine spezielle Assertion:

```
[Test]
public void ViewModel_raises_Pro-
pertyChanged_Event() {
   var sut = new MyViewModel();
   sut.MonitorEvents();

   sut.Vorname = "Stefan";
```

```
    sut.ShouldRaisePropertyChangeFor(x => x.Vorname);
}
```

Der Methode *ShouldRaisePropertyChangeFor* wird die Eigenschaft, hier Vorname, in Form eine Lambda Expression übergeben. Das ist super, weil damit das Rename Refactoring nicht nach Strings suchen muss.

Eine weitere nützliche Annahme kann zu Objekten getroffen werden, die auf den ersten Blick gleich aussehen. Im folgenden Listing haben die Instanzen x und y die gleichen Werten. Trotzdem gelten sie nicht als gleich. Bei *class* gilt beim Vergleich mit Equals oder == die Referenzgleichheit. Nur bei *struct* ist es die Inhaltsgleichheit.

```
var x = new Adresse { Street = "Street 1", Town = "Town A" };
var y = new Adresse { Street = "Street 1", Town = "Town A" };
var z = new Adresse { Street = "Street 2", Town = "Town A" };

Assert.That(x, Is.Not.EqualTo(y));

x.ShouldBeEquivalentTo(y);
//x.ShouldBeEquivalentTo(z);     // schlägt fehl!
```

In NUnit Syntax wird durch *Is.Not.EqualTo* ausgedrückt, dass die beiden Objekte nicht gleich sind, im Sinne der *Equals Methode*. *ShouldBeEquivalentTo* drückt dagegen aus, dass die beiden Objekte den gleichen Inhalt haben. Im Beispiel sind x und z in diesem Sinne nicht gleich, da sie sich bei der Eigenschaft Street unterscheiden. Durch die *ShouldBeEquivalentTo* Assertion wird der Einsatz der *Equalidator* Bibliothek überflüssig [5]. *ShouldBeEquivalentTo* vergleicht die öffentlichen Eigenschaften und Felder der beiden Objekte. *Equalidator* vergleicht zusätzlich auch die privaten Felder.

Die syntaktischen Möglichkeiten von Extension Methods werden für Datum- und Uhrzeitvergleiche auf besonders elegante Weise ausgeschöpft.

```
var d = new DateTime(2015, 2, 4);
d.Should().Be(4.February(2015));
```

Die Definition eines Datum über den *DateTime* Konstruktor ist etwas schwerer lesbar, als die flüssige Schreibweise "4.February(2015)". Mit der *At* Methode kann zusätzlich auch die Uhrzeit angegeben werden:

```
d = 5.March(2016).At(12, 45);
d.Should().Be(new DateTime(2016, 3, 5, 12, 45, 0));
```

Neben einer Prüfung auf den exakten Wert sind auch Vergleiche möglich, wie die folgenden Beispiele zeigen:

```
d.Should().BeAfter(1.January(2010));
d.Should().BeAtLeast(1.Days()).After(1.March(2016));
d.Should().BeLessThan(2.Hours()).Before(5.March(2016).
At(14, 0));
```

Auch das Testen von Exceptions wird von der Fluent Assertions Bibliothek unterstützt. Die Annahme, dass eine Methode eine Ausnahme auslöst, kann hiermit in gut lesbarer Form ausgedrückt werden:

```
public class Sut
{
    public void ThrowAnException() {
throw new ArgumentException("Please don't call me.");
    }
}

[TestMethod]
public void Exceptions() {
    var sut = new Sut();
    sut.Invoking(x => x.ThrowAnException())
.ShouldThrow<ArgumentException>()
.WithMessage("Please don't call me.");
}
```

FAZIT

Wer mit der wenig ausdrucksstarken *Assert.AreEqual* Syntax unzufrieden ist, sollte unbedingt mit der *Fluent Assertions* Bibliothek experimentieren. Sie steht auch schon für .NET Core zur Verfügung. Wenn Projektzwänge das Team daran hindern, von MSTest wegzugehen, bietet *Fluent Assertions* eine elegante Lösung, um den wichtigsten Teil der automatisierten Tests deutlich lesbarer zu gestalten. Und auch wer NUnit oder xUnit einsetzt profitiert von der besseren Lesbarkeit. Werden alle Assertions mit dieser Bibliothek formuliert, spielt die Wahl des Unit Test Frameworks eine geringere Rolle.

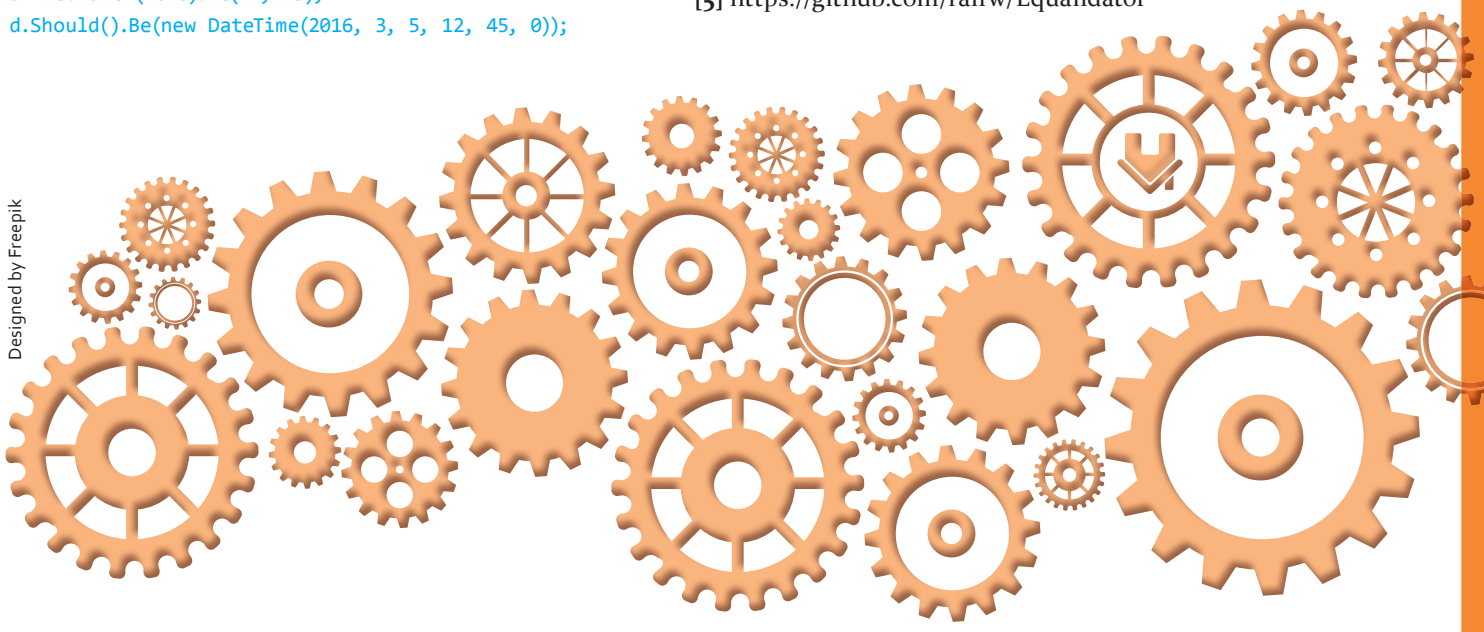
[1] https://www.nuget.org/packages/MSTest.TestFramework/

[2] http://nunit.org/

[3] https://xunit.github.io

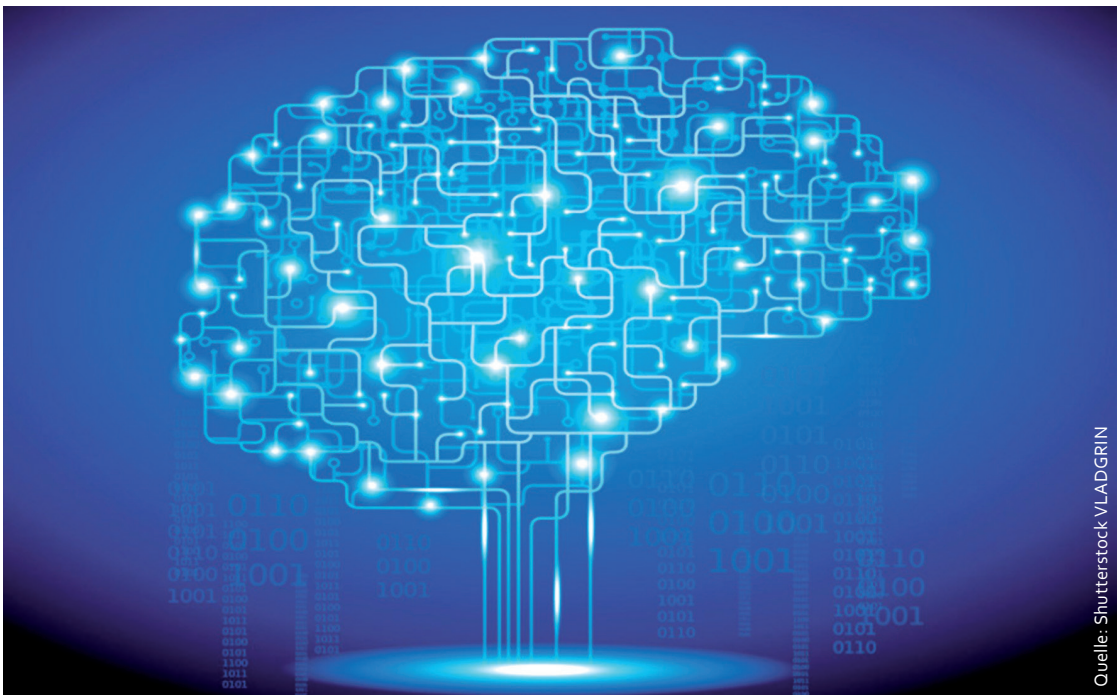
[4] http://www.fluentassertions.com

[5] https://github.com/ralfw/Equalidator



Designed by Freepik

MASCHINELLES LERNEN



Wer bei maschinellem Lernen an neuronale Netze und Deep-Learning denkt, liegt nicht falsch, verkennt aber zahlreiche andere Möglichkeiten. Ein Überblick über die verschiedenen Optionen.

Autor: GOLO RODEN

Maschinelles Lernen ist ein Bereich der künstlichen Intelligenz. Während es bei der künstlichen Intelligenz um das Finden, Begründen, Lernen und Begreifen im Allgemeinen geht, beschäftigt sich der Zweig des maschinellen Lernens speziell mit den Problemen, auf Basis von in der Vergangenheit gemachten Erfahrungen künftig besser agieren zu können.

Die Forschung in diesem Bereich der künstlichen Intelligenz ist alles andere als neu, lediglich die Aufmerksamkeit ist in den vergangenen Jahren wieder gestiegen. Es wäre aber falsch anzunehmen, dass das maschinelle Lernen die künstliche Intelligenz verdrängt hätte: Auf Grund der Beziehung zwischen den beiden Themen ergibt eine solche Aussage wenig Sinn.

Spannend ist die Frage, ob Maschinen überhaupt lernen können. Viele Menschen streiten das ab, da sie mit dem Begriff des Lernens automatisch andere Aspekte verbinden, zum Beispiel die Frage nach Bewusstsein, Bedürfnissen oder Ideen. Während eine Maschine über kein Bewusstsein verfügt, kann sie durchaus lernen: Die Erfolge von [AlphaGo](https://deepmind.com/research/alphago/) im Frühjahr 2016 zeigen, dass eine Maschine beispielsweise in einem Brettspiel eigenständig besser werden kann als ein Mensch.

Lernen bedeutet in dem Zusammenhang also lediglich, dass ein Ergebnis auf Basis von Erfahrung verbessert wird. Fährt ein Staubsaugerroboter mehrfach gegen ein Hindernis und verringert diese Zusammenstöße nach und nach, lernt er seine Umgebung kennen. Selbstverständlich bedeutet das nicht, dass er ein Bewusstsein oder gar ein Verständnis seiner Umgebung hätte, aber er hat gelernt: Er verbessert sein Ergebnis auf der Basis von Erfahrungen.

WAS IST GUT, WAS IST SCHLECHT?

Das Beispiel zeigt bereits, dass zum Lernen aber Feedback gehört: Wenn niemand den Roboter anweist, Zusammenstöße zu vermeiden, wird er sein Verhalten nicht ändern, da er keinen Anlass dazu sieht. Im maschinellen Lernen spricht man dabei von sogenanntem *Supervised Learning*. Das bedeutet, dass die Maschine Feedback zu ihrem Verhalten erhält, um darauf reagieren zu können.

Im Gegensatz dazu steht das *Unsupervised Learning*, bei dem eine Maschine kein Feedback erhält. Obwohl das scheinbar im Widerspruch zu der vorigen Erkenntnis steht, gibt es durchaus Algorithmen, die ohne menschliches Zutun auskommen. Es hängt allerdings stark vom Anwendungsfall ab, welche Art Algorithmus überhaupt zur Verfügung steht.
Apropos Anwendungsfall: Hier un-

terscheidet man hauptsächlich zwei Kategorien, nämlich die *Regression* und die *Klassifizierung*. Bei der Regression geht es darum, Ergebnisse und Werte vorherzusagen, beispielsweise den Aktienkurs oder das Wetter. Die Klassifizierung hingegen befasst sich mit der Einordnung von Dingen in Klassen, um beispielsweise qualitativ minderwertige Ware in der Produktion erkennen zu können.

Ein wesentlicher Faktor beim maschinellen Lernen ist außerdem die Frage danach, was als *Fehler* anzusehen ist und wie sich ein solcher bewerten lässt. Eine große Rolle spielen dabei sogenannte Fehlerfunktionen, die einen gegebenen Fehlerwert bewerten. Eine häufig genutzte Fehlerfunktion ist die mittlere quadratische Abweichung.

Sie lässt sich einfach berechnen und bringt dank der Quadratur einige äußerst praktische Konsequenzen mit sich: So werden beispielsweise eine Abweichung von 0.2 und -0.2 als gleich schlimm angesehen. Auf dem Weg wird verhindert, dass sich positive und negative Fehler im Mittel ausgleichen. Außerdem bewertet sie, ebenfalls auf Grund der Quadratur, große Fehler als schlimmer als kleine Fehler.

KLASSIFIZIERUNG, UNBEAUFSICHTIGT

Ein bekannter Algorithmus zur Klassifizierung von Objekten, der ohne voriges Training und ohne menschliches Feedback auskommt, ist K-Means. Er klassifiziert eine beliebige Menge von Objekten, wobei einzig die Anzahl der zu erzeugenden Klassen vorgegeben werden kann. K-Means erwartet die Eingabedaten als Vektoren. So wäre beispielsweise denkbar, Farben mit Hilfe von K-Means nach Ähnlichkeit zu gruppieren.

Jede Farbe lässt sich als dreidimensionaler Vektor darstellen, wobei die drei Dimensionen für die R-, G- und B-Werte der Farbe stehen. Die Farbe Weiß ließe sich so als [255, 255, 255] abbilden, Schwarz hingegen als [0, 0, 0]. Angenommen, es sind eine Reihe von Farben als solche Vektoren gegeben, könnte die Aufgabe darin bestehen, sie in fünf Gruppen einzuteilen.

Dazu wählt K-Means zunächst fünf zufällige Farben aus und initialisiert damit die fünf Gruppen. Anschließend wird für jede Farbe der Abstand von den fünf zufällig ausgewählten Farben berechnet. Die Farbe wird jener Gruppe zugeordnet, wo der Abstand am geringsten ist. Anschließend werden die Mittelpunkte der Gruppen neu berechnet und das Verfahren startet von vorn. Irgendwann ändert sich die Zuordnung der Farben zu den Gruppen nicht mehr: Dann endet das Verfahren.

Einer der großen Nachteile an K-Means ist, dass je nach (zufälliger) Wahl der Startwerte gänzlich unterschiedliche Ergebnisse herauskommen, und man keinerlei Einfluss darauf hat, ob inhaltliche sinnvolle Cluster gebildet werden und sich aus ihnen eine Aussage herleiten lässt.

LINEARE REGRESSION

Im Gegensatz zu K-Means steht die lineare Regression, die - wie der Name schon sagt - der Regressionsanalyse dient, außerdem erfordert sie menschliches Feedback. Bei der linearen Regression geht es um die Vorhersage eines Wertes für unbekannte Daten. So könnte man mit linearer Regression beispielsweise berechnen, wie hoch der Preis beim Verkauf eines Hauses anzusetzen ist, auf Grund der Daten des Hauses und den in der Vergangenheit erzielten Preisen.

Im einfachsten Fall hängt der Preis in dem genannten Beispiel nur von einer einzigen anderen Variable ab, beispielsweise der Größe des Hauses. In dem Fall lässt sich der optimale Verkaufspreis als Gerade ermitteln (siehe Abbildung 1). Dazu bedarf es allerdings einer Menge an Trainingsdaten, das heißt einer im Vorfeld erhobenen Sammlung von Beispieldaten. Die optimale Gerade ist dann diejenige, die den Fehler über alle Datenpunkte minimiert.

Um die optimale Gerade zu finden dient das *Gradientenabstiegsverfahren* (im Englischen *Gradient Descent*). Dazu wird zunächst die Fehlerfunktion definiert, die dem Quadrat der Abweichung von den tatsächlich berechneten Werten und den erwarteten Werten entspricht. Angenommen, als Parameter für die Gerade wurden ein y-Achsenabschnitt von 100.000 und eine Steigung von 1.000 gewählt (anschaulich bedeutet das, dass ein Haus an sich bereits 100.000 Euro kostet, zuzüglich 1.000 Euro pro Quadratmeter).

Aus diesen Werten ergibt sich dann für eine Größe von 200qm ein Preis von 300.000 Euro:

```
const squareMeters = 200;

const getPrice = function (square-
Meters) {
  const base = 100000,
        pricePerSquareMeter = 1000;

  const price = base + pricePerS-
quareMeter * squareMeters;

  return price;
};

const price =
getPrice(squareMeters);
// => 300000
```

Angenommen, der eigentlich erwartete Verkaufspreis liegt jedoch bei 400.000 Euro, dann lässt sich der Fehler wie folgt berechnen. Aus den zuvor bereits erwähnten Gründen wird der Fehler dabei quadriert:

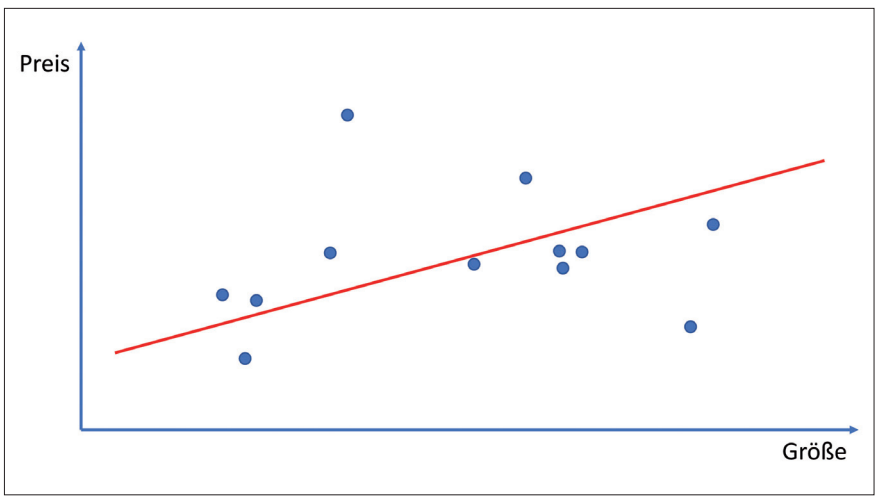


Abbildung 1

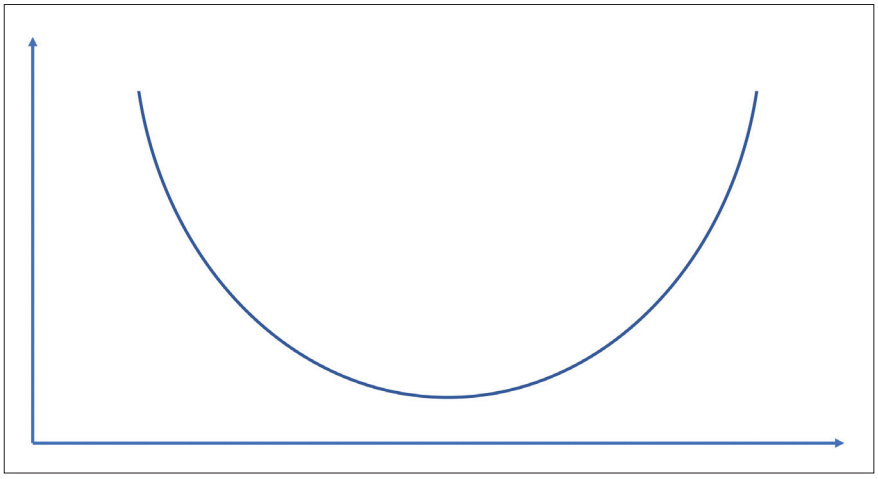


Abbildung 2

```
const expectedPrice = 400000;

const error = expectedPrice
- getPrice(squareMeters),
      errorSquared = Math.pow(error,
2);
```

Da das allerdings nicht nur für einen Preis, sondern für alle Preise der Trainingsdaten interessant ist, werden die einzelnen Fehler aufsummiert und der Durchschnitt gebildet. Das lässt sich wiederum als Funktion abbilden, die nun von zwei Parametern abhängt, nämlich dem y-Achsenabschnitt und der Steigung: Ändert man diese Parameter, verändert sich auch der durchschnittliche Fehler. Das Ziel ist nun, diesen durchschnittlichen Fehler zu minimieren.

Zum Finden eines Minimums einer Funktion lässt sich deren Ableitung verwenden, da die Ableitung die Steigung beschreibt. Man verschiebt also die Parameter in die Richtung des größten Abstiegs bezüglich der Steigung. Das klingt sehr abstrakt, lässt sich aber anschaulich darstellen. Da die Fehlerfunktion den quadratischen Fehler ermittelt, hat sie die Form einer verzerrten Parabel (siehe Abbildung 2).

Jeder Punkt auf dieser Parabel steht für eine andere Kombination von y-Achsenabschnitt und Steigung. Berechnet man die Ableitung der Fehlerfunktion, lässt sich an ihr ablesen, in welcher Richtung der Fehler kleiner wird (siehe Abbildung 3). Nun gilt es lediglich, neue Werte für den y-Achsenabschnitt und die Steigung zu berechnen, so dass sich die Ableitung der Fehlerfunktion in Richtung eines kleineren Fehlers bewegt, und den Vorgang dann so lange zu wiederholen, bis der Fehler minimal ist.

Dabei muss man aufpassen, keine zu großen Schritte zu machen, ansonsten könnte man über das Ziel hinausschießen. Das resultiert dann in einem Aufschaukeln des Fehlers, so dass er immer größer statt kleiner wird. Doch zu klein dürfen die Schritte auch nicht ausfallen, sonst dauert das Erreichen des Minimums nämlich zu lang. Daher muss man mit der sogenannten *Lernrate* experimentieren, die die Geschwindigkeit des Fortschritts definiert. Es empfiehlt sich, mit größeren Werten zu starten und diese dann nach und nach zu reduzieren.

Anstatt mit einer festen Lernrate lässt es sich auch mit einer variablen Lernrate arbeiten. Dazu existiert das Verfahren *Simulated Annealing*, das den Prozess des Abkühlens von heißem, flüssigem Metall nachbildet: Das geht zunächst sehr rasch, wird aber zunehmend langsamer. Wendet man einen solchen Algorithmus auf die Lernrate an, wird sie im Lauf der Zeit kleiner, so dass sich das Problem des Aufschaukelns vermeiden lässt.

KOMPLEXERE PROBLEME LÖSEN

Bis jetzt wurde die lineare Regression nur mit einer Variablen betrachtet. Doch was, wenn das Modell mehrere Variablen umfasst, der Verkaufspreis eines Hauses also beispielsweise nicht nur von dessen Größe, sondern auch dessen Baujahr abhängt? Die gute Nachricht ist, dass das Verfahren dann auf die gleiche Art funktioniert: Bei zwei Variablen erhält man allerdings keine zweidimensionale Kurve als Fehlerfunktion, sondern eine dreidimensionale gekrümmte Fläche. Das Finden des Minimums entspricht dann dem Abstieg vom Gipfel eines Berges in das umliegende Gelände.

Bei mehr als zwei Variablen entstehen entsprechend hochdimensionale Gebilde, die man sich nicht mehr bildlich vorstellen kann - das Verfahren an sich bleibt jedoch das Gleiche.

Auch andere Verfahren des maschinellen Lernens, unter anderem neuronale Netze, basieren auf dem Gradientenabstiegsverfahren. Auch dort dient das Verfahren dazu, den Fehler zu berechnen und zu minimieren, allerdings nicht über eine einzige Funktion, sondern ein ganzes Netzwerk von Funktionen. Selbst das derzeit so hoch gehandelte Deep-Learning basiert auf dem Verfahren, denn Deep-Learning ist nichts anderes als eine komplexe Variante von neuronalen Netzen.

Wie man sieht basieren die bislang vorgestellten Algorithmen zum maschinellen Lernen auf im Grunde sehr einfachen Algorithmen, die bereits seit Jahren und Jahrzehnten bekannt sind. Dass dem maschinellen

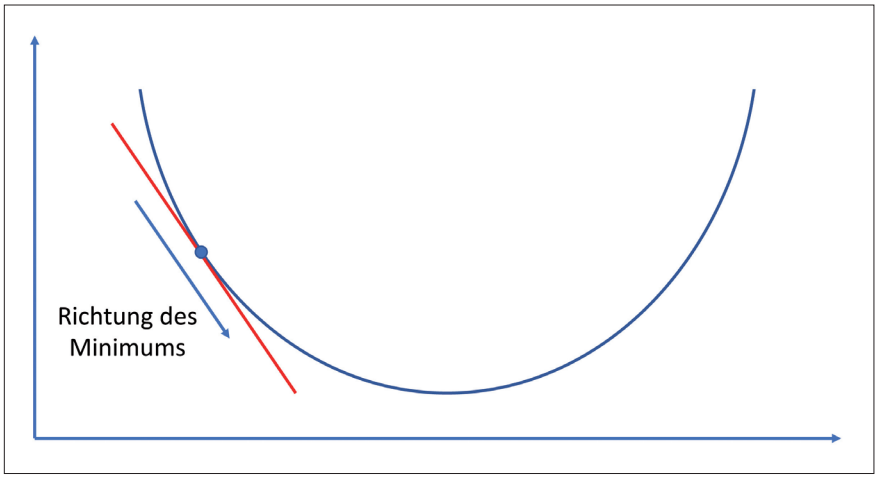


Abbildung 3

Lernen trotzdem erst seit einigen wenigen Jahren höhere Aufmerksamkeit zuteilwird, liegt vor allem daran, dass die häufig involvierten iterativen Berechnungen aufwändig sind, und bis vor einigen Jahren keine Computer zur Verfügung standen, die die Berechnungen in angemessener Zeit leisten konnten.

ANGST VOR DER SINGULARITÄT?

Zugleich zeigt das aber auch, wie wenig maschinelles Lernen mit „Lernen" im umgangssprachlichen Sinn gemein hat. Wie bereits eingangs erwähnt gibt es einen großen Unterschied zwischen dem Verfahren des Lernens an sich und der Frage nach Intelligenz. Um lernen zu können, muss man kein Bewusstsein oder eine höhere Intelligenz besitzen: Lernen beschreibt lediglich das Verbessern von Ergebnissen auf Basis von Erfahrungen.

In dem Zusammenhang interessant ist der Begriff der *Singularität*. Er steht in der Zukunftsforschung für den Zeitpunkt, ab dem Maschinen in der Lage sind, sich eigenständig zu verbessern und den technologischen Fortschritt damit dermaßen beschleunigen, dass die Zukunft der Menschheit danach nicht mehr vorhersagbar ist. Je nach persönlichem Standpunkt wird dieser Zeitpunkt entweder als Beginn eines goldenen Zeitalters oder der Apokalypse verstanden. Ob die Singularität allerdings jemals eintreffen wird, ist umstritten.

Das relativiert auch die derzeit häufig genannte Sorge, dass in naher Zukunft künstliche Intelligenz im Allgemeinen und maschinelles Lernen im Speziellen dafür verantwortlich sein werden, dass zahlreiche Arbeitsplätze entfallen. Das wird sicherlich passieren, aber nicht in allen Bereichen: Je weniger repetitiv eine Arbeit ist und je stärker sie von einer persönlichen Note des Ausführenden abhängt, desto schwieriger lässt sie sich durch eine Maschine erledigen.

Langfristig gesehen ist die Frage nach den Veränderungen, die solche Systeme auf die Gesellschaft haben werden, aber dennoch ein wichtiges Thema, das es zu diskutieren gilt.

FAZIT

Maschinelles Lernen umfasst neuronale Netze und Deep-Learning, besteht aber aus weitaus mehr. Es selbst ist wiederum ein Bestandteil des übergeordneten Themas der künstlichen Intelligenz. Die meisten der Algorithmen, die in dem Bereich zum Einsatz kommen, sind bereits seit Jahren und Jahrzehnten bekannt, wurden aber erst in den vergangenen Jahren in nennenswertem Umfang ausführbar, da zuvor keine entsprechend leistungsfähigen Computer zur Verfügung standen.

Das Lernen an sich wird im Kontext des maschinellen Lernens auf die Frage reduziert, wie sich ein Ergebnis auf Grund von Erfahrungen durch eine Maschine autonom verbessern lässt. Die dazu verwendeten Algorithmen können entweder mit oder ohne menschliches Feedback lernen, je nach Aufgabenstellung. Allen gemein ist aber, dass sie kein Bewusstsein haben, sondern rein mathematischen Regeln folgen.

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.

👎 **Verdammte 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.

👎 **So ein Scheiß!**
👍 Ich liebe Herausforderungen.

👎 **Dieser Trottel baut einen Blödsinn nach dem anderen!**
👍 Möglicherweise haben wir noch nicht die richtige Position für ihn gefunden.

👎 **Wir sind am Arsch!**
👍 Die Produktivitätsindizes unseres Unternehmens zeigen einen sensiblen Rückgang.

👎 **Ich habe von Anfang an gewusst, dass alles ein Blödsinn ist!**
👍 Verzeihung, ich hätte Sie warnen können, wenn man mich gefragt hätte.

👎 **Du Volltrottel kennst dich überhaupt nicht aus!**
👍 Das ist nicht gerade Ihre Kernkompetenz, habe ich Recht?

WISSENSWERTES ÜBER HTTP/2

Quelle: Tim Reckmann / pixelio.de

Es ist nun bereits über ein Jahr her, dass HTTP/2 offiziell verabschiedet wurde. Dieser Beitrag informiert, wie es dazu kam, welche Vorteile HTTP/2 mit sich bringt und was es für Surfer und Entwickler zu berücksichtigen gilt.

Autor: WALTER SAUMWEBER

In erster Linie soll das neue HTTP-Protokoll eine Beschleunigung des Internets herbeiführen. Wenn man die Voraussetzungen von heute mit denen von früher vergleicht, wird deutlich, dass dies notwendig ist.

SURFGESCHWINDIGKEIT – FRÜHER UND HEUTE

Während früher zwar nicht die ausgefeilten Webtechniken und die leistungsfähige Hardware von heute zur Verfügung standen, mussten die Browser von den Servern aber fast nur Text und allenfalls ein paar Grafiken anfordern. Dagegen wimmelt es heutzutage auf den Webseiten von Bildern, Videos, Werbeeinblendungen, Flash-Inhalten usw.. Und tatsächlich fühlt man sich beim Surfen manchmal an die 90er-Jahre zurückerinnert, wenn man die teilweise langen Ladezeiten von Webseiten ertragen muss, oder auch wie behäbig sich die Interaktion auf manchen Seiten gestaltet. Das liegt daran, dass für jedes eingebundene Element (z. B. auch für im HTML-Code referenzierte Skript- oder Style-Sheet-Dateien) eine separate GET-Anforderung erforderlich ist. So verhält es sich jedenfalls unter dem Hypertext Transfer Protocol 1.1 (kurz: HTTP/1.1), das seit 1999 im Einsatz ist. In den Jahren davor war es unter Verwendung von HTTP/1.0 sogar noch so, dass eine TCP-Verbindung nach jeder Anfrage geschlossen, also bei der nächsten wieder neu aufgebaut werden musste. Zwar kann unter HTTP/1.1 eine TCP-Verbindung grundsätzlich für mehrere Requests verwendet werden. Allerdings werden separate TCP-Verbindungen benötigt, wenn die Elemente von verschiedenen Hosts angefordert werden müssen (Beispiel: Eine referenzierte

Grafik oder eine Skriptdatei liegen auf einem anderen Server als dem, der die HTML-Datei index.html beherbergt). Erfolgt ein Request seitens des Client, während der vorausgegangene noch in Arbeit ist, dann wird ebenfalls eine weitere TCP-Verbindung geöffnet, auch wenn es sich um denselben Host handelt (das so genannte Pipelining – mehrere laufende Anfragen über die gleiche TCP-Verbindung – ist in den Browsern aus Sicherheitsgründen standardmäßig deaktiviert). Da Clients unter HTTP/1.1 nur eine beschränkte Anzahl von parallelen TCP-Verbindungen unterhalten dürfen, stellen sich entsprechende Wartezeiten ein.

ENTWICKLUNGSSTADIEN VON HTTP/2

Abhilfe soll nun das neue Protokoll schaffen. Die Internet Engineering Task Force (IETF) hat den HTTP/2-Standard als RFC 7540 [1] sowie die damit verbundene Headerkompression als RFC 7541 [2] veröffentlicht. Die ersten Diskussionen im Hinblick auf eine Weiterentwicklung, des bis dato verwendeten Hypertext Transfer Protocol, begannen bereits nach seiner Einführung. Konkret wurde das Ganze aber erst, als Google das Netzwerkprotokoll SPDY (ausgesprochen: speedy) als Alternative zu HTTP/1.1 entwickelte. SPDY zeichnete sich vor allem durch Multiplexing (mehrere Datenpakete können über eine einzelne TCP-Verbindung parallel übertragen werden), das Priorisieren von Anfragen und das Server-Push-Verfahren aus. Letzteres bedeutet, dass ein Server die Übertragung von Inhalten ohne Client-Anfrage selbst anstoßen kann. Die Analyse erfolgt anhand des HTML-Quellcodes und des Zugriffsverhaltens. Mit anderen Worten: Der Server sendet Daten, von denen er weiß, dass der Client sie benötigt, aus eigener Initiative und erspart dem Client damit zusätzliche Requests. Die SPDY-Implementation seitens Google Chrome, Mozilla Firefox und Opera folgte auf dem Fuß. Serverseitig wurde SPDY unter anderem von Apache unterstützt. Schließlich nahm sich die für Internetstandards zuständige Organisation Internet Engineering Task Force (IETF) der Weiterentwicklung des HTTP-Protokolls an. Dabei wurde zwar von SPDY ausgegangen, jedoch zahlreiche Änderungen und Verbesserungen eingefügt. Im Ergebnis unterscheidet sich HTTP/2 (die ursprüngliche Bezeichnung der Versionsnummer war 2.0) deutlich von SPDY, die oben genannten Konzepte wurden jedoch – unter Behebung von aufgetretenen Mängeln, einschließlich potentieller Sicherheitsrisiken – übernommen. Die folgende Tabelle zeigt die wichtigsten Abschnitte der HTTP/2-Entwicklung (Quelle: [3]; die Meilensteine sind in englischer Sprache belassen).

Neuerung	*Beschreibung*
Binärformat mit Frame-Headern	Ermöglicht, den Datenverkehr in Streams zu unterteilen. Jeder Stream erhält eine eindeutige Stream-ID.
Header-Kompression	Vermeidet Overhead durch Metadaten
Multiplexing	Mehrere Datenpakete können über eine einzelne TCP-Verbindung parallel übertragen werden.
Priorisieren von Requests	Stellt sicher, dass wichtige Elemente möglichst schnell und vollständig übertragen werden, z. B. Text vor Grafiken.
Server-Push	Server kann die Übertragung von Inhalten ohne Client-Anfrage anstoßen.

Zeitpunkt	*Milestone (engl.)*
20.12.2007	First HTTP 1.1 Revision Internet Draft
23.01.2008	First HTTP Security Properties Internet Draft
Anfang 2012	Call for Proposals for HTTP 2.0
14.10.2012 bis 25.11.2012	Working Group Last Call for HTTP 1.1 Revision
28.11.2012	First WG draft of HTTP 2.0, based upon draft-mbelshe-httpbis-spdy-00
Nicht abgeschlossen/verworfen	Working Group Last Call for HTTP Security Properties
September 2013	Submit HTTP 1.1 Revision to IESG for consideration as a Proposed Standard
12.02.2014	IESG approved HTTP 1.1 Revision to publish as a Proposed Standard
06.06.2014	Publish HTTP 1.1 Revision as RFC 7230, 7231, 7232, 7233, 7234, 7235
01.08.2014 bis 01.09.2014	Working Group Last call for HTTP/2
16.12.2014	Submit HTTP/2 to IESG for consideration as a Proposed Standard
31.12.2014 bis 14.01.2015	IETF Last Call for HTTP/2
22.01.2015	IESG telechat to review HTTP/2 as Proposed Standard
17.02.2015	IESG approved HTTP/2 to publish as Proposed Standard
14.05.2015	Publish HTTP/2 as RFC 7540

WEITERE VORTEILE VON HTTP/2

Neben den bereits genannten besitzt HTTP/2 noch weitere Vorzüge. So verwendet es nicht mehr das Textformat, sondern ein Binärformat mit Frame-Headern, wobei jeder Frame eine eigene Stream-ID enthält. Dadurch können mehrere Ressourcen auf einmal verschickt werden. Parallel dazu können auf einem speziellen Stream Client und Server Kontrollinformationen und Requests austauschen. Die Datenkompression schließt in HTTP/2 dank eines speziellen Algorithmus (HPACK) nun auch die Header der Datenpakete mit ein, was den Overhead reduziert. Die folgende Tabelle enthält eine kurze Zusammenfassung der wesentlichen Neuerungen des HTTP/2-Protokolls.

Von der Leistungsfähigkeit des neuen HTTP-Protokolls können Sie sich auf der Webseite https://http2.golang.org überzeugen (Abbildung 1). Dabei handelt es sich um einen Testserver für die HTTP2-Unterstützung der von Google entwickelten Programmiersprache Go. Falls die Verknüpfungen zu den Tests nicht erscheinen, bietet Ihr Browser keine HTTP/2-Unterstützung bzw. diese ist nicht aktiviert.

Inzwischen unterstützen die aktuellen Browserversionen der bekannten Hersteller HTTP/2, Internet

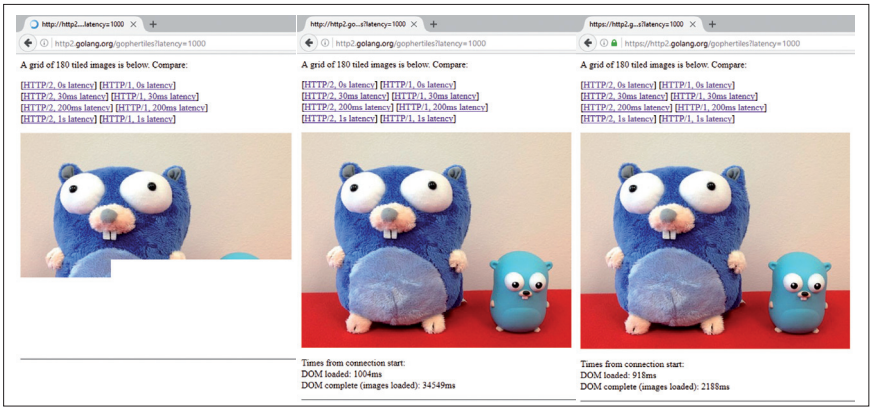


Abbildung 1: Test (https://http2.golang.org/gophertiles) im Firefox-Browser: Unter Verwendung von HTTP/2 (rechtes Bild) baut sich die Grafik deutlich schneller auf.

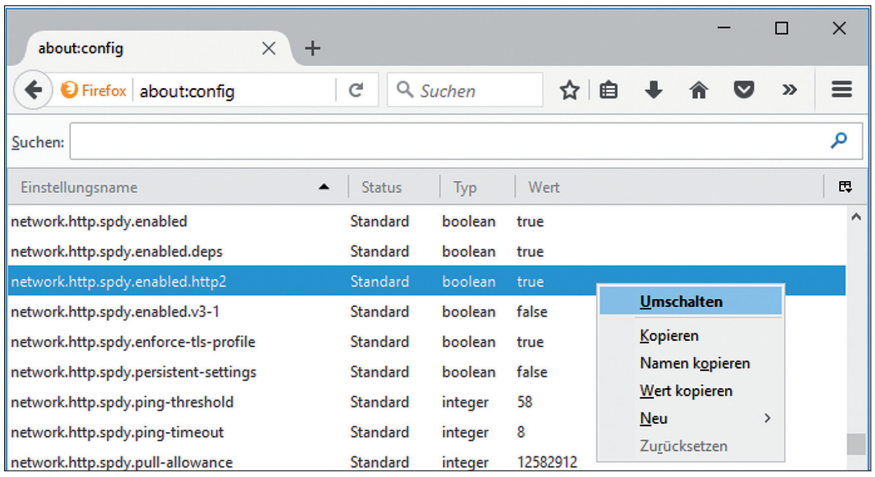


Abbildung 2: Im Firefox-Browser lässt sich die HTTP/2-Unterstützung nach Belieben aus- und wieder einschalten – standardmäßig ist sie eingeschaltet.

Explorer 11 allerdings erst mit Windows 10 (der neue Browser Microsoft Edge unterstützt hier ebenfalls HTTP/2). In Mozilla Firefox lässt sich HTTP/2 sogar nach Belieben ein- und ausschalten. Geben Sie about:config in die Adressleiste des Browsers ein. Zuständig ist die Einstellung network.http.spdy.enabled.http2 (Abbildung 2). Wenn in der Wert-Spalte true steht, ist HTTP/2 aktiviert. Ein Doppelklick auf die Zeile deaktiviert HTTP/2, mit einem weiteren Doppelklick können Sie die Unterstützung des HTTP/2-Protokolls wieder aktivieren. Alternativ verwenden Sie den Befehl Umschalten im Kontextmenü (bei ausgeschalteter HTTP/2-Unterstützung schaltet auch der Befehl Zurücksetzen HTTP/2 wieder ein). Eventuell muss der Browser neu gestartet werden, bevor die Änderung zum Tragen kommt.

Gibt es nun gegenüber den genannten Vorteilen auch Nachteile?

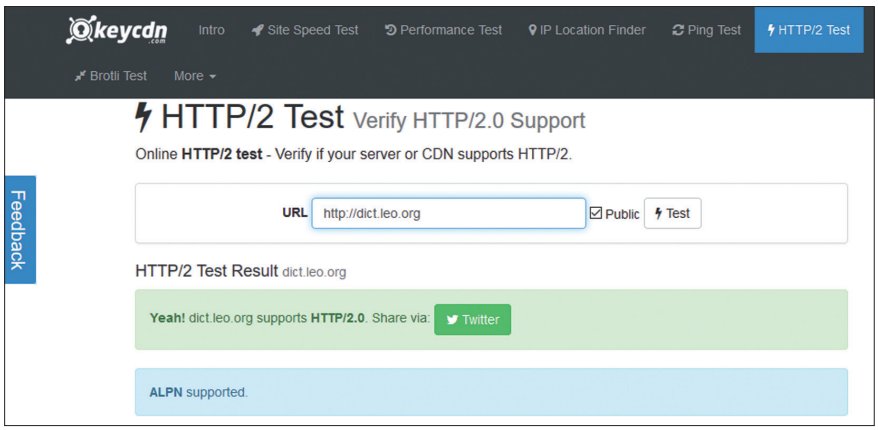


Abbildung 3: Auf der Seite **https://tools.keycdn.com/http2-test** können Sie testen, ob Ihr Provider oder eine bestimmte Website HTTP/2 unterstützt.

Die Frage kann mit einem klaren Nein beantwortet werden. Zwar gab es während der Entwicklungsphase hier und da Bedenken bezüglich eines erhöhten Sicherheitsrisikos. In der Praxis ist nun aber das Gegenteil der Fall. Obwohl HTTP/2 grundsätzlich auch unverschlüsselte Verbindungen zulässt, bieten die Browser HTTP/2 durchweg nur für verschlüsselte Seiten an.

WAS IST ZU BEACHTEN UND WIE SIEHT DIE ZUKUNFT AUS?

HTTP/2 unterstützt alle wesentlichen Features von HTTP/1.1, ist aber darauf ausgerichtet, effizienter zu arbeiten. Für den Surfer ändert sich gegenüber HTTP/1.1 grundsätzlich nichts, außer dass er mit schnelleren Ladezeiten rechnen kann – das gilt vor allem bei Seiten mit viel Content (Bilder, Videos, etc.). Dies setzt allerdings voraus, dass HTTP/2 nicht nur vom verwendeten Client (Browser), sondern auch von den angeforderten Webseiten umgesetzt wird. Handlungsbedarf besteht daher noch bei den Webseitenbetreibern bzw. den Webhosting-Anbietern. Derzeit wird auf den meisten Servern noch HTTP/1.1 eingesetzt, was sich jedoch in absehbarer Zeit ändern dürfte. Die Umstellung sollte kein allzu großes Hindernis darstellen, da praktisch alle bestehenden Programmierschnittstellen – gegebenenfalls mit erweiterter Funktionalität – auch für HTTP/2 verwendet werden können. Im Übrigen ist HTTP/2 abwärtskompatibel, daher können Browser ohne HTTP/2-Unterstützung auch über HTTP/1.1 auf HTTP/2-Seiten zugreifen und umgekehrt. Wenn eine angeforderte Webseite kein HTTP/2 anbietet oder es sich um keine sichere Verbindung handelt, schalten die HTTP/2-fähigen Browser automatisch auf die Protokollversion 1.1 um.

LINKS UND QUELLEN

[1] https://tools.ietf.org/html/rfc7540

[2] https://tools.ietf.org/html/rfc7541

[3] https://en.wikipedia.org/wiki/HTTP/2

SCRUM COMMITMENTS DÜRFEN NICHT ERFÜLLT WERDEN

Ziele sind NICHT zum Erreichen da, schreibt Conny Dethloff [1]. Und Elon Musk will bis 2025 Menschen auf den Mars bringen [2]. Beides gefällt mir.

Autor: RALF WESTPHAL

Ob und wann 1.000.000 Menschen auf dem Mars leben werden, ist nicht so wichtig. Es geht darum, welchen Sog diese Vision ausübt. Wenn wir uns darauf einlassen, kann Kohärenz entstehen. Kräfte werden gebündelt. Das macht an sich schon Freude. Und was man am Ende bzw. auf dem Weg erreicht, ist Fortschritt; der macht auch Freude. (Dass es auch andere Visionen gibt, die verfolgenswert sind, ist klar. Nicht „entweder ...oder", sondern „sowohl ... als auch". Auf den Mars und Schluss mit Plastikmüll usw.)

Wer alles haarklein so erfüllt sehen will, wie Elon Musk es beschreibt, der engt sich selbst ein.

Übertragen auf die Softwareentwicklung bzw. noch konkreter auf das beliebte Vorgehensmodell Scrum bedeutet das:

Es geht eben nicht darum, dass ein Sprint Commitment erfüllt wird. Die Punktlandung zum Sprintende mit allen User Stories im Sprint Backlog durch zu sein (done), ist nicht das, worum es geht.

Ansonsten *„werden alle [Entwickler] ihre ganze Energie darin legen, [das Commitment zu erfüllen] und dafür unter Umständen auch [Qualitäten] herunterschrauben, unabhängig davon, ob diese dann noch einen Mehrwert darstellen oder nicht.“*

Ganz plakativ und extrem bedeutet das: Das Commitment für den nächsten Sprint, sollten alle (!) User Stories im Backlog sein.

Sehr wahrscheinlich ist das ein nicht erreichbares Ziel. Aber das ist egal, weil letztlich nie klar ist, ob ein Commitment wirklich, absolut erfüllt werden kann. Das ist immer nur Wunschdenken. Das ist immer nur eine Wette – die der Erfahrung nach meistens verloren wird; zumindest, wenn man genau hinschaut.

SCHRITT FÜR SCHRITT AUF DEM WEG ZUM ZIEL

Aber nehmen wir das Commitment für einen Moment mal hin. Das Ziel ist nicht erreichbar: Was nun? Man muss sich auf den Weg machen. Man muss einen Schritt tun. Und dann macht man noch einen Schritt. Schritt für Schritt geht man voran. So gut man eben kann. Zügig, aber sorgfältig.

Irgendwann ist man entweder fertig – oder die Zeit ist um.

Was auch sonst?

So sind Sie wahrscheinlich in der Schule bei Klausuren auch vorgegangen. Der Lehrer hat für die angesetzten 90 Minuten fünf Aufgaben vorgelegt. Sie haben die Augen aufgerissen und nicht gewusst, ob Sie das alles in der Zeit schaffen werden. Dann haben Sie mit einer Aufgabe einfach begonnen. Vielleicht war es die scheinbar leichteste, vielleicht umgekehrt die scheinbar schwierigste. Egal, ob Sie zuerst priorisiert haben oder nicht, Sie haben überhaupt nur mit genau einer Aufgabe beginnen können. Danach die nächste und danach die nächste usw.

Nicht anders, so meint Ralf Westphal, sollte es bei der Softwareentwicklung sein. Wer in einer Planungssitzung steckt, darf sich gern einen tollen Plan machen und ein Commitment abgeben. Nur er sollte nicht darauf schielen, den Plan zu erfüllen. Der Plan dient lediglich einer Ausrichtung auf einen Weg. Er bündelt Kräfte, stellt Kohärenz her. Und dann... dann macht man nur den ersten Schritt. „That’s it.“

Nach dem ersten Schritt kommt eine Orientierung mit der Frage: „Wo bin ich? Was ist jetzt angesagt?“ und dann der nächste Schritt, der zur aktuellen Lage passt. Der mag dem Plan entnommen sein oder er ist ad hoc auf die veränderte Situation angepasst.

Eigentlich gibt es immer nur erste Schritte. Nach dem Schritt ist vor dem Schritt. Keiner unterscheidet sich vom anderen. Denn alle sind gleich in dem, dass sie im jeweiligen Moment die sind, die gemacht werden sollten, um möglichst viel Wert herzustellen.

Das Ziel tritt in den Hintergrund. Der Blick richtet sich nur auf den durch das Ziel ausgerollten Weg. Man folgt der „gelben Ziegelsteinstraße“, ohne zu wissen, wann (und ob) man die smaragdene Stadt aus Frank L. Baums Geschichte „Der Zauberer von Oz“ erreichen wird.

Das mag paradox klingen. Aber – wie auch Dethloff sagt – Paradoxie gehört zum Leben. Wer sich auf die Paradoxie einlässt, ist insofern lebendig. Erfolg wird dann nicht mehr kontrolliert und erzwungen, sondern emergiert.

Fußnoten:

[1] http://www.lean-knowledge-base.de/ziele-sind-nicht-zum-erreichen-da/

[2] http://waitbutwhy.com/2016/09/spacexs-big-fking-rocket-the-full-story.html

Testabdeckung mit NCover

Autor: FABIAN DEITELHOFF

Mittels Testabdeckung Fehlern auf die Schliche kommen

Mittlerweile sind Unit Tests in der Softwareentwicklung eine feste Größe. Dennoch ist das Schreiben von Tests häufiger schwieriger als gedacht. Denn wie wird entschieden, ob ein Test gut oder schlecht ist? Ein erfolgreicher Test kann eine Banalität überprüfen, was einer Nullaussage gleichkommt. Die Testabdeckung spielt als Metrik innerhalb der Qualitätssicherung eine große Rolle. NCover hilft dabei, Daten zur Testabdeckung systematisch auszuwerten.

Wer erinnert sich noch an die Quizshow für Kinder 1, 2 oder 3 [1], die im ZDFtivi, KiKA und ORF eins ausgestrahlt wird. Ja, richtig gelesen! Die Show wird seit Anfang Dezember 1977 jeden Samstag ausgestrahlt und läuft tatsächlich auch heute noch. Fester Bestandteil dieser Show ist unter anderem der Satz „Ob ihr wirklich richtig steht, seht ihr, wenn das Licht angeht." Damit wird den teilnehmenden Kindern verraten, ob die von ihnen gewählte Antwort korrekt ist.

Manchmal scheint es so, als hätten die heutigen Softwareentwickler und -Entwicklerinnen diese Show in ihrer Kindheit gesehen. So stark fokussiert sind viele auf die grünen Lämpchen in ihrer Entwicklungsumgebung, die anzeigen, ob ein Test erfolgreich durchgelaufen ist oder nicht. Diese visuelle Kennzeichnung ist auf jeden Fall wichtig, bitte nicht falsch verstehen. Dennoch zeigt sie lediglich an, dass kein Fehler vorliegt.

DAS PROBLEM...

Problematisch an der ganzen Sache ist, dass ein Test eine Banalität abprüfen kann oder gar gänzlich an der essentiellen Sache vorbeitestet. Ein grüner Testfall garantiert somit nicht, dass das richtige getestet wurde. Es wird damit nur visualisiert, dass der aktuell vorhandene Code in einem konkreten Test ein erwünschtes Verhalten zeigt. Erwünscht in dem programmierten Sinne, da die erwarteten Ergebnisse ebenfalls im Testfall hinterlegt sind.

Ob die zu einem Zeitpunkt x vorhandenen Testfälle y aber alle relevanten beziehungsweise kritischen Bereiche einer Anwendung überprüfen, wird in keiner Weise angezeigt. Es ist noch nicht einmal sicher, ob die relevanten Stellen innerhalb einer Unit, zum Beispiel einer Klasse oder Methode, abgedeckt sind. Das führt zur der Erkenntnis, dass es zahlreiche Unit Tests gibt, die für das Auffinden von Fehlern mehr oder weniger irrelevant sind. Im schlimmsten Fall erkennen sie niemals einen Fehler in

INFOS ZUR TESTABDECKUNG

Die Testabdeckung lässt sich als Ausmaß der Testfälle definieren, mit denen eine Anwendung oder ein Teil davon getestet werden muss, um einen bestimmten Überdeckungsgrad zu erreichen. Es werden die Überdeckungen C0 bis C7 unterschieden, die alle einen unterschiedlichen Überdeckungsgrad definieren. Am gebräuchlichsten sind die Überdeckungen C0, C1 und C2.

C0: Jede Anweisung wird mindestens einmal durchlaufen.

C1: Jeder Programmzweig wird mindestens einmal durchlaufen.

C2: Bei zusammengesetzten Bedingungen in bedingten Anweisungen wird jede Bedingung (true, false) getestet.

Abbildung 1: Die Installation von NCover Desktop inklusive Visual Studio Erweiterung.

der Anwendung, versprühen aber die trügerische Sicherheit eines grünen Testfalls. Im besten Fall können sie, mehr oder weniger aus Zufall, einen Fehler in der Anwendung anzeigen, für dessen Aufdeckung sie im Grunde niemals geschrieben worden sind.

...UND EINE MÖGLICHE LÖSUNG

Fehlerhafte Testfälle zu überprüfen und anschließend zu verändern, ist Aufgabe der Qualitätssicherung. Oder andersherum ausdrückt ist es Aufgabe der Qualitätssicherung, dass solche fehlerhaften Testfälle erst gar nicht entstehen. Das ist sicherlich der beste Fall. Innerhalb der Qualitätssicherung kommen Metriken zum Einsatz, die angeben, wie gut oder schlecht ein Testfall ist. Eine Metrik ist die sogenannte Testabdeckung [2] (Englisch: Code Coverage), die das Verhältnis an tatsächlich getroffenen Aussagen eines Tests gegenüber den theoretisch möglichen Aussagen angibt. Ein paar zusätzliche Infos gibt es im Kasten zur Testabdeckung.

Zusammengefasst gesagt, zeigt die Testüberdeckung an, welche Bereiche eines Code-Abschnitts getestet (abgedeckt) wurden und welche nicht. Es liegt in der Natur der Dinge, dass es Tests gibt, die auf magische Weise einen erheblichen Umfang des Quelltextes abdecken. Scheinbar mit Leichtigkeit. Von diesen Tests gibt es zahlenmäßig mehr als vielleicht zu vermuten wäre. Das liegt an der Tatsache, dass es viele Bereiche der eigenen Anwendung gibt, die von verschiedenen Stellen aus durchlaufen werden. Etwas technischer ausgedrückt bedeutet das, dass es Code-Zeilen gibt, die häufiger im Einsatz sind als andere. Kritisch sind aber die Randbedingungen, die Ausnahmen, die sogenannten Edge Cases. Also ebenjene Code-Stellen, die kaum oder gar nicht durchlaufen werden, wenn nicht explizit dafür Testfälle geschrieben werden.

KRITISCHE STIMMEN

Testabdeckung wird häufig als der heilige Gral verstanden und von vielen Stellen auch so angepriesen. Das ist genauso falsch wie diese Metrik als völlig absurd und überflüssig abzustempeln. Wie so oft liegt die Wahrheit irgendwo dazwischen.

Die Testabdeckung muss unter zwei Gesichtspunkten betrachtet werden. Sie ist gut dafür geeignet, zu

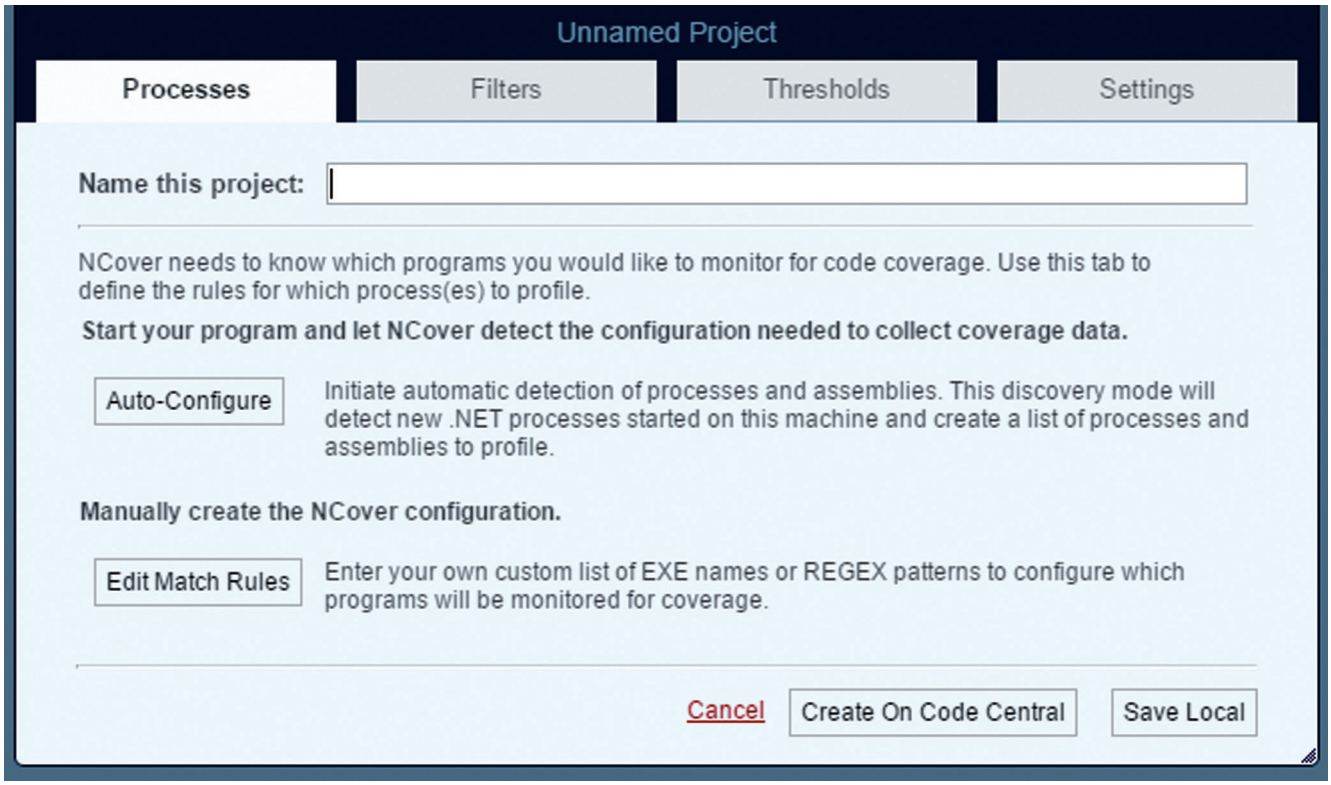


Abbildung 2: Der Dialog zum Anlegen eines neuen NCover-Projekts.

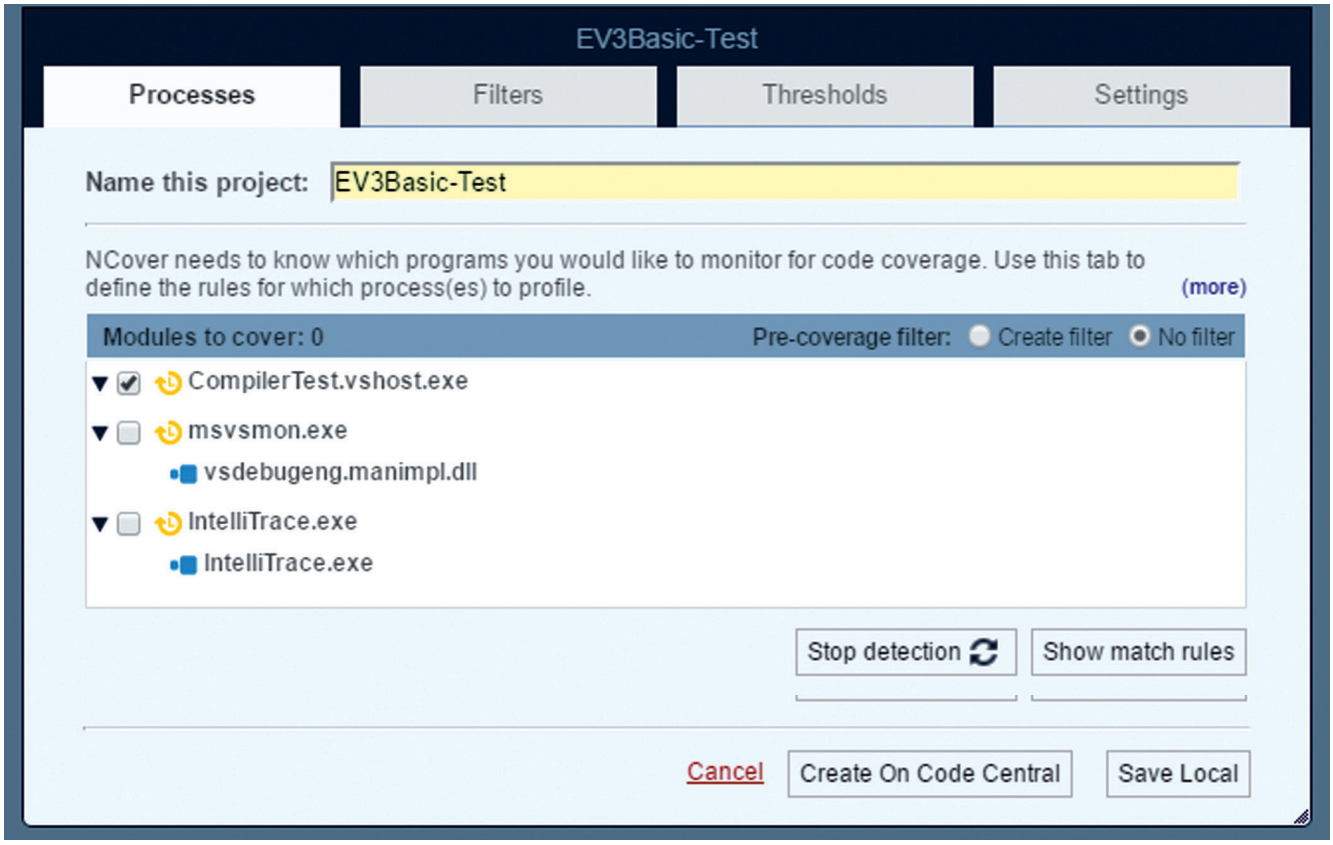


Abbildung 3: Die Auswahl der von NCover zu überwachenden Module.

einem Zeitpunkt x nicht durch Tests abgedeckten Code aufzufinden. Die so gefundenen Stellen können anschließend mit zusätzlichen oder veränderten Tests abgedeckt werden. Die Testabdeckung als Qualitätsziel zu definieren, ist allerdings keine gute Idee oder zumindest schwierig. Denn wie hoch soll diese Testabdeckung sein? Reichen 80%, müssten es 90% oder gar 100% sein? Letzteres ist nur durch eine hohe Anzahl von Testfällen möglich, deren Testaussagen mindestens zweifelhaft sind. Auf der anderen Seite ist eine hohe Testabdeckung teilweise recht schnell erreichbar. In vielen Fällen durch qualitätsmäßig zweifelhafte Testfälle. Das ist kein gutes Ziel, dass es zu erreichen gilt.

Nicht zu vergessen, dass eine Testabdeckung von 100% kein Garant für perfekten Code ist. Denn die Überdeckung misst nicht zwingend die Qualität der Testfälle. Es gibt durchaus Code, der durch einen einzelnen sehr simplen Testfall eine sehr hohe Testabdeckung erreicht. Manchmal sogar schon 100%. Diese Testfälle sind häufig aber weder aussagekräftig noch ist ein einzelner Testfall ausreichend, um über die Qualität des getesteten Codes eine Aussage zu treffen.

NCOVER IM ÜBERBLICK

Eine Möglichkeit, die Testabdeckung von Unit Tests zu überprüfen, ist NCover [3]. Bevor der Artikel näher auf diese spezifische Auswahl eingeht, sei gesagt, dass es nur eine Möglichkeit von vielen ist. Nicht nur für .NET gibt es zahlreiche Alternativen, auch in anderen Sprachen haben sich über die Jahre eine breite Palette an Tools zur Messung der Codeabdeckung etabliert. Für .NET als Zielplattform sind unter anderem JetBrains dotCover [4] und PartCover [5] beziehungsweise das daraus hervorgegangene Projekt OpenCover [6] als Alternativen zu nennen.

Der Name NCover lässt es bereits erahnen, dass es sich um ein Tool spezifisch für die Aufgabe der Testabdeckung bei .NET Anwendungen handelt. Das Tool wurde im Jahr 2003 von Peter Waldschmidt initiiert und ist nach einigen Jahren in der NCover, LLC aufgegangen. Dort wird die Software weiterentwickelt und kommerziell vertrieben.

Über die Jahre hat sich NCover zu einem vollumfassenden Produkt weiterentwickelt, das zahlreiche Features und Services rund um die Testabdeckung von .NET Anwendungen anbietet. NCover wird in verschiedenen Versionen angeboten. Die *Desktop*-Variante zielt auf Entwickler ab, die während der Implementierung Informationen brauchen, um zum Beispiel bessere Tests zu schreiben. Die *Code Central* genannte Variante zielt auf ein gesamtes Team ab. Zum Beispiel um Daten zur Testabdeckung von verschiedenen Entwicklern zu sammeln und aufzubereiten. Zu guter Letzt erweitert die *Collector Version* Code Central unter anderem zu Build Servern, um auch dort Daten über die Testabdeckung zu sammeln.
Im Folgenden konzentriert sich der Artikel ausschließlich auf die Desktop-Variante. Getestet wurden die beschriebenen Features mit der 21 Tage frei nutzbaren Trial-Version.

INSTALLATION

NCover wird klassisch über eine ausführbare Datei installiert. Bei der Desktop-Variante kann zusätzlich eine Erweiterung für die verschiedensten Visual Studio Versionen installiert werden, wie Abbildung 1 zeigt. Das

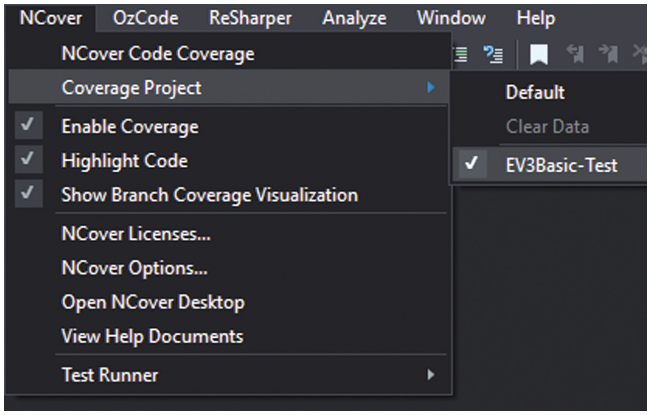


Abbildung 4: Das Projekt für die Datenaufzeichnung in der Visual Studio Erweiterung einstellen.

erlaubt es, NCover nicht nur als externes Tool zur Analyse zu nutzen, sondern auch direkt in Visual Studio Daten zu sammeln.

NCover startet nach der Installation auf der lokalen Maschine ein Prozess, der auf Localhost und standardmäßig dem Port 11235 hört. NCover wird primär über eine Web-Oberfläche bedient. Dort können Projekte verwaltet und die gesammelten Daten eingesehen werden.

EIN NEUES PROJEKT

NCover das Überwachen eines neuen Projekts mitzuteilen, ist zu Beginn etwas gewöhnungsbedürftig. Abbildung 2 zeigt den Dialog, um ein Projekt anzulegen. Der Name ist obligatorisch. NCover ermittelt die zu überwachende Anwendung anhand eines laufenden Prozesses, der zur Anwendung gehört. Welcher Prozess das ist, kann entweder automatisch von NCover ermittelt oder durch manuelle Regeln angegeben werden. Die Auto-Konfiguration geht dabei so vor, dass .NET-Prozesse, die auf der lokalen Maschine neu gestartet werden, erfasst und in einer Liste dargestellt werden. Bei den manuellen Regeln kann zum Beispiel der Dateiname angegeben werden.

In diesem Beispiel soll das Projekt mit Namen EV3Basic überwacht werden. Der relevante Prozess trägt den Namen CompilerTest.vshost.exe, da es sich um eine Konsolenanwendung handelt. Abbildung 3 zeigt die Liste mit erfassten Prozessen auf der lokalen Maschine. Abschließend lässt sich die neue Konfiguration lokal abspeichern oder auf einen Code Central-Server übertragen.

Es besteht zudem die Möglichkeit, zu einem Projekt diverse Einstellungen vorzunehmen. Zum Beispiel

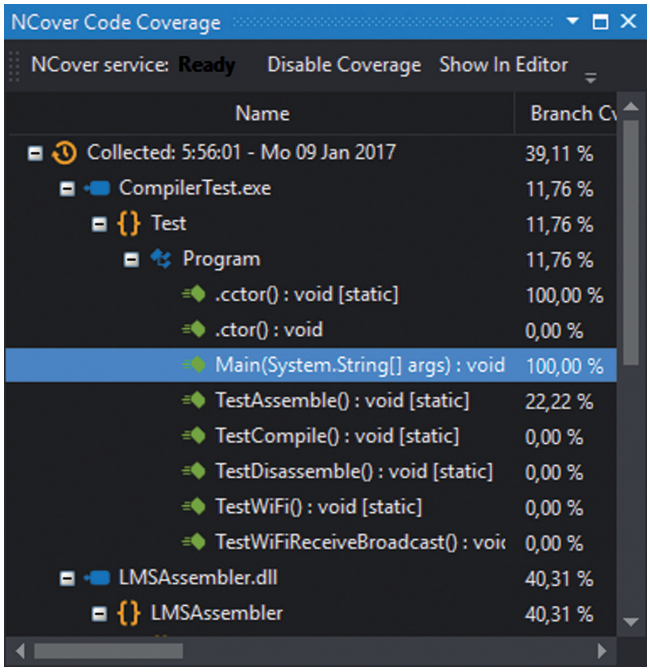


Abbildung 5: Die überdeckten Methoden in Tool-Windows von NCover.

können Module, Namensräume, Methode oder gar Eigenschaften von der Überwachung ausgenommen werden. Auch Schwellwerte können festgelegt werden, die angeben, wann Daten als zufriedenstellend oder nicht zufriedenstellend gelten sollen. Das beeinflusst die farbliche Markierung beispielsweise bei den Reports.

DATEN SAMMELN UND VISUALISIEREN

Nachdem das Projekt existiert, kann es in Visual Studio über die NCover Erweiterung als Ziel-Projekt definiert werden (siehe Abbildung 4). Von nun an landen alle gesammelten Daten in diesem Projekt, statt in einem anonymen Standardprojekt, was NCover bei der Installation anlegt.

Wichtig ist, dass die Überwachung der Testabdeckung in NCover aktiviert ist, da ansonsten keinerlei Daten erfasst werden. Das geht entweder über den bereits gezeigten Menüpunkt zu NCover (siehe Abbildung 4) oder über das eigenständige Tool-Window zu NCover, zu sehen in Abbildung 5.

Dieses Tool-Window zeigt zudem die gesammelten Daten an. Heruntergebrochen bis auf die Ebene der

```
class Program
{
    private static string smallBasicInput = @"D:\Dropbox\EV3 ASM\LMS-Tests\test3.sb";
    private static string smallBasicOutput = @"D:\Dropbox\EV3 ASM\LMS-Tests\subcall-mit-data8-data16.lms";
    private static string smallBasicBinaryOutput = @"D:\Dropbox\EV3 ASM\LMS-Tests\subcall-mit-data8-data16.rbf";

    0 references | cOpperdragon, 428 days ago | 1 author, 11 changes
    static void Main(string[] args)
    {
        //            TestWiFiReceiveBroadcast();
        //            TestWiFi();
        //TestCompile();
        TestAssemble();
        //TestDisassemble();
    }
```

Abbildung 6: Eine überdeckte Methode mit farblichen Markierung der Überdeckung.

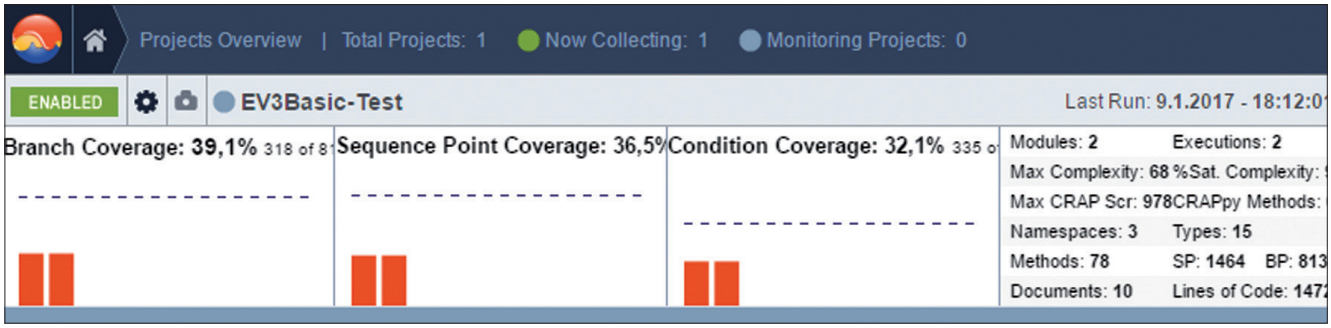


Abbildung 7: Die Durchläufe als Einträge im Browser von NCover Desktop.

Methoden, zu denen über das Fenster navigiert werden kann. Die so gesammelten Testdaten lassen sich nach den Durchläufen direkt in Visual Studio anzeigen. Dafür werden verschiedene Farben genutzt, um durchlaufende und nicht durchlaufende Bereiche voneinander abzugrenzen. Abbildung 6 zeigt das für die main-Methode des Beispielprojekts, dass von NCover eine Testabdeckung von 100% attestiert bekommen hat. So etwas erfreut natürlich das Entwicklerherz, beim genauen Blick in die Methode wird aber schnell klar, dass lediglich eine weitere Methode aufgerufen wird. Ein weiteres Beispiel für eine trügerische Sicherheit und dass selbst eine vollständige Testabdeckung nicht zwingend die allergrößte Aussagekraft hat.

Abbildung 7 zeigt, wie die gesammelten Daten im Browser dargestellt werden. Zu sehen ist die Hauptübersicht von NCover, mit den bisherigen Daten von zwei Testdurchläufen. Diese sind als Balken nebeneinander dargestellt, um für die einzelnen Kategorien direkt Änderungen zum Positiven oder zum Negativen auszumachen.

Ein Klick auf den Eintrag zeigt die einzelnen Durchläufe (siehe Abbildung 8). Diese Einträge sind wieder Anklickbar, so dass eine Navigation bis hinunter auf Methoden-Ebene möglich ist, wie Abbildung 9 zeigt. Wird zu einer Methode navigiert, wird der eingefärbte Code sichtbar, den wir vorher schon einmal bei Visual Studio zu Gesicht bekommen haben (vgl. Abbildung 6). Ein wichtiges Feature, da somit auch Außenstehende Einblick in den Code der Testabdeckung haben, ohne direkten Zugriff auf das Projekt zu haben.

TESTABDECKUNG BEI UNIT TESTS

Bisher wurden die Daten durch manuelle Tests erfasst. NCover sammelt die Daten und stellt sie im Browser sowie der Visual Studio Erweiterung bereit. Diese Erweiterung ist es auch, die das Einfärben der Codezeilen übernimmt. Aber vollautomatische Tests waren dabei noch nicht im Spiel. Immer wurde das Beispielprojekt manuell gestartet.

Dieses Vorgehen kann sinnvoll sein, wenn bei einer Anwendung, zum Beispiel mit einer grafischen Oberfläche, die Codeabdeckung bei manueller Bedienung aufgezeichnet und ausgewertet werden soll. Der Regelfall ist allerdings, dass die automatische Ausführung von Unit Tests dafür sorgt, dass NCover im Hintergrund fleißig Daten sammelt, die

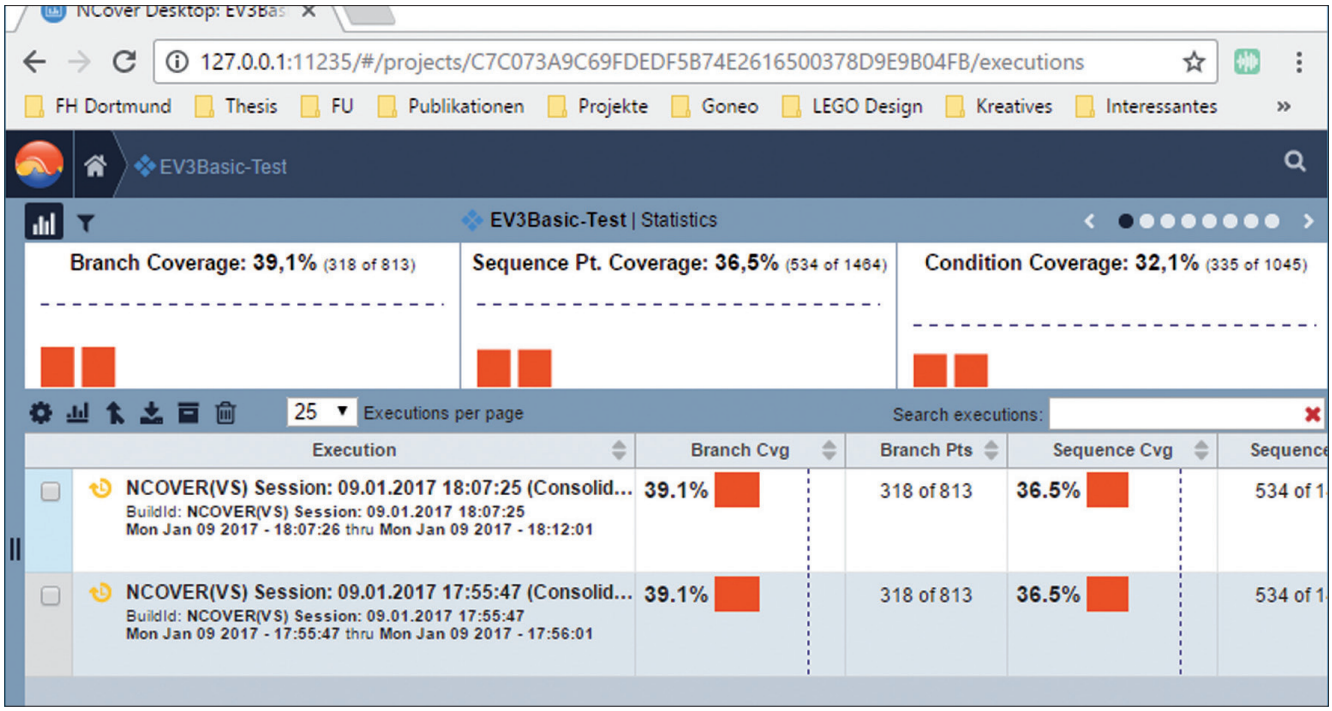


Abbildung 8: Die konkreten Durchläufe einer durch NCover analysierten Anwendung.

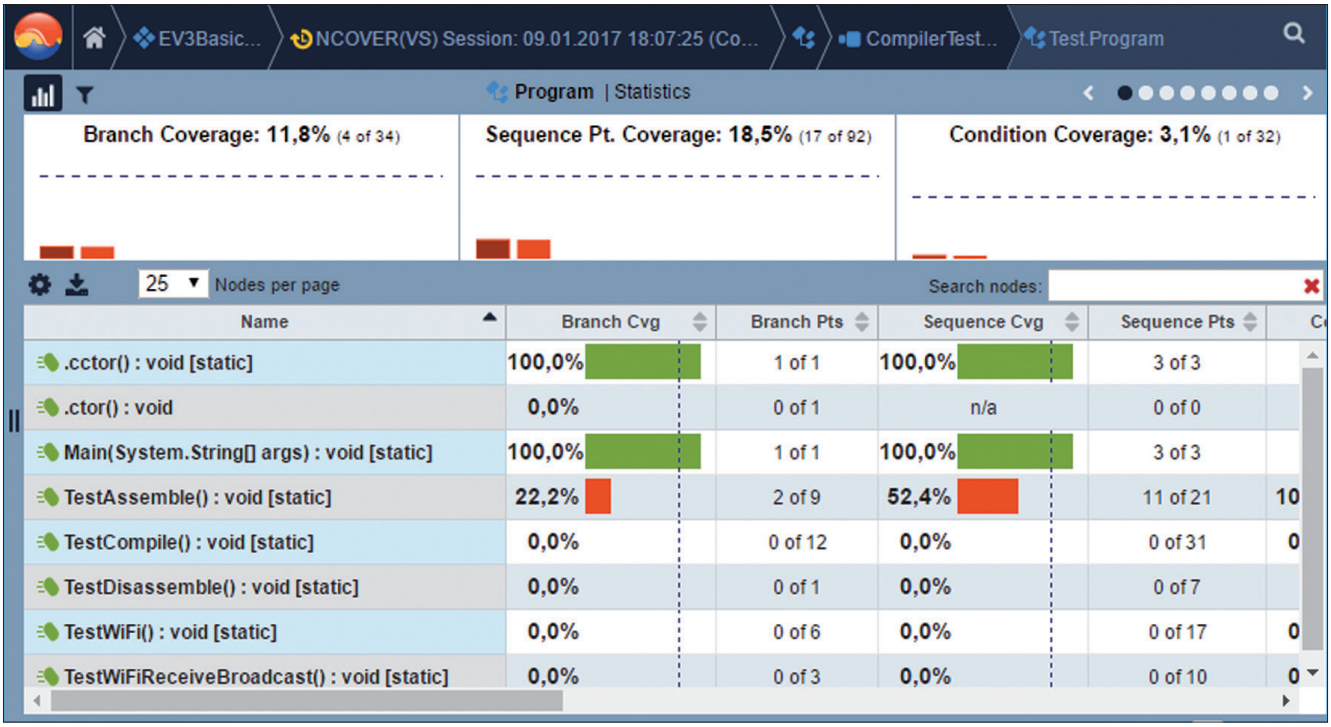


Abbildung 9: Detailliertere Einträge eines spezifischen Durchlaufs.

im Anschluss ausgewertet werden können.

Die Visual Studio Erweiterung von NCover übernimmt diese Aufgabe. Obwohl in den Optionen der Erweiterung noch die Möglichkeit vorhanden ist, den hauseigenen Test Runner zu aktivieren, ist das nicht mehr erforderlich. NCover arbeitet gut mit bekannten Test Runnern wie NUnit und MSTest zusammen. Die Interaktion läuft direkt über die Markierungen neben einem Unit Test ab (siehe Abbildung 10). Der Eintrag *Cover All* führt alle Unit Tests der Test-Klasse aus und sorgt direkt für das Datensammeln mittels NCover. Alternativ kann eine einzelne Testmethode separat ausgeführt und überwacht werden.

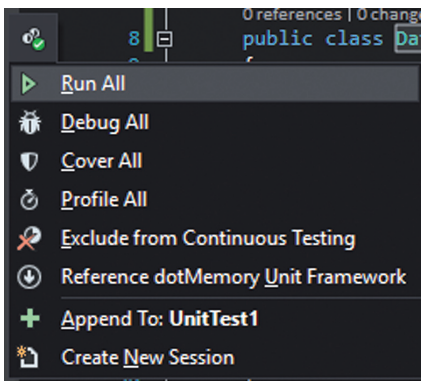


Abbildung 10: Die Integration von NCover in die seitlichen Markierungen eines Unit Tests.

***Listing 1:* Ein Unit Test zur Messung der Testabdeckung der Klasse DataArea.**

```
[TestMethod]
public void TestDataAreaPadding()
{
    var dataArea = new DataArea();
    dataArea.Add("test var1", 1, 1, DataType.I8, false);
    dataArea.Add("test var2", 4, 1, DataType.F, false);
    dataArea.Add("test var3", 2, 1, DataType.I16, false);

    Assert.AreEqual(dataArea.TotalBytes(), 10);
}
```

Nach der Datenerfassung zeigt NCover die gesammelten Informationen in der aktuellen Test-Session an. Dort integriert sich ein neuer Bereich Namens *Coverage Tree*. Abbildung 11 zeigt das am Beispiel eines sehr einfachen Unit Tests. Der Code für diesen Test ist in Listing 1 hinterlegt. Der Test erzeugt eine DataArea, die verschiedene Variablen mit Informationen wie deren Größe, Typ und Position erfasst. Hintergrund ist, dass diese DataArea als Binär-Stream persistiert wird und dabei ein Padding zu beachten ist. Je nachdem, welche Variablen von welchem Datentyp in welcher Reihenfolge hinzugefügt werden. Abgeprüft wird das anhand der Methode `TotalBytes`.

Wie die Testabdeckung aus Abbildung 11 zeigt, wird der Tests selbst vollständig überdeckt (grüner Bereich). Die Klasse DataArea befindet sich in der Assembly LMSAssembler im gleichnamigen Namensraum. Auch dort wird die Testüberdeckung angezeigt, die für diesen Testfall zwei Prozent erreicht. Ein niedriger Wert, der allerdings dadurch zustande kommt, dass nur ein Testfall mit einer Klasse und nur zwei unterschiedlichen Methoden ausgeführt wird.

Und dennoch sind mehrere Werte interessant. Wird der Namensraum LMSAssembler im Coverage Tree aufgeklappt, wird ersichtlich, dass die Klasse DataArea nur zu 51% und die enthaltene Klasse Dataelement zu 67% überdeckt wird. Für einen einzigen Test vielleicht keine schlechten Werte, trotzdem wären in einem realen Szenario noch Testfälle zu ergänzen, um wichtigere Bereiche mit Tests abzudecken.

Die gesammelten Daten der Unit Tests sind ebenfalls über die NCover-Zentrale im Browser einsehbar. Inklusive einzelne Überdeckungen der Methoden, sowie die einzelnen Codezeilen inklusive Einfärbung.

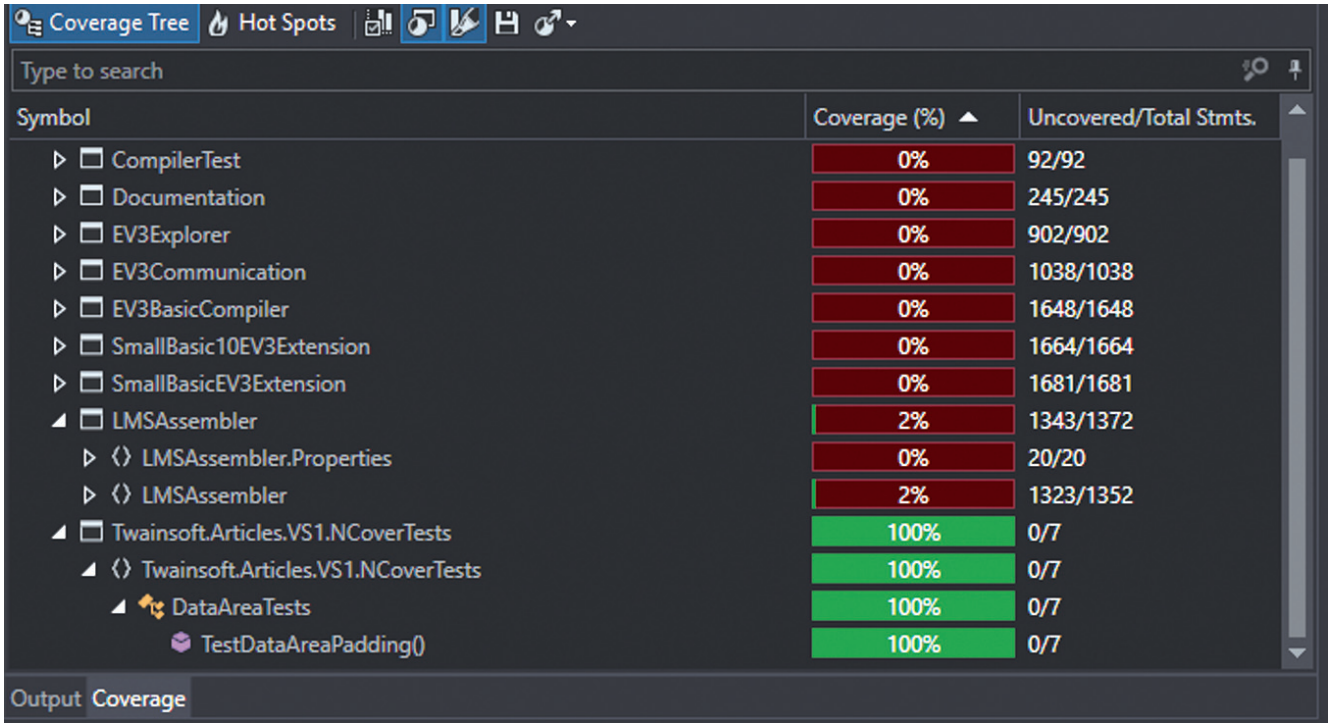


Abbildung 11: Die spezifischen Überdeckungen durch einen einzigen Unit Test.

FAZIT

NCover bietet zahlreiche Möglichkeiten, den Testabdeckung einer Anwendung zu erfassen und zu analysieren. Dabei spielt es keine Rolle, ob der Code durch manuelle Interaktionen mit der Anwendung ausgeführt wird oder ob automatisch ausgeführte Unit Tests im Spiel sind. Der Mechanismus, um die Daten zu sammeln ist einfach zu nutzen. Etwas Einarbeitungszeit benötigen dagegen die verschiedenen Darstellungen der gefundenen Daten, um diese anschließend sinnvoll interpretieren zu können. Wer sich vor dem Kauf noch etwas über die Features und die allgemeine Funktionsweise von NCover informieren möchte, findet auf der NCover Webseite eine ausführliche Dokumentation [7] zu allen Produktversionen.

Letztlich bietet NCover genau so viel Einfluss, wie wir als Entwickler das zulassen. Denn das Messen der Testabdeckung ist nur der erste Schritt hin zu einer Verbesserung. Wer diese Daten nicht nutzt, um anschließend konsequent seine Unit Tests zu überdenken und zu überarbeiten, hat am Ende nichts gewonnen. Diese Überarbeitungen kosten allerdings wiederum Zeit, die eingeplant sein muss. Es bringt nichts, wenn Tools fleißig Metriken sammeln, deren Daten dann irgendwo vor sich hingammeln.

LINKS UND QUELLEN

[1]: Die Quizshow 1, 2 oder 3 auf Wikipedia: https://de.wikipedia.org/wiki/1,_2_oder_3

[2]: Der Begriff Testabdeckung auf Wikipedia: https://de.wikipedia.org/wiki/Testabdeckung

[3]: Die Webseite von NCover: https://www.ncover.com/

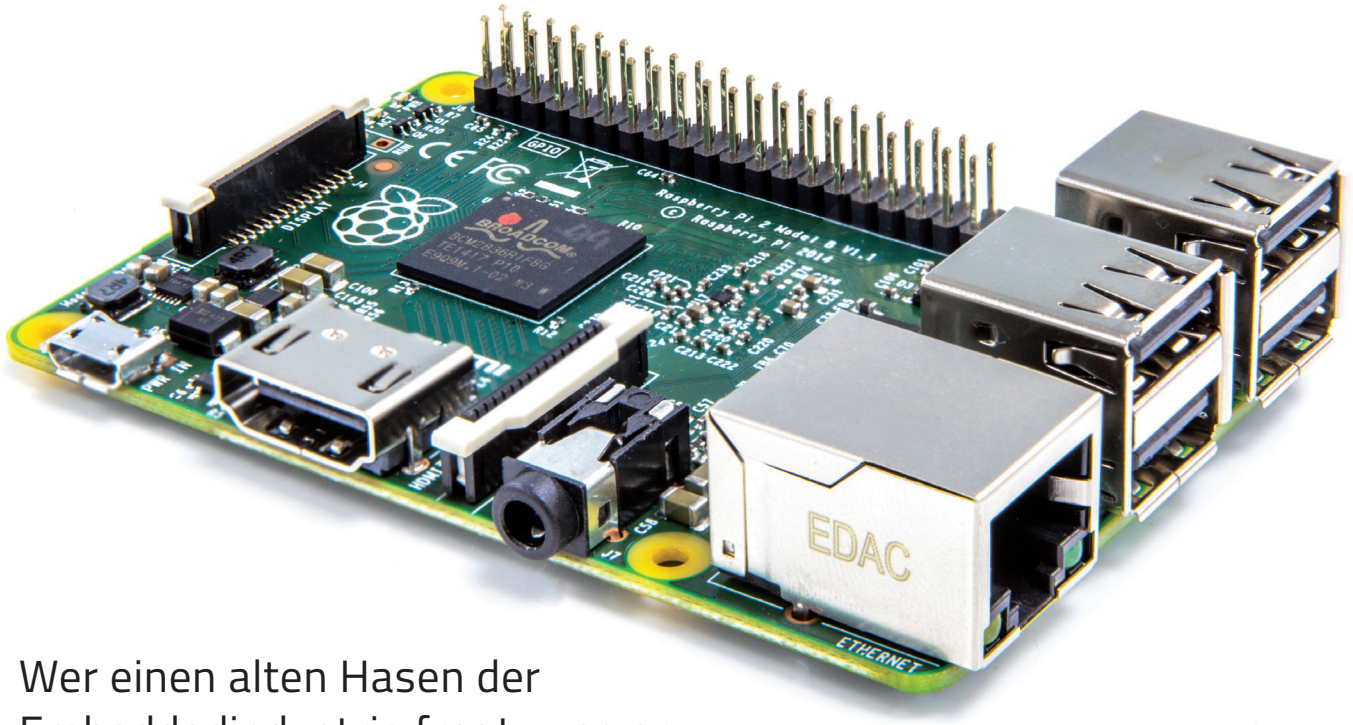
[4]: Die Webseite von JetBrains dotCover: https://www.jetbrains.com/dotcover/

[5]: Die Webseite von PartCover: https://github.com/sawilde/partcover.net4

[6]: Die Webseite von OpenCover: https://github.com/OpenCover/opencover

[7]: Die Online-Dokumentation von NCover: http://www.ncover.com/support/docs/index

SCHAU DER WELLENFORM AUFS MAUL!



Wer einen alten Hasen der Embeddedindustrie fragt, was er von Microsoft als Prozessrechnerbetriebssystem hält, bekommt eine eindeutige und kurze Antwort: Mist!

Autor: ANNETTE HEIDI BOSBACH

Nachdem der Gadgeteer nicht sonderlich erfolgreich war – mit den Ursachen des Niedergangs dieses Produkts möchten wir uns hier nicht näher befassen – hat Microsoft nun mit Eben Upton's Mannen einen neuen Partner gefunden. Windows 10 steht in einer stark abgespeckten Version für den Upton'schen Prozessrechner der Bauart RaspberryPi2 und RaspberryPi3 zur Verfügung: Der im Rasberry Pi1 verbaute Prozessor der Bauart BCM2835 hat die notwendigen Instruktionssätze nicht und kann Windows 10 deswegen nicht ausführen. Anzumerken ist, dass das Windows 10 am Prozessrechner eine reine Hostplattform für MSR- und sonstige Applikationen darstellt: Klassische Features wie den Desktop oder die von Tablet und Phone bekannte Kacheloberfläche suchen Sie vergebens.

VORBEREITUNGSHANDLUNGEN

Wir wollen uns in diesem Artikel auf die Nutzung von Windows 10 als MSR-Plattform konzentrieren. Dazu befolgen Sie bitte die unter *https://developer.microsoft.com/en-us/windows/iot/GetStarted* bereitstehenden Anweisungen. Alternativ können Sie Ihren RaspberryPi auch über NOOBS mit einem Windows10-Image ausstatten – lassen Sie uns in den folgenden Schritten die Insider Preview nutzen, die als Image mit dem Dateinamen *Windows10_InsiderPreview_IoTCore_RPi_ARM32_en-us_14986.iso* ausgeliefert wurde. Die Nutzung älterer Versionen ist insofern unsinnig, als Microsoft in den letzten Wochen massiven Aufwand in die Verbesserung der IO-Performance gesteckt hat. Klicken Sie die .iso-Datei im ersten Schritt doppelt an, um ihren Inhalt am Desktop zu öffnen. Microsoft belastet seine Entwickler mit einem separaten Installationsprogramm (s. Abbildung 1) – nach seiner Abarbeitung steht das Image unter *C:\Program Files (x86)\Microsoft IoT\FFU\RaspberryPi2\flash.ffu* zum Einsatz bereit.

Zwecks einfacherer Bedienung unserer Programme empfiehlt sich das Anschließen von Maus und Tastatur. Wegen der sehr geringen Reichweite des WLAN-Moduls ist die Verwendung von Ethernet ratsam – das Deployment der Binaries dauert auch so lang genug. Wundern Sie sich nicht, wenn der erste Start die eine oder andere Minute in Anspruch nimmt: Der Prozessrechner muss unter anderem seine SD-Karte aufräumen und diverse Partitionen erweitern. Bei 32 GB–Karten ist eine Wartezeit von 15

Minuten völlig normal – so lange sich der Kreis dreht, ist alles in Ordnung. Auf Seiten der Workstation ist Visual Studio 2015 mit Service Pack 3 erforderlich, da am Raspberry Pi wegen des Fehlens von GUI und Co nicht entwickelt werden kann. Laden Sie zudem das unter *https://marketplace.visualstudio.com/items?itemName=MicrosoftIoT.WindowsIoTCoreProjectTemplates* bereitstehende Erweiterungspaket herunter, das Visual Studio um einige Hilfswerkzeuge für die Arbeit mit Devices erweitert. Falls Sie das Paket nicht von Hand deployen möchten, können Sie in VS auch auf Tools –> Extensions and Updates –> Online klicken und das Modul *Windows IoT Core Project Templates* von Hand suchen.

ES GEHT AUCH OHNE!

Die offizielle Dokumentation von Microsoft beschreibt ein Hilfsprodukt, das allerdings nur unter Windows10 funktioniert. Da wir den Upgradeaufwand bisher gescheut haben, sei angemerkt, dass Sie auch ohne das Programm auskommen: Zum Deployment des Images gibt es andere Werkzeuge. Das manuelle Einrichten einer Debuggingbeziehung können Sie folgender Textbeschreibung entnehmen:

IoT-Busse wie I2C sind auch außerhalb des Internet-Of-Things relevant: Ein Gutteil der zum Zugriff auf Hardware notwendigen Klassen ist witzigerweise schon Teil von Visual Studio; die VSIX-Datei installiert primär eine zusätzliche Vorlage zur Realisierung von Hintergrundapplikationen. Erstellen Sie für unsere ersten Übungen eine neue Applikation auf Basis der Vorlage Windows –> Universal –> Blank App. In den folgenden Schritten hört der Code auf den Dateinamen PPEDVGPIO. Als erstes echtes Hardwareprodukt wollen wir eine Gruppe von LEDs aufleuchten lassen. Verbinden Sie entweder drei verschiedenfarbige Dioden oder eine RGB-LED wie in Abbildung 2 gezeigt mit dem Prozessrechner.

Vor dem Beginn der Programmierung muss das Projektskelett um einen Verweis auf Universal Windows –> Extensions –> Windows IoT Extensions for the UWP-SDK ergänzt werden. Fehlt dieser Verweis, so kann IntelliSense die betreffenden Klassen zur Laufzeit nicht erreichen. Als nächsten Akt legen Sie bitte zwei globale Membervariable an: Die GPIO-Controller-Instanz liefert einen Verweis auf die Controllerklasse, die für die Interaktion mit den GPIO-Pins des Prozessrechners als Ganzes verantwortlich ist. Der einzelne Pin wird über eine Instanz der Klasse GpioPin abgebildet:

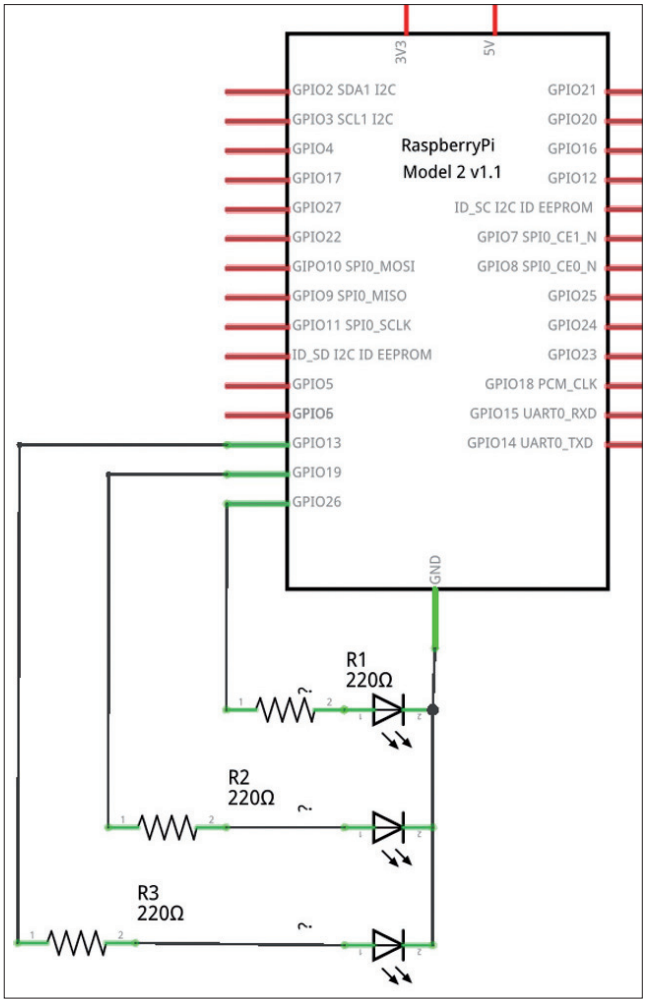


Abbildung 2: Das alte Spiel mit den Vorwiderständen sollte bekannt sein

```
public sealed partial class MainPage : Page
    {
    GpioController myController;
    GpioPin myPin;
```

Im Konstruktor der Main-Page findet sich die eigentliche Initialisierung. Nach dem Beschaffen einer Instanz der Controllerklasse, weißt man diese unter Nutzung der Funktion OpenPin dazu an, eine Pin-Instanz bereitzustellen:

```
    public MainPage()
    {
        this.InitializeComponent();
        myController = GpioController.GetDefault();
        myPin = myController.OpenPin(26);
```

Interessant ist hier die Zuweisung zwischen Zahl und Pin: Der im Code verwendete Pin Nummer 26 wird am Header

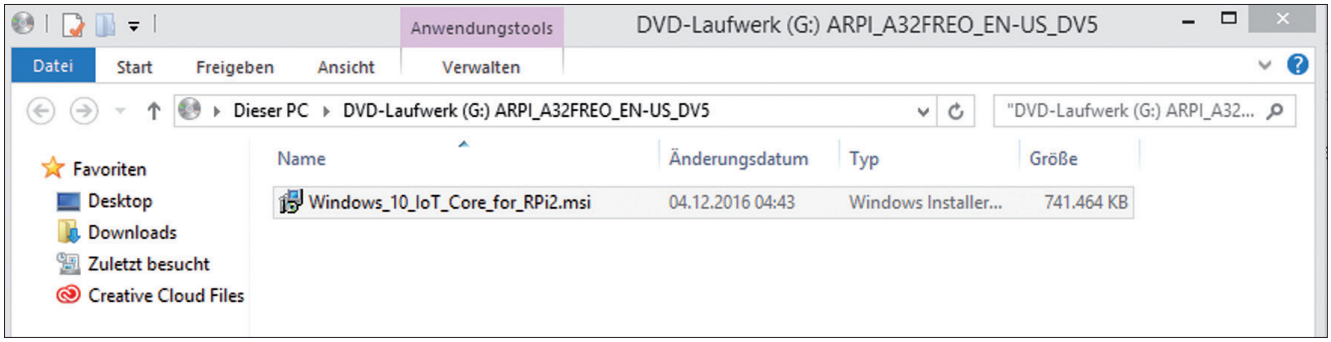


Abbildung 1: Verstehe, wer will...

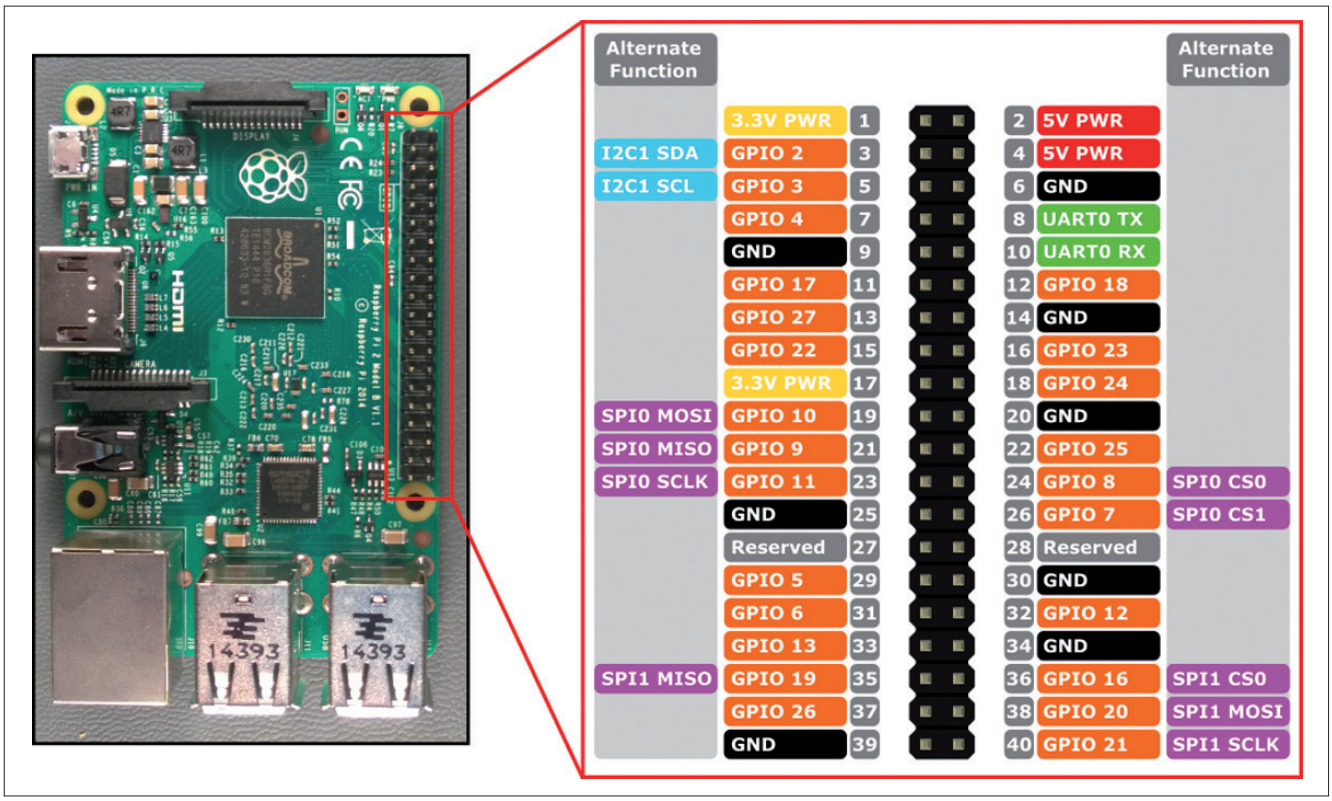


Abbildung 3: Diese Tabelle beseitigt Unklarheiten

Pin Nummer 37 entsprechen. Microsoft liefert die in Abbildung 3 gezeigte Zuweisungstabelle, die das babylonische Chaos ein wenig aufräumt.

GPIO-Pins sind nach dem Start normalerweise inert, verhalten sich also als hochohmige Eingänge. Im ersten Schritt setzen wir den Drive-Mode auf Output, um im zweiten Schritt den gewünschten Wert auszugeben.

Dieses auf den ersten Blick seltsame Verhalten ist insofern wichtig, weil hochohmige Pins mit der Schaltung normalerweise nicht interagieren - ein auf Low oder High gesetzter Ausgangspin könnte während des Starts Strom liefern oder senken, was zu Beschädigungen der angeschlossenen Hardware führen könnte.

```
        public MainPage(){
        . . .
            myPin.
SetDriveMode(GpioPinDriveMode.
Output);
            myPin.
Write(GpioPinValue.High);
        }
```

Für die eigentliche Programmausführung wechseln Sie in die Konfiguration Remote Devices. Visual Studio wird den Prozessrechner in vielen Fällen automatisch erkennen und durch Anklicken von Select zum Debugging freigeben. Schicken Sie das Programm sodann wie gewohnt in Richtung des Prozessrechners - bei korrekter Verkabelung wird eine der drei Leuchtdioden aufleuchten, wie in Abbildung 4 gezeigt.

NICHT ÜBERALL!

Achten Sie darauf, dass manche Prozessrechner – Shenzhen Xunlong ist hier ein besonderer Verdächtiger – nicht voll inert starten.

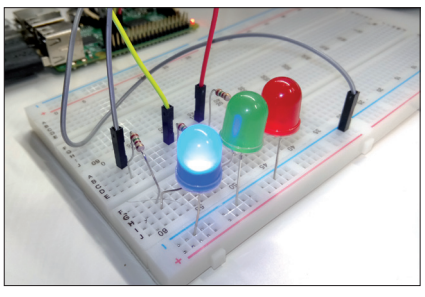


Abbildung 4: Die LED zeigt: Alles in Ordnung!

EINE FRAGE DER HELLIGKEIT

Windowsbasierte Prozessrechner kompensieren ihr eher schwaches Echtzeitverhalten durch die immensen Möglichkeiten zur Realisierung ansprechender Benutzerinterfaces – wer einmal mit STM CUBE oder einer ähnlichen API in einem Framebuffer herumgefuhrwerkt hat, ist von den Vorteilen eines GUI-Stacks überzeugt.

Nun werden wir unser Programm zur Ausgabe von beliebigen Farbwerten befähigen. Eine naive Implementierung würde auf einen DAC zurückgreifen, um die LED mit beliebigen Spannungswerten versorgen zu können. Bei für “menschlichen Konsum” vorgesehenem Licht ist dies allerdings nicht notwendig, da das Auge das eingehende Licht über eine kurze Zeitspanne integriert. Dies ermöglicht die Nutzung der Pulsbreitenmodulation, deren Grundprinzip in der Abbildung 5 kurz angerissen ist.

Realisieren Sie als erstes ein Formular, dass drei Slider aufweist. Im Interesse des einfacheren Handlings vergeben wir Werte von 0-20. Es steht ihnen allerdings frei, andere Minimal- und Maximalwerte zu nutzen, die an ihre Situation besser angepasst sind. Eben Upton’s Entscheidung, eine BroadCom-CPU zu verwenden, rächt sich an dieser Stelle: Während durchschnittliche Mikrocontroller mindestens vier oder fünf PWM-Kanäle mitbringt, hat ein Raspberry PI derer nur zwei. Aus diesem Grund - und auch aufgrund der unklaren Unterstützungslage - wollen wir PWM stattdessen von Hand realisieren. Als ersten Schritt in diese Richtung realisieren wir eine Endlosschleife, in der der im vorigen Schritt verwendete

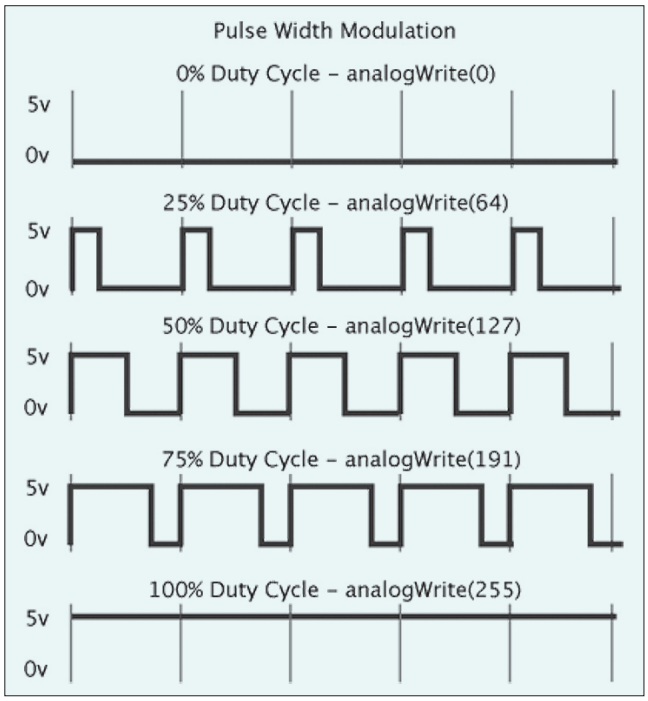


Abbildung 5: Die "Wellenpakete" verwandeln sich bei integrierenden Verbrauchern in konstante Spannungen *(Bildquelle: Arduino.cc)*

Pin permanent ein und ausgeschaltet wird. Ob das Fehlen einer dedizierten Threadklasse - hier arbeiten wir besser nicht mit nativen Code - setzen wir stattdessen auf das Async-Framework:

```
public MainPage()
{
    . . .

    var t = Task.Run(() => doPWMCycling());
}

void doPWMCycling()
{
    while (1 == 1)
    {
        myPin.Write(GpioPinValue.High);
        myPin.Write(GpioPinValue.Low);
    }
}
```

An dieser Stelle lohnt sich das Vorhandensein eines leistungsfähigen Digitalspeicheroszillographen. Abbildung 6 zeigt, was am Bildschirm eines LeCroy 9354AM erscheint. Besonders stabil ist diese Wellenform nicht, reicht ob der vergleichsweise hohen Geschwindigkeit allerdings für grundlegendes PWM mit gelegentlichem Flimmern aus.

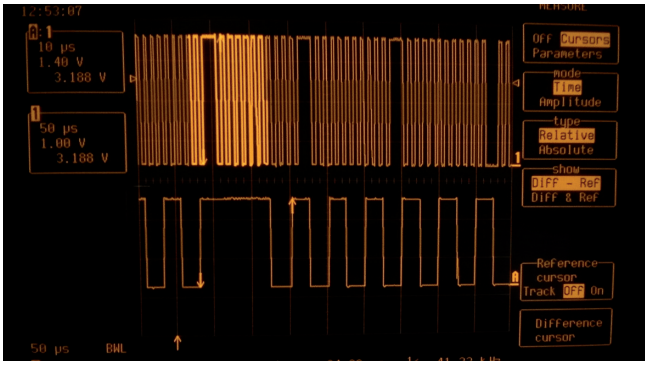


Abbildung 6: Das GPIO-Subsystem ist nicht sonderlich schnell, und wird vom Garbage-Collector unterbrochen

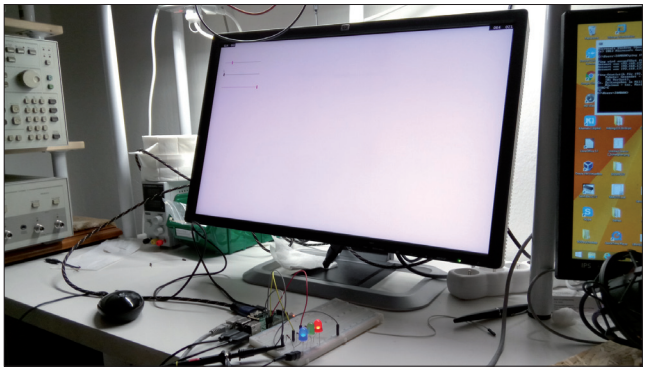


Abbildung 7: PWM in Action

Im nächsten Schritt realisieren wir das eigentliche Software-PWM. Die dahinterstehende Logik ist einfach. Sie lassen innerhalb der Endlosschleife einen Index von 0-20 laufen, und schalten die betreffenden Dioden in Abhängigkeit ihres jeweiligen Grenzwerts ein und aus:

```
void doPWMCycling()
{
   while (1 == 1)
   {
       for (int i = 0; i < 20; i++)
       {
           if (i >= myOffR) myPinR.Write(GpioPinValue.
           Low);
           else myPinR.Write(GpioPinValue.High);
           if (i >= myOffG) myPinG.Write(GpioPinValue.
           Low);
           else myPinG.Write(GpioPinValue.High);
           if (i >= myOffB) myPinB.Write(GpioPinValue.
           Low);
           else myPinB.Write(GpioPinValue.High);
       }
    }
}
```

Jagen Sie das Programm sodann abermals auf den Prozessrechner – Abbildung 7 zeigt die unterschiedlichen Helligkeiten der einzelnen LEDs.

Anzumerken ist, dass es bei längerer Betrachtung der Leuchtdioden immer wieder zu kleinen Rucklern kommt: Das liegt daran, dass Raspberry PI mitunter mit anderen

SELEKTIERTE LEDS!

Die Helligkeit von LEDs ist – auch innerhalb einer Serie – kein besonders konstanter Wert. Wer selbst Dioden zu einem RGB-Cluster zusammenstellen möchte, kommt meist nur mit manueller Selektion ans Ziel.

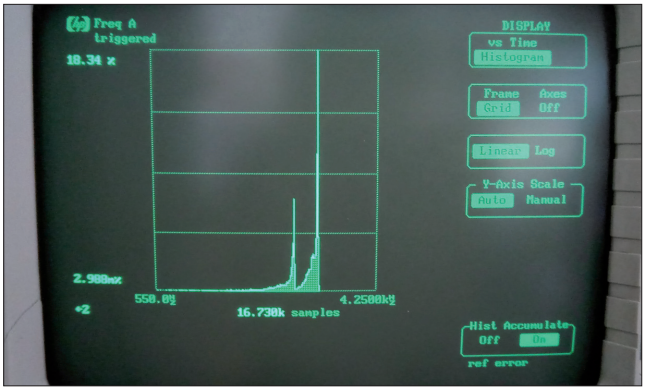


Abbildung 8: Sonderlich stabil sieht diese Wellenform nicht aus

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18

Abbildung 9: Hier wird scharf geschossen!

Aufgaben beschäftigt ist. Zur Quantifizierung dieses Verhaltens empfiehlt sich die Nutzung eines Modulationsdomänenanalysators - wer einen 53310A anschließt, erhält das in Abbildung 8 gezeigte Schirmbild.

BITTE ETWAS SCHNELLER

Aufmerksame Leser fragen sich bei der Betrachtung der obigen Schleife, warum wir Write mehrfach hintereinander aufrufen und bei jedem Durchlauf entweder High oder Low schalten. Das liegt daran, dass der von Haus aus verbaute GPIO-Treiber sehr langsam ist: Das Ein- und Ausschalten eine Diode nimmt eine im Vergleich zum restlichen Schleifendurchlauf immens größere Zeit in Anspruch. Der Zeitunterschied ist so hoch, dass eine nicht regelmäßige Abarbeitung des Ein- und Ausschaltcodes zu einer nicht funktionierenden PWM-Implementierung führen würde: Die Dioden würden permanent gleich hell leuchten, weil der durch die Schleifenimplementierung entstehende Unterschied in der Einschaltzeit irrelevant wird.

Ein Weg zur Umgehung dieses Problems ist die Nutzung eines Memory-Map-Treibers. Zum Verständnis dieser Technik will dieser Artikel ein wenig in die Welt primitiver Mikrocontroller entführen - als Beispiel erwähnen wir den berühmtberüchtigten PIC16F84A. Jeder der als Port bezeichne-ten Ausgabepin-Gruppen hat im Arbeitsspeicher ein Register, das den aktuellen Zustand der Pins beschreibt. Will man den Zustand eines Pins ändern, so schreibt man einfach einen Wert in dieses Register - aus Softwaresicht ist es so, als ob man eine Variable beeinflussen würde (siehe Abbildung 9).

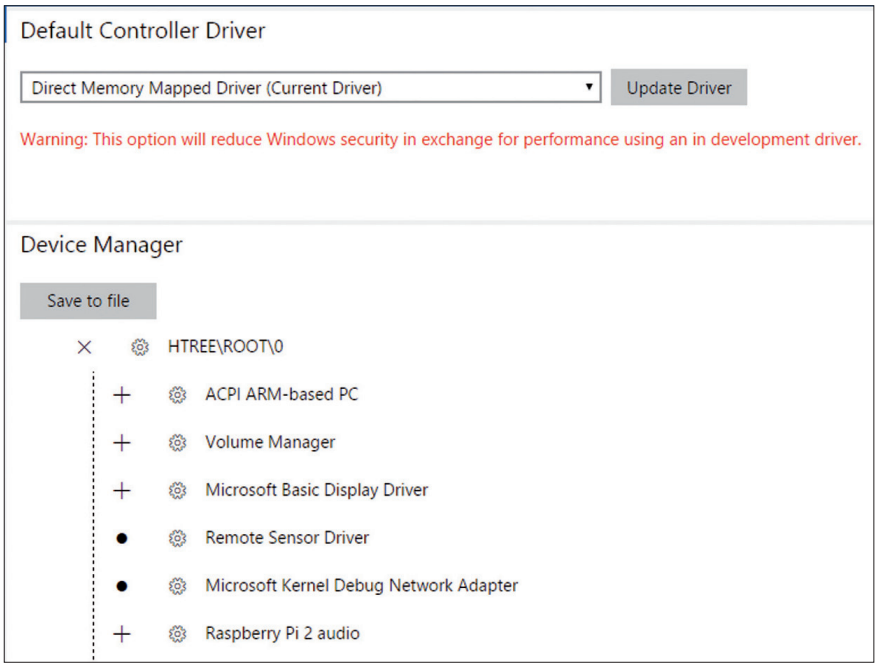


Abbildung 10: Dieser Treiber sorgt für optimale Leistung

Auf Basis von Windows 10 für Devices arbeitende Geräte, stellen eine per Webbrowser ansprechbare Steueroberfläche bereit: Auf meiner Workstation lautet die zu öffnende URL *http://192.168.137.189:8080/*. Nutzen Sie als Benutzername Administrator, das von Microsoft vor-gegebene Passwort lautet p@ssword – die im Rahmen des ersten Logins erscheinende Aufforderung zur Passwortänderung kann getrost ignoriert werden. Das Backend bietet einige Dutzend Optionen an - darunter auch eine Art Taskmanager, mit dem Sie Applikationen löschen und die CPU-Auslastung kontrollieren können. Für unsere Bedürfnisse ist an dieser Stelle allerdings nur das Devices-Tab relevant, indem Sie den Lightning-Treiber auswählen (s. Abbildung 10).

Achten Sie darauf, dass die Änderung des Treibers einen Neustart des Prozessrechners voraussetzt. Das Backend bietet dies in aktuellen Versionen von selbst an - im schlimmsten Fall müssen Sie Visual Studio neu starten, weil der Reboot den Debugger aus dem Takt bringt. Manchmal ist auch ein Aus- und Einschalten der Stromversorgung des Prozessrechners erforderlich.

Zur Arbeit mit Lightning ist eine komplett andere Programmstruktur erforderlich: Wer ein für den normalen GPIO-Treiber vorgesehenes

SICHERHEITSRISIKO!

Das Aktivieren vom direkten Speicherzugriff ermöglicht bösartigen Applikationen das Kapern des Prozessrechners. Berücksichtigen Sie dies insbesondere dann, wenn ein System regulatorischen Anforderungen unterlegen ist oder einen Internetzugang aufweist.

Programm ausführt, erntet zur Laufzeit eine NullPointerException. Öffnen Sie im ersten Schritt den NuGet-Paketmanager, und installieren Sie das Lightning-Paket. Achten Sie dabei darauf, die in der Abbildung 11 gezeigte Checkbox zu markieren.

Als nächstes müssen Sie das Manifest um die Deklaration des IOT-Namespaces ergänzen:

```
<Package
  xmlns="http://schemas.microsoft.com/appx/manifest/
  foundation/windows10"
  . . .
  xmlns:iot="http://schemas.microsoft.com/appx/manifest/
  iot/windows10"
  IgnorableNamespaces="uap mp iot">
```

In der Capabilities-Deklaration sind gleich zwei neue Berichtigungen erforderlich. Erstens der allgemeine Zugriff auf Hardwaregeräte, und zweitens der spezifische Zugriff auf das Lightning-Gerät. Es wird zum Zeitpunkt der Drucklegung dieses Artikels über seine GUID beschrieben:

```
  <Capabilities>
    <Capability Name="internetClient" />
   <iot:Capability Name="lowLevelDevices" />
   <DeviceCapability
   Name="109b86ad-f53d-4b76-aa5f-821e2ddf2141"/>
  </Capabilities>
```

Zur Aktivierung von Lightning selbst ist dann folgendes Snippet erforderlich, das 1:1 aus der von Microsoft vorgegebenen Dokumentation übernommen werden kann.

```
public MainPage()
{
    this.InitializeComponent();
    if (LightningProvider.IsLightningEnabled)
    {
        LowLevelDevicesController.DefaultProvider =
        LightningProvi-der.GetAggregateProvider();
     }

     myController = GpioController.GetDefault();
     . . .
```

Ein interessanter Randfall ist die Nutzung von Lightning für einzelne Protokolle. In diesem Fall können Sie auf die in der Tabelle gezeigten Syntaxvarianten zurückgreifen, die nur eine bestimmte Busart aktivieren:

Lightning für	*Code*

GPIO	if (LightningProvider.IsLightningEnabled)
	{
	GpioController gpioController =
	(await GpioController.GetControllers
	Async(LightningGpioProvider.
	GetGpioProvider()))[0];
	GpioPin pin = gpioController.OpenPin(LED
	PIN, GpioSharingMode.Exclusive);
	}

I2C	if (LightningProvider.IsLightningEnabled)
	{
	I2cController controller =
	(await I2cController.GetControllersAsync
	(LightningI2cProvider.GetI2cProvider()))[0];
	I2cDevice sensor = controller.GetDevice
	(new I2cConnectionSettings(0x40));
	}

SPI	if (LightningProvider.IsLightningEnabled)
	{
	SpiController controller = (await
	SpiController.GetControllersAsync(Lightning
	SpiProvider.GetSpiProvider()))[0];
	SpiDevice SpiDisplay = controller.GetDevice
	(spiConnectionSettings);
	}

Schicken Sie das Programm nach getaner Arbeit abermals in Richtung des Raspberry PI. Auffällig ist, dass die Frequenz der Wellenform nun wesentlich höher ist - dies lässt sich unter anderem im Diagramm in Abbildung 12 überprüfen.

Berücksichtigen Sie, dass das periodisch auftretende Flimmern durch Lightning nicht wesentlich verbessert wird. Das liegt daran, dass es sich hierbei um ein Problem des Schedulings handelt: Wenn das Windows-Betriebssystem einen anderen Thread als wichtiger erachtet,

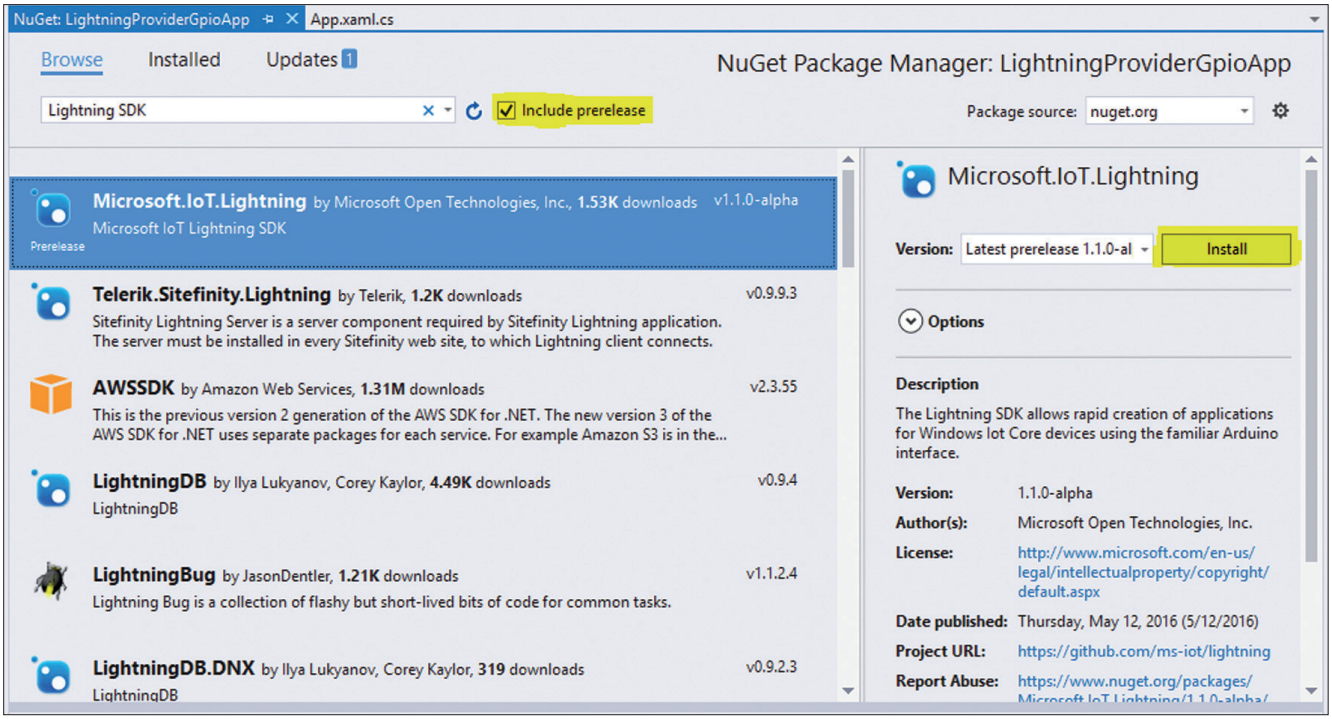


Abbildung 11: Dieses SDK ist noch nicht final *(Bildquelle: Microsoft)*

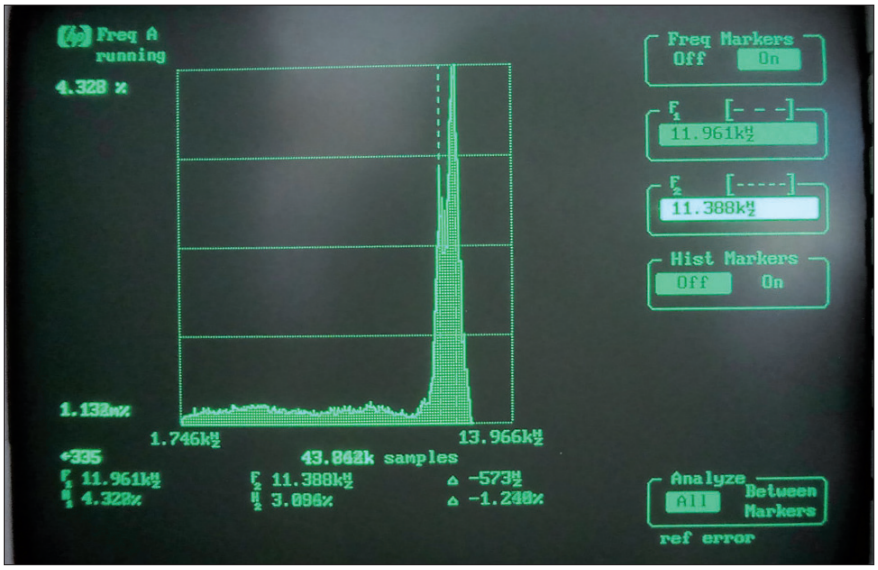


Abbildung 12: Dank Lightning arbeiten die GPIO-Pins wesentlich schneller – stabiler werden Sie allerdings nicht.

muss unsere MSR-Aufgabe pausieren. Dies wirkt sich in Form von Jitter aus – zu seiner Umgehung gibt es diverse Methoden (Stichwort C++), auf die wir in diesem elektronikbezogenen Text allerdings nicht weiter eingehen möchten.

MIT ATTENUATOR!

Prozessrechner Upton'scher Bauart arbeiten mit einer Signalspannung von 3.3V. In der Praxis kommt es immer wieder vor, dass man Kontakt mit anderen Spannungspegeln aufnehmen muss – ein Klassiker wären von einem Fahrzeug emittierte Signale, die entweder 12 oder 24 Volt aufweisen. Klassische resistive Spannungsteiler kommen an dieser Stelle bald an ihre Grenzen: Als ersten Test wollen wir uns das Schaltverhalten des RPi ansehen. Bei der Arbeit mit Prozessrechnersystemen ist es empfehlenswert, sowohl die Software als auch die Hardware holistisch zu betrachten und an die zu diskutierende Aufgabenstellung anzupassen. Als erstes Beispiel realisieren wir einen "Spiegel", der die am GPIO-Pin anliegende Spannung in einen weiteren Pin übernimmt. Legen Sie hierzu ein weiteres Programm unter Nutzung der Lightning-APIs an – im Code zum Heft heißt es GPIOMirror:

```
public MainPage()
{
    . . .
    myController = GpioController.
    GetDefault();
        myPinO = myController.
        OpenPin(13);
        myPinO.SetDriveMode
                (GpioPinDriveMode.Output);
        myPinO.Write(GpioPinValue.
        High);
        myPinI = myController.
        OpenPin(19);
        myPinI.SetDriveMode(Gpio
        PinDriveMode.Input);
```

Neben der hier aus Platzgründen nicht abgedruckten Initialisierung des Controllers muss man dieses Mal einen Eingang und einen Ausgang anlegen: Der Unterschied zwischen den beiden Pinfunktionen wird über die an die Methode SetDriveMode übergebenen Parameter bestimmt. Damit sind wir zum Anstoßen des eigentlichen Ausgabetasks bereit. Er liest die am Eingangspin anliegende Spannung ab, und schreibt sie sodann in Richtung des Ausgangspins weiter. Im Interesse der "Balanz" nutzen wir hier absichtlich kein ElseIf-Kommando – nur so ist sichergestellt, dass die Abarbeitungszeit von Low- und High-Durchläufen der Schleife weitgehend identisch bleibt:

```
        var t = Task.Run(() =>
        doPWMCycling());
    }
    void doPWMCycling()
    {
        while (1 == 1)
        {
            if (myPinI.
            Read()==GpioPinValue.
            High) myPinO.
            Write(GpioPinValue.
            High);
            if (myPinI.Read() ==
            GpioPinValue.Low) myPi-
            nO.Write(GpioPinValue.
            Low);
        }
    }
```

Als nächste Aufgabe müssen wir den Generator anregen. Wir nutzen dazu einen Funktionsgenerator, Bauart Tektronix AWG2021, der das in Abbildung 13 gezeigte langsam ansteigende Signal ausgibt.

Zur Entgegennahme der entstehenden Wellenformen greifen wir sodann auf einen Digitalspeicheroszillographen zurück, der auf den Nulldurchlauf des Funktionsgenerators getriggert (negative Slope auf halber Spannung) wird. Auf diese Art und Weise kann man feststellen, wo die Spannung von null auf eins umgeschaltet wird (siehe Abbildung 14).

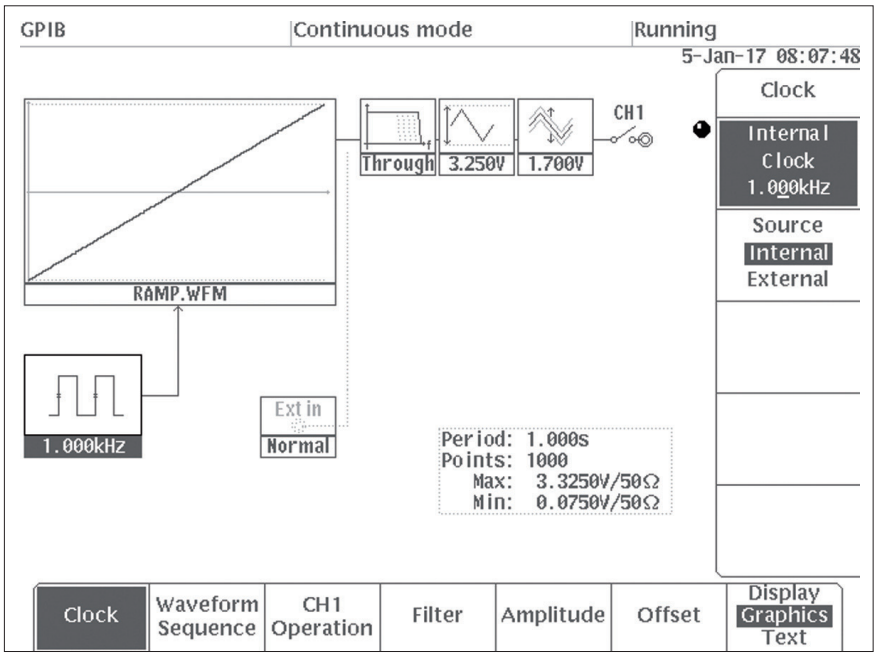


Abbildung 13: Diese Wellenform ermöglicht langsames Umschalten von Low auf High

UNGUTE TERMINIERUNG!

Aufgrund des Fehlens einer Termination von 50 Ohm schießt die Ausgangsspannung des AWG2021 im Oszillogramm auf 6V hoch, was wegen der minimalen Ausgangsleistung über die Schutzdioden im Prozessor abgefangen wird. Ignorieren Sie dies für die Dauer dieses Artikels – betrachten Sie es allerdings KEINESFALLS als Freibrief zum Anschließen beliebiger Spannungen an beliebige Prozessrechner!

ABER NUN ERNSTHAFT

Jetzt können wir uns wieder unserer eigentlichen Aufgabe, also dem Umschalten zwischen 12V und 24V zuwenden. Als erstes sollten Sie sich einen resistiven Spannungsteiler vorstellen: Da eine Dimensionierung auf 12V beim Anlegen von 24V durch Überspannung zu einer Zerstörung des Ein-gangspins führen würde, müssen Sie auf 24V dimensionieren: Bei 24V sollten Sie 3V am Eingang haben. Beim Anlegen von 12V folgt frei nach Adam Riese eine Spannung von 1.5V: Dies liegt sehr nahe an der Schaltschwelle und sollte vermieden werden.

Zur Lösung dieses Problems bietet sich die Nutzung der Zenerdiode an. Es handelt sich dabei um eine spezielle Diode, die im Grunde genommen in beide Richtungen leitfähig ist. In Durchlassrichtung verhält sie sich dabei wie eine gewöhnliche Diode, um in Sperrrichtung ab dem Anliegen einer gewissen, als Zenerspannung bezeichneten Spannung durchzubrechen.

Als erste Aufgabe müssen wir hierbei einen Vorwiderstand wählen, der die jeweils anliegende Spannung, die von der Diode nicht verbraucht wird aufnimmt. Wir dimensionieren hier bei 12 V Eingangsspannung einen Zenerstrom von 5 mA. Bei Annahme eines Spannungsabfalls von 3,3 V an der Diode folgt aus dem Ohm'schen Gesetz ein Wert von 1740 Ohm. Falls Sie diesen Wert nicht in ihrer Dekade finden, nehmen sie stattdessen einen mit 1K5.

KARREN BEISSEN!!

Eine von einem Fahrzeug gespeiste Überspannung ist wesentlich weniger dozil als ein AWG2021. Bei einem Auto reichen 6V aus, um den Raspberry PI zu zerstören.

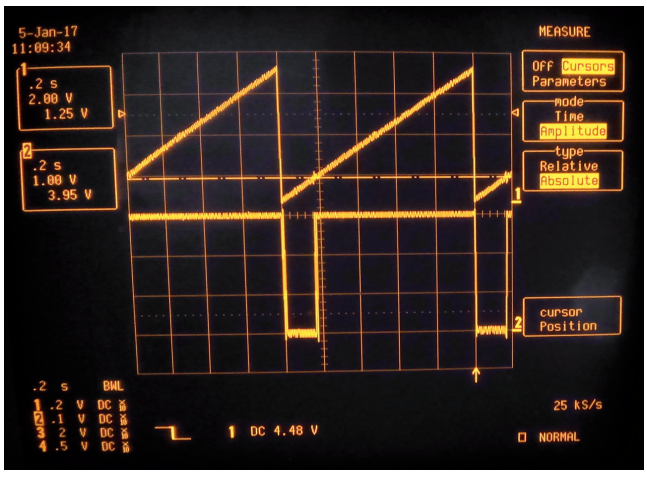


Abbildung 14: Der Schaltpegel liegt bei 1.25 Volt

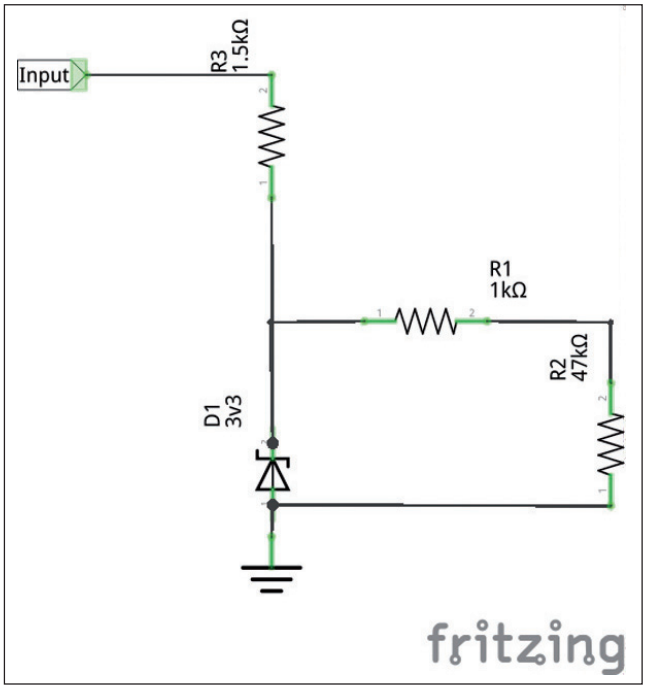


Abbildung 15: Dieser Attenuator funktioniert

Damit sind wir zur Messung der Spannungen bereit. Wir messen absichtlich ohne angeschlossenen Raspberry Pi, um das Attenuatorverhalten so genau wie möglich zu qualifizieren. Mit einem Widerstand von 1K5 kommt es dabei in der Tabelle gezeigten Messwerten - beachten Sie, dass ein gewöhnlicher Widerstand bei 24 V vergleichsweise heiß wird.

Im Zusammenspiel mit dem in der Schaltung gezeigten Vorwiderstand sind die Schutzdioden problemlos in der Lage, die überflüssigen 0,3 V abzuführen. Alternativ dazu könnten Sie natürlich auch eine Zenerdiode mit einer etwas geringeren Spannung wählen und/oder den Widerstand anders dimensionieren. Für eine gute weitere Einführung in dieses wichtige Thema - das Werk geht auch auf die hier schon aus Platzgründen nicht besprochenen kaskadierten Diodenschaltungen ein - empfehle ich den Klassiker "Analoge Schaltungen" von Seifart.

FAZIT

Auch wenn der RPi unter Windows 10 nicht bzw. mit C++ nur leidlich zum Bit-Banging von Protokollen geeignet ist: Das Ansprechen von GPIO-Pins funktioniert problemlos. Systeme mit Echtzeit-Bedürfnissen lassen sich als kombinatorische Prozessrechner realisieren – ein Thema, dem wir uns in der nächsten Ausgabe zuwenden werden. Schon an dieser Stelle sollte man allerdings anmerken, dass der Raspberry PI in manchen Situationen geradezu grenzgenial ist. Zum langsamen Ein- und Ausschalten von Motoren und ähnlichen Geräten ist der Prozessrechner auch in der vorliegenden Form geeignet. Mit einem Transistor, einer Freilaufdiode und einem Relais könnten Sie theoretisch sogar Netzspannungen schalten - achten Sie dabei allerdings darauf, dass dies in manchen Staaten staatlich zertifizierten Elektronikern vorbehalten ist.

24	3,66
22	3,62
20	3,58
18	3,52
16	3,47
14	3,408
12	3,33
10	3,23
8	3,13
6	2,94
4	2,66

TABELLE

BLUESCREEN MIT QR-CODE

Autor: HANNES PREISHUBER

Windows 10 ist bekanntlich final und deswegen wird es auch keine neuen Versionen mehr geben. Da aber Windows auch am Desktop von Microsoft nun als Service betrachtet wird, gibt es dafür doch laufende Erweiterungen per Updates, um letztlich die Nutzer bei der Stange zu halten. Diese Fortschritte werden nur anders nummeriert. 1507, 1511 und aktuell 1607 auch als Anniversary bekannt. Dieser Artikel behandelt die wichtigsten Änderungen für den Entwickler zur letzten Version

Aus Benutzersicht tut sich kaum was bei Windows 10. Mit jedem Update werden gefühlt die Blue Screens häufiger (neu mit QR-Code (s. Abbildung 1)) und überhaupt macht das Ding (vormals PC) jeden zweiten Tag zu ausgewählt unpassenden Zeitpunkten einen Reboot.

Die wenigen verfügbaren Store Apps sind optisch auch eher Rückschritt, verglichen zum Windows 8 Metro Design. Entwickler sind eben keine Designer. Die Funktionen der neuen App Generation beschränken sich auch auf das Nötigste und nutzen bei weitem nicht die Möglichkeiten eines Betriebssystems. Auch hier, wen wundert es. So richtig verbreitet ist Windows 10 auch in seiner neuesten Variante nicht.

BACK TOT HE ROOTS

Aber Microsoft tut einiges um diesem Missstand zu beseitigen. Man wird es kaum glauben, aber appx-Pakete lassen sich heute wieder einfach so installieren, ganz ohne Store. Einfach wie früher.

Auch ein bisschen Retro und überraschend, findet nun das animierte GIF wieder Einzug. Als Eigenschaft eines BitmapImages lässt es sich wie ein Video behandeln, mit Funktionen wie Play oder Stop. Ähnlich revolutionär sind Access Keys. Vom Nutzer als [ALT]+[x] oder so genutzt. Per AccessKey="t" wird so z.B. ein Button click ausgelöst. Nur wird im Gegensatz zu klassischem Windows kein _ oder sonst wie das UI Element markiert, wenn Mann/Frau [Alt] drückt. Das UI Verhalten muss allen Ernstes über zwei Events AccessKeyDisplayRequested="" und AccessKeyDisplayDismissed="" programmiert werden. In der Doku schwebt Microsoft dafür vor, die TooltipService Klasse zu nutzen. Mangels Dokumentation und Beispiel bleibt der bei der Gelegenheit auch gefundene neue AccessKeyManager allerdings mit bleibendem Fragezeichen versehen.

NEUER NAMENSRAUM

Ein weitaus größerer und neuer Bereich findet sich im Windows.Composition Namensraum. Völlig beeindruckt von der Balkengrafik, steigt der Autor in Dokumentation und

Abbildung 1: Blue Screen mit QR-Code

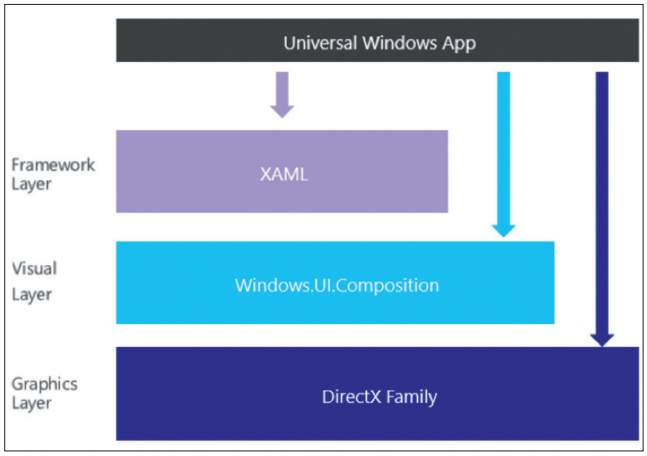


Abbildung 2: Windows.Composition

Beispiele ein. Ein Erfahrener API Designer definiert einen neuen Layer, hier Visual Layer genannt und schiebt diesen irgendwo dazwischen (s. Abbildung 2).

Kurz zusammengefasst werden damit völlig neue Möglichkeiten für Grafik und Animation möglich, die weit über Storyboards und den Effekten von UI Element hinausgehen. Zusätzlich liest man in verschiedenen Quellen, dass nicht der UI Thread damit belastet wird. Eine Aussage die vom Autor so ungeprüft stehen gelassen wird.

Wenn man sich der praktischen Anwendung von Composition widmet, stellt man schnell fest, dass die Code Beispiele das UI im Source Code definieren und nicht deklarativ per XAML. Bei näherer Betrachtung verwundert das auch nicht, da Composition ein eigner Layer ist. XAML ist davon unabhängig. Allerdings existiert die statische Klasse ElementCompositionPreview, mit derer man die Brücke zu existierenden XAML Elementen schlagen kann. In jedem Fall sieht es aber für den Designer von Visual Studio und mutmaßlich auch Blend mau aus. Will man z.B. einem Bild einen Effekt wie Schatten und/oder Graustufe hinzufügen, findet man sich schnell in umfangreichen Code wieder.

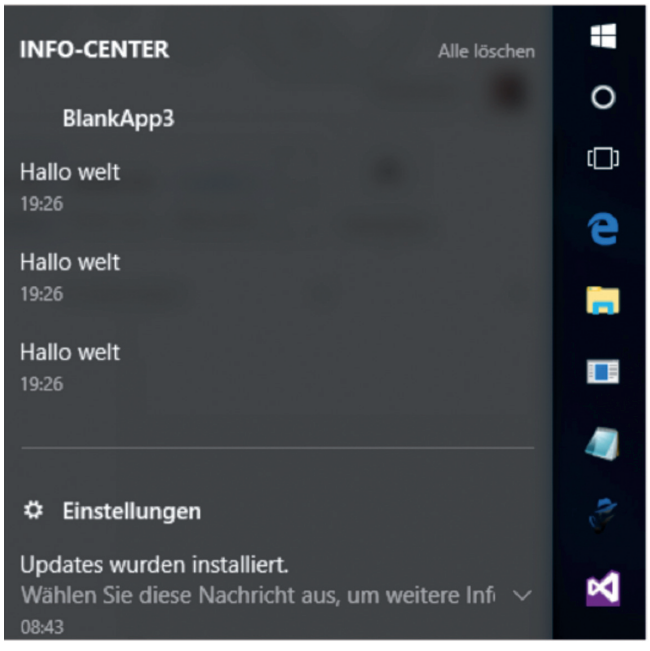


Abbildung 3: NotificationMirroring

Folgendes VB.NET Beispiel fügt einem existierenden UWP XAML UI Element ein graues Quadrat hinzu.

```
Dim compositor = ElementCompositionPreview.
GetElementVisual(image1).Compositor
Dim visual = compositor.CreateSpriteVisual()
visual.Size = New Vector2(100, 100)
visual.Offset = New Vector3(50, 50, 0)
visual.Brush = compositor.CreateColorBrush(Colors.
Silver)
ElementCompositionPreview.SetElementChildVisual(image1,
visual)
```

Um Klassen wie Vector2 oder Vector3 richtig einzuordnen, sieht man sich am besten das XNA Framework oder Unity an. Der Schluss liegt nahe, dass die Designer der UWP Anniversary Framework Erweiterungen eher die Xbox oder Hololens im Auge hatte. Das vorhandene Tooling scheint wenig für eine LOB Anwendung geeignet zu sein. Will man das halbwegs sinnvoll einsetzen, sollte man Effekte in Behaviors packen.

IM DIALOG

Auch das nächste Feature ist nicht weltbewegend, baut aber auf der Cloud Fähigkeit auf. Seit einiger Zeit poppen recht unten Dialogboxen, auf wie entgangene Anrufe von einem Windows Phone (s. Abbildung 3). Die Redmonder nennen die Funktion und Klasse NotificationMirroring. Das scheint nur von Windows Phone nach Windows PC zu funktionieren, wenn beide Geräte mit dem identen Account betrieben werden. Der Entwickler muss dazu nichts tun, außer er will es abdrehen.

Das Notification Center von Windows 10 speichert die Toast Notifications eine Weile. Der Inhalt kann wie bisher mit einem XML Format mit Grafik oder Text gefüllt werden. Jetzt neu auch mit adaptiver Layout Unterstützung. Zusätzlich kann eine App auch auf bestimmte Notifications lauschen. Als zukünftiger Einsatzzweck werden, tragbare Devices ala Microsoft Band genannt.

Das App *Livecycle Modell* wurde mit Windows 8 komplett umgestellt. Apps die nicht im Vordergrund waren, bekommen vom Betriebssystem keine Prozessor Ressourcen mehr zugewiesen und werden bei Bedarf automatisch ganz aus dem Arbeitsspeicher entfernt. Dazu werden einige Events angeboten wie onSuspending oder onLaunched angeboten. Das Konzept von der alleinigen oder später von zwei Apps hat sich allerdings als zu beschränkt erwiesen. Apps wollen auch im Hintergrund laufen um z.B. Audio abzuspielen. Dazu passend zwei neue Events EnteredBackground und LeavingBackground. Neuerdings existieren zwei weitere Events, um mit veränderten Speicherplatz Bedingungen umgehen zu können. Es macht nun somit einen Unterschied ob der Benutzer eine App minimiert oder zu einer anderen wechselt.

Mit dem Konzept der Contracts versucht Microsoft seit Windows 8, die Kommunikation von Anwendungen zu verbessern über die Funktion der Zwischenablage hinaus. In der Praxis haben nur wenige Apps die verschiedenen Sharing Contracts implementiert. Die vorhandenen Apps mit Contract kann man ganz einfach ausprobieren, indem man ein Bild mit der Windows Store App öffnet und auf *teilen* klickt. Offensichtlich war dies den freien App Entwicklern zu aufwändig. Wesentlich einfacher

Teilen

und auch verbreitet ist das Registrieren einer App mit Protokoll URL, analog zur Zuordnung von Datei-Erweiterungen zu einer oder mehreren Anwendungen. So kann der Benutzer per *tel://0867798890* einen Telefon Anruf initiieren. Per Code hilft eine Methode des Launcher Objektes um wie hier gezeigt die persönlichen Einstellungen zur Nutzung der Webcam anzuzeigen.

```
Launcher.LaunchUriAsync(new
Uri("ms-settings:privacy-webcam"));
```

Allerdings bisher ohne die Rückgabemöglichkeit aus der aufgerufenen App. Wenig überraschend, sonst wäre ja nichts dazu zu berichten, gibt es genau dieses Feature jetzt. Die aufrufende App muss das Protokoll und den *packageFamilyName* der Ziel App kennen. Um Fehler behandeln zu können, macht es Sinn per Launcher.QueryUriSupportAsync erst zu prüfen ob die passende Ziel App auch installiert ist.

Die angerufene App kennt ihren PackageFamilyName mit der Funktion *Windows.ApplicationModel.Package.Current.Id.FamilyName*. Die übergeben Werte werden in der Eigenschaft Parameter der NavigationEventArgs des Navigation Events in der Ziel App ausgelesen. Die Rückgabe von Werten erfolgt in einer Liste ebenfalls vom Typ ValueSet.

Allerdings müssen beide Apps eine Menge voneinander wissen. Das sind z.B. die übergebenen Key Value Paare in der Valueset Auflistung und das in beide Richtungen. Sowie das Protokoll und eben den Familyname. Wie gehabt und browsertypisch kann man beim Aufruf des URL Strings auch per Querystring einen oder mehrere Werte übergeben.

APPSOLUT VERKNÜPFT

Als weitere Neuerung von Windows ist es möglich eine echte URL mit einer App zu verbinden. Der Benutzer ruft die Webadresse z. B. im Menü auf und erhält, wenn installiert die passende App gestartet. Wenn die App nicht installiert ist, wird die entsprechende Website im Browser geladen. Gesteuert wird das über das appxmanifest in der AssociationLaunching.App Sektion.

```
<uap3:Extension Category="windows.appUriHandler">
        <uap3:AppUriHandler>
          <uap3:Host Name="app.ppedv.de" />
        </uap3:AppUriHandler>
</uap3:Extension>
```

Windows ruft die Website auf und sucht dort nach der JSON Datei *windows-app-web-link*. Der Inhalt steuert die Zuordnung zu einer oder mehreren lokalen Apps. Ein Hyperlink in einem Word Dokument kann so auf die Website verweisen oder falls installiert einen App öffnen.

Man hat nun also zwei Möglichkeiten mit Protokoll Launching zu arbeiten. Das App-spezifische wie *tel://* oder das URI-spezifische. Nur ersteres lässt sich per Browser Adress Eingabe aktivieren.

Wenn die aufgerufene App die Aufgabe ohne eigene und weitere UI lösen kann, spricht man von einem App Service. Der Service kann so als Task im Kontext der App laufen. Die zugrundeliegenden Mechanismen basieren auf der WinRT Api und sind nicht neu in Windows 10. Die kleine Änderung ist, dass der Background-Prozess allein lauf- und lebensfähig ist. Die Service App UWP Solution beinhaltet mindestens zwei Projekte. Das Service Projekt wird als Klassenbibliothek angelegt und eine darin enthaltene Klasse implementiert das Interface IBackGroundTask. Damit sollte Visual Studio einen Methodendummy Run anlegen, indem die Programmlogik codiert wird.

Im Unterschied zu Windows 8, muss allerdings der App Service um lokal aktiviert werden zu können, im APPX Manifest den passenden Eintrag aufweisen.

```
<Extensions>
    <uap:Extension Category="windows.
    appService" EntryPoint="RandomNumberService.
    RandomNumberGeneratorTask">
      <uap3:AppService Name="com.microsoft.
      randomnumbergenerator"
      SupportsRemoteSystems="true"/>
      </uap:Extension>
</Extensions>
```

Microsoft nennt die Apps mit beigeordneten Backgroundtask Service auch App Service Provider in seinen API Funktionsnamen.

Eine Client APP muss von der Service App den *PackageFamilyName* und AppService wissen. Letzteren definiert der Service App Entwickler im Manifest wie vorhin gezeigt. Den *FamilyName* erfährt man per Visual Studio visuellen Manifest Editor.

Außerdem kann man in der Service App beim Start durch den Benutzer diese ID per Package.Current.Id.FamilyName anzeigen lassen (s. Abbildung 4). Diesen Dialog sieht der Benutzer nicht, wenn die lokale Aktivierung der Service App über die Client App erfolgt.

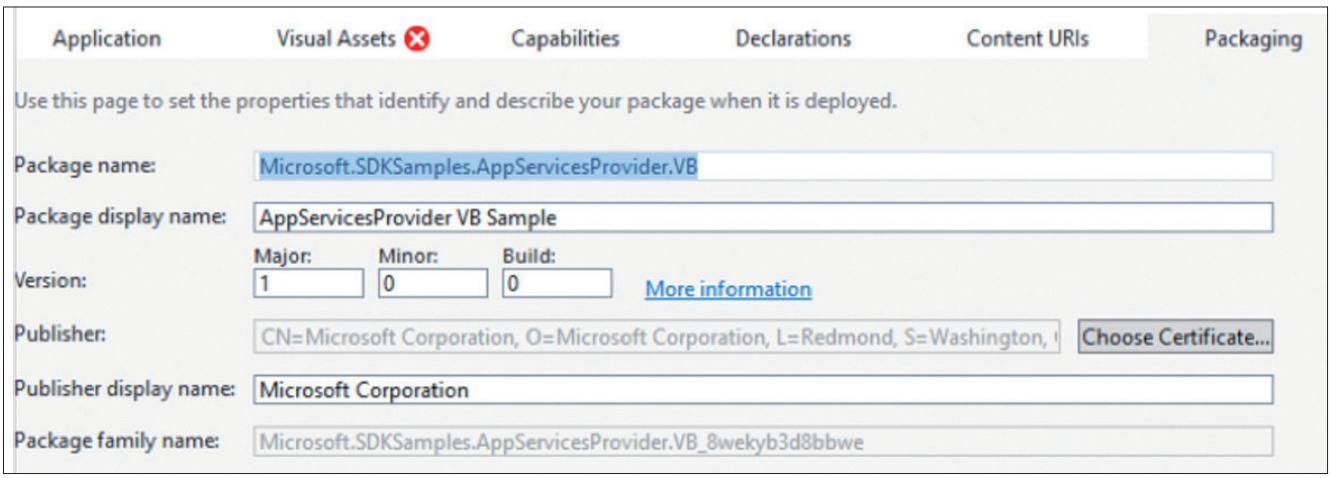


Abbildung 4: Package.Current.Id.FamilyName anzeigen lassen

Laut Dokumentation kann man in der Client App alle verfügbaren, Services einer bestimmten Familie auflisten. Der Befehl dazu lautet:

```
var services = await AppService
Catalog.
      FindAppServiceProvidersAsync
      (”com.my.appservice“);
```

Allerdings sucht man vergeblich nach passenden Anwendungsbeispielen bzw. einer guten Dokumentation. Da bleibt in dieser Phase nur die Möglichkeit die AppServiceConnection mit beiden Parametern (Service und Name) direkt zu erstellen.

```
connection = New
AppServiceConnection()
connection.AppServiceName = ”com.
microsoft.randomnumbergenerator“
connection.PackageFamilyName = ”Mi-
crosoft.SDKSamples.AppServicesProvi-
der.VB_8wekyb3d8bbwe“
Dim status = Await connection.
OpenAsync()
```

Das erzeugte Objekt steht nur während der Nutzung der Verbindung durch den Client zur Verfügung. Wird die Client App beendet, wird auch der App Service beendet. Das Betriebssystem kann sich weigern einen App Service zu starten - meist wegen mangelnder Ressourcen - so dass eine adäquate Fehlerbehandlung von Nöten ist.

Auf einer Connection werden Nachrichten mit SendMessageAsync zwischen App und Service ausgetauscht. Die Daten werden in ein Wörterbuch vom Type ValueSet gelegt und als Parameter dem Funktionsaufruf mitgegeben. Da der Aufruf asynchron erfolgt, kann er Client auf die Antwort warten (await). Diese ist vom Typ AppServiceResponse und beinhaltet Status und Message.

APP RECYCLING

Im Ergebnis könnte man so ein Konzept für granulare und erweiterbare Apps erstellen. Aus Architektursicht eleganter wurde das in der Vergangenheit mit dem MEF (Managed Extensibilty Framework) gelöst. Wobei auch Windows 10 Anniversary ein echtes Plugin Konzept mit App Extensions anbietet. Die Idee ist bestechend: Statt immer wieder das Rad neu zu erfinden, werden Funktionen bestehender Apps eingebunden und genutzt. Ein Service Konzept passend zu Azure. Der benötigte Namensraum lautet Windows.ApplicationModel.AppExtensions.

Die Windows App und seine installierten Erweiterungen finden über einen gemeinsamen Namen definiert als Extension in der Manifest Datei zusammen. Im Host windows.appExtensionHost und in den Projekten, die die Erweiterungen darstellen windows.appExtension.

```
<Extensions>
        <uap3:Extension Category=“windows.appExtension“>
          <uap3:AppExtension Name=“build2016.appextensibility.demo“
Id=“base“ PublicFolder=“Public“ DisplayName=“Invert“ Description=“Invert
Image“ />
        </uap3:Extension>
</Extensions>
```

Für die Erweiterung uap3 wird ein eigener Namensraum bereitgestellt, der natürlich auch eingebunden werden muss.

```
xmlns:uap3=“http://schemas.microsoft.com/appx/manifest/uap/windows10/3“
```

Auch die Extensions sind Windows 10 Store Apps, die mit den üblichen Methoden verteilt oder deinstalliert werden können. Um an die richtige Stelle in der Extension zu gelangen, sind aktuell zwei Wege erkennbar. Beide werden über die Extension Sektion in der Datei package.appxmanifest gesteuert. Im Falle des InvertImages Beispiels von vorher, wird ein Pfad per PublicFolder definiert. Dieser kann dem Austausch von größeren Daten zwischen App und Extension dienen. Man kann aber auch eine HTML Datei mit dem Namen Extension dort platzieren. In dieser findet sich JavaScript Code der die Logik abwickelt. Da dies einen Bruch in der Programmiersprache darstellt und auch einigermaßen seltsam abgewickelt wird, wird hier nur ein JavaScript Code Beispiel ohne weitere Kommentierung gezeigt.

```
<canvas id=“inPicture“ class=“image-container“></canvas>
<canvas id=“outPicture“ class=“image-container“></canvas>
<script type=“text/javascript“>
        var tempImageStore = new Image();
        var inCanvas = document.getElementById(‚inPicture‘),
            inContext = inCanvas.getContext(‚2d‘);
        var outCanvas = document.getElementById(‚outPicture‘),
            outContext = outCanvas.getContext(‚2d‘);

        function extensionLoad(url) {
            tempImageStore.src = url;
            inCanvas.height = tempImageStore.height;
            inCanvas.width = tempImageStore.width;
            inContext.drawImage(tempImageStore, 0, 0);

            outCanvas.setAttribute(‚width‘, tempImageStore.height);
            outCanvas.setAttribute(‚height‘, tempImageStore.width);
            outContext.rotate(-90 * Math.PI / 180);

            outContext.drawImage(tempImageStore, -tempImageStore.width, 0);

            var sendYourDataUrl = outCanvas.toDataURL();
            console.log(”Done“);

            // this is the callback
            window.external.notify(sendYourDataUrl);
        }
```

In einer streng typisierten und managed Code Umgebung, kann das leider nicht so einfach funktionieren. Microsofts Entwickler lösen die Aufgabe mit einem zwischengeschalteten App Service.

```
<Extensions>
  <uap:Extension Category=“windows.appService“ EntryPoint=“GrayscaleService.
Service“>
    <uap:AppService Name=“com.microsoft.grayscaleservice“ />
   </uap:Extension>
 <uap3:Extension Category=“windows.appExtension“>
 <uap3:AppExtension Name=“build2016.appextensibility.demo“ Id=“base“
PublicFolder=“Public“ DisplayName=“Grayscale“ Description=“Grayscale
Image“>
     <uap3:Properties>
         <Service>com.microsoft.grayscaleservice</Service>
     </uap3:Properties>
```

```
  </uap3:AppExtension>
  </uap3:Extension>
</Extensions>
```

ÜBERGABE DER APP AN DIE EXTENSION

Um das Problem des fehlenden UI, bei der Übergabe der APP an die Extension zu lösen, muss die Extension einen Background Prozess mit der Logik betrauen. Dieser läuft im Kontext der aufrufenden App. Somit muss die Windows UWP Solution, welche die Extension Logik beherbergt, ein UWP Projekt und ein Universal Windows Klassenbibliotheks Projekt enthalten.

Letzteres implementiert das Interface IBackgroundTask und damit eine Run Methode als Einstiegsfunktion. Mit einem Objekt vom Typ AppServiceTriggerDetails wird dann die Kommunikation von Daten und Events abgewickelt. Die wichtigste Methode übergibt sowohl Daten, als auch Commands.

```
OnRequestReceived(AppServiceConnection sender, AppServiceRequestReceivedEventArgs args)
```

Mit Hilfe der Kommandos kann der Service mehrere Aufgaben übernehmen. Das Kommando Load muss per Konvention immer implementiert werden. Siehe auch JavaScript ExtensionLoad. Die Übergeben Parameter findet man in args.Rquest.Message als Dictionary Typ ValueSet. Nach getaner Arbeit wird args.Request.SendResponseAsync(returnData) aufgerufen.

Soweit das Extension Konzept. Die App muss beim Start und wenn man möchte auch nachher seine verfügbaren Erweiterungen finden, darstellen und aktivieren können. Dazu wird AppExtensionCatalog.Open mit dem Namen der ExtensionGruppe aufgerufen und so das Objekt erzeugt, das die Liste der zugehörigen Erweiterungen beinhaltet. Dies ist der Idente Wert wie im Manifest bereits definiert.

Das ist aus UI Sicht schon nicht trivial. Gelöst wird dies mit einer unter dem MVVM Pattern gebundene Liste. Die Liste muss fähig sein, das UI über Änderungen zu informieren, wie das eine ObervableCollection leistet.

Gekapselt wird dies im Microsoft Beispiel in eine Klasse ExtensionManager. Dieser statisch implementierte ExtensionManager ist der Dreh- und Angelpunkt für die Liste der Extensions, Events und Methoden. Per Enable Eigenschaft, gebunden und gesteuert über Checkboxen aus der in Abbildung 5 gezeigten UI, werden diese erst in der App vom Benutzer aktiviert. Will man auch nachträglich hinzugefügte Erweiterungen im laufenden Betrieb der Host App benutzen können, wird der Event Catalog_PackageInstalled genutzt. Im Code der abonnierten Methode fügt man die neue Erweiterung in die Liste hinzu, die wiederum, da Observable, das UI aktualisiert.

Im nächsten Schritt einer UI, müssen dann die Erweiterungen z.B. in einer Toolbar auch verwendbar werden. Dazu wird eine Liste von Buttons an die verfügbaren Extensions gebunden. Im Button Click wird dann das Load Kommando samt Parametern initiiert.

```
ext.InvokeLoad(ImageTools.AddDataURIHeader(AppData.currentImageString));
```

So steht dem eigenen App Ecosystem kaum mehr was im Wege.

BASIS COMPONENT OBJECT MODEL

Im Zuge der Recherchen zu diesem Artikel wurde immer wieder bewusst, dass die zugrundeliegenden APIs auf Betriebssystemebene nach wie vor auf COM basieren. Das Component Object Model war einfach die Grundlage für den Abschied von der großen monolithischen Anwendung im Windows Umfeld. Mit seiner Erweiterung DCOM war es schon vor 20 Jahren möglich Komponenten auf verschiedene Computer zu verteilen und remote zu nutzen. Im Zuge der Client Server oder Mehrschicht-Architekturen ist dieser Ansatz zu Gunsten von Application Servern ein wenig in Vergessenheit geraten. Nun erfinden die Redmonder das Rad gerade wieder einmal neu und nennen es Projekt Rome. Remote Application Services ist eine zentrale Neuerung des Anniversary Updates. Es erlaubt von einem Gerät an ein anderes Daten zu übergeben und eine App zu starten die diese verarbeiten kann. Eine Aufgabe wird an einem Gerät gestartet und an anderem Gerät weiter geführt. Mit der Device Familie IoT (Raspberry), Phone, Desktop, Xbox oder auch Hololens sind dazu eine Reihe Szenarien denkbar. Das einfachste ist noch, ich schaue einen Film auf dem Phone und übergebe an die Xbox. Der Kleber dazu ist ein Microsoft Account (MSA), unter dem alle Geräte registriert sein müssen. Die eigentliche Kommunikation zwischen den Geräten muss dann nicht über die Cloud, sondern kann auch lokal im Netzwerk oder per Bluetooth erfolgen. Soweit dokumentiert, kann auch die initiale Verbindung ohne Cloud über eine Annäherung lokal erfolgen. Das wirft eine Reihe Fragen auf, aber wir stehen damit auch erst am Anfang. Rom wurde auch nicht an einem Tag erbaut. Alle Wege führen nach Rom oder simples Roaming?

REMOTE DEVICE

So kann heute bereits der Windows PC des Autors die Notifications des Windows Smartphones anzeigen. Sei es ein entgangener Anruf oder eine SMS. Wenn man als Benutzer solche Nachrichten als erledigt wegklickt, geschieht dies auch auf allen verbundenen Devices.

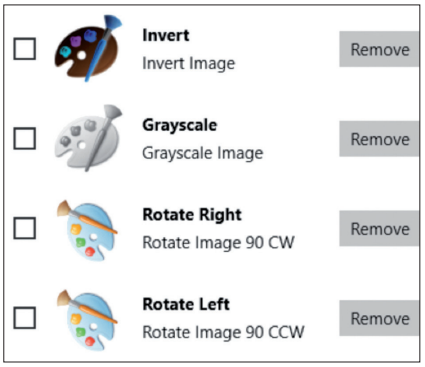


Abbildung 5: Checkboxen zur Aktivierung

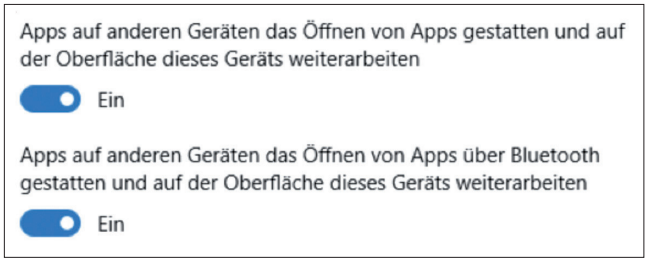


Abbildung 6: Datenschutz Einstellungen für App Handover

In der nächsten Version von Windows 10 dem Creators update, soll es auch mit verbundenen Freunden möglich sein, Apps remote zu aktivieren.

Technisch orientiert sich die Lösung am bisher besprochenen Launch per URI und den App Services. Der Client muss allerdings den RemoteLauncher nutzen um z.B. eine URI auf einem anderen Gerät aufzurufen.

```
launchUriStatus = await RemoteLaun-
cher.LaunchUriAsync(new RemoteSyste
mConnectionRequest(selectedDevice),
uri);
```

Im Status steckt dann die Information, ob das Remote Device erfolgreich die App gestartet hat. Das klappt im Versuch auch bei einem ausgeschalteten Windows Phone in 2-8 Sekunden (success), sofern das Gerät eine Wifi Verbindung hat. Falls nach 30 Sekunden niemand antwortet, wird eine Meldung RemoteSystemUnavailable genannt.

Der Benutzer kann über die Datenschutz Einstellungen, kontrollieren ob dieser App Handover erlaubt ist. Bei einem deutschsprachigen Windows Phone lautet der Menüpunkt App-Oberfläche weiter nutzen. Auf einem Windows 10 Desktop unter Datenschutz-Allgemein. Der Punkt Aktivierung per Bluetooth findet sich nur am Desktop.

Im ersten Schritt erhält man per RemoteSystem.CreateWatcher eine Liste von verfügbaren Geräten. Diese könnten gefiltert werden nach Art des Gerätes (z.B. Xbox) oder nach Verbindung (Cloud, Proximal). In Anbetracht von zwei bescheidenen Devices ist dieser eher überflüssig. Die Verbindung per Annäherung (Bluetooth) hat überhaupt nicht geklappt. Proximal soll neben Bluetooth auch eine direkte Verbindung über Wifi ermöglichen. Dieses Szenario ließ sich nicht testen, da nicht auszuschließen ist, dass bei bestehender Internet Verbindung beide Geräte den Cloud Service in Anspruch nehmen.

Im Manifest finden sich an zwei Stellen Einträge für remote Aktivierung. Der Service muss entsprechend mit dem Attribut versehen werden und die App benötigt die Capability. Beides ist nur direkt im XML erstellbar.

```
<uap:Extension Category="windows.
appService">
     <uap3:AppService
Name="com.project-rome.echo"
SupportsRemoteSystems="true" />
</uap:Extension>
<Capabilities>
    <Capability
Name="internetClient" />
    <uap3:Capability
Name="remoteSystem"/>
```

Ebenfalls noch ohne erfolgreiche Forschungsergebnisse ist die Möglichkeit ein Gerät anzusprechen, das nicht mit der gleichen MSID angemeldet ist. Dies geschieht per Hostname oder IP Adressen.

```
var deviceHost = new Windows.Net-
working.HostName(IPaddress);
var remotesys = await RemoteSystem.
FindByHostNameAsync(deviceHost);
```

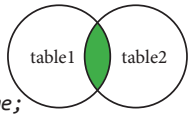
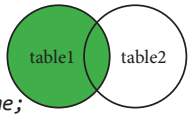
FAZIT

Zusammenfassend ergeben sich damit vier Möglichkeiten von connected Apps

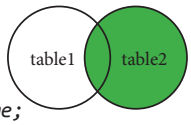
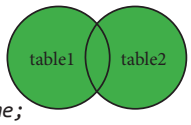
- App Handover
- App Services
- App Extensions
- App Across Devices-Project Rom

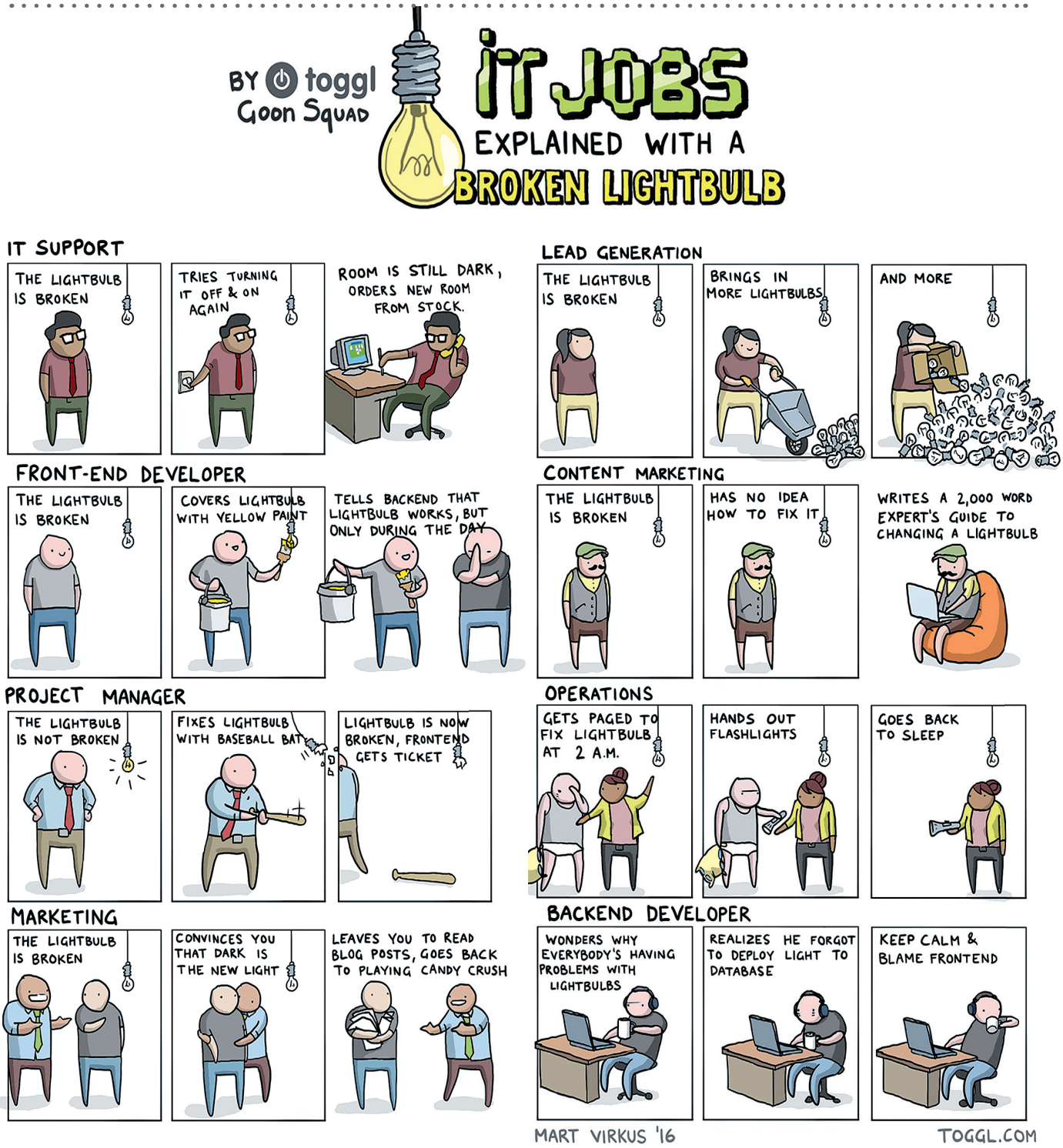
Windows 10 schlägt damit ein neues Kapitel in der Anwendungsentwicklung auf. Connected Apps über Device Grenzen hinweg erfordern neue Architekturkonzepte. Ob es im Web Stack dazu eine vernünftige Alternative gibt darf bezweifelt werden. Kommt so das Revival der nativen Anwendungen und damit XAML und .NET Sprachen? Hoffnung besteht, dass diese Konzepte, wenn auch langfristig, greifen. In Zukunft könnte eine APP ohne Connectivity schlicht vom Benutzer als defekt betrachtet werden.

SQL INNER JOIN	SQL LEFT JOIN

`SELECT column_name(s)`
`FROM table1`
`INNER JOIN table2`
`ON table1.column_name=table2.column_name;`
 
	`SELECT column_name(s)`
	`FROM table1`
	`INNER JOIN table2`
	`ON table1.column_name=table2.column_name;`
	 

SQL RIGHT JOIN	**SQL FULL OUTER JOIN**

`SELECT column_name(s)`
`FROM table1`
`INNER JOIN table2`
`ON table1.column_name=table2.column_name;`
 
	`SELECT column_name(s)`
	`FROM table1`
	`INNER JOIN table2`
	`ON table1.column_name=table2.column_name;`
	 



> ZUM VORAUSPLANEN

WAS IST LOS IN DEUTSCHLAND?

21. März 2017, 19:00 Uhr in München

Azure Xpress Route Deep Dive & Office 365 Dev Patterns & Practicies

Ort: Microsoft Deutschland GmbH, Walter-Gropius-Str. 5, München

Florian Klaffenbach (MVP) vermittelt die Grundlagen von Azure ExpressRoute und welche Informationen zwischen Kunde und ISP für ein erfolgreiches Deployment ausgetauscht werden müssen. Im Rahmen einer Live Demo wird eine Express-Route Connection beispielhaft deployed. Nicki Borell (MVP) spricht über die 2013 von Microsoft ins Leben gerufene Office 365 Dev Patterns & Practicies Initiative. Hauptziel war es Kunden bei der Überführung von klassischem Fulltrust-Code hin zu Lösungen basierend auf dem Add-in Model zu unterstützen. Inzwischen hat sich die OpenSource Plattform wesentlich weiterentwickelt.

http://bit.ly/2jYRPTu

22. April 2017

Global Azure Bootcamp 2017 by ppedv

Ort: ppedv AG, Chiemgaustraße 116, 81549 München

http://bit.ly/2jGCDcF

10. und 11. Mai 2017 in Magdeburg

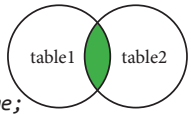
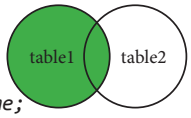
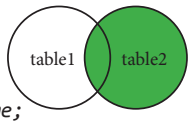
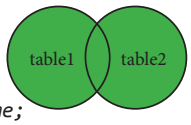
http://md-devdays.de/

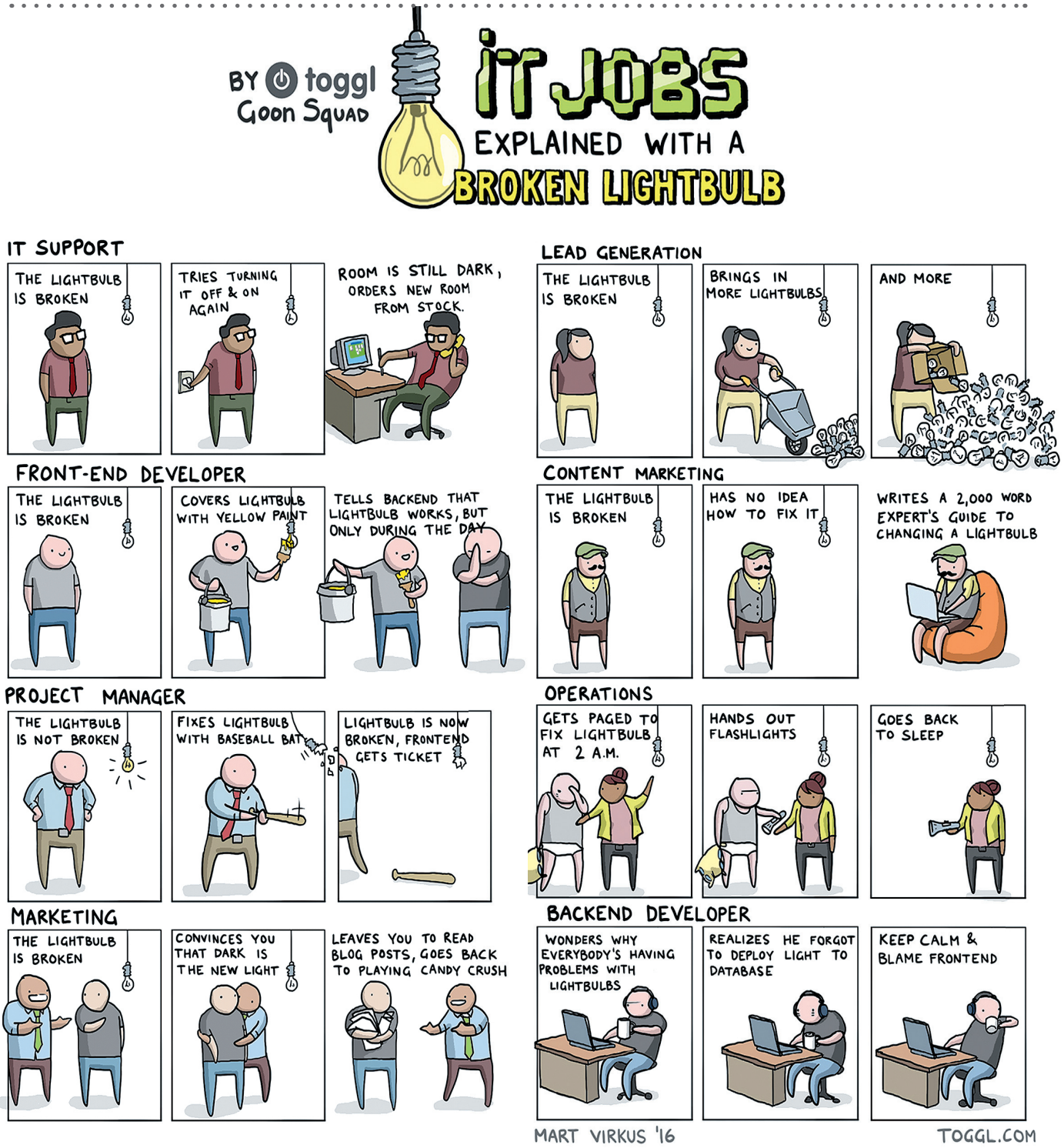
Am 10. Mai 2017 ist es wieder so weit: Dann startet die 2. Entwicklerkonferenz im KONGRESS & KULTURWERK – fichte in Magdeburg! Wir freuen uns über die zahlreichen hilfsbereiten Unterstützer, die es uns ermöglichen ein erstklassiges Programm mit renommierten Speakern rund um die Themen .NET, JAVA, HTML 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

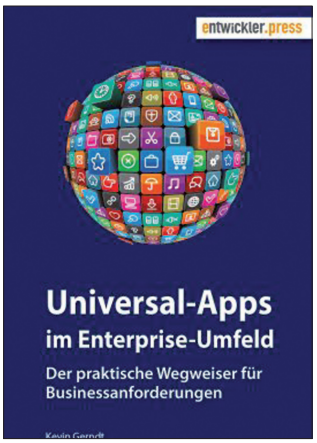
SQL INNER JOIN	SQL LEFT JOIN
SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name; 	SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name; 
SQL RIGHT JOIN	**SQL FULL OUTER JOIN**
SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name; 	SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name; 



UNIVERSAL-APPS IM ENTERPRISE-UMFELD

DER PRAKTISCHE WEGWEISER FÜR BUSINESSANFORDERUNGEN

von Kevin Gerndt



Buch: 24,90€
180 Seiten
ISBN 9783868021660 /
PDF: 19,99€ / PDF-
ISBN: 9783868023534 /
EPUB: 19,99€ / EPUB-
ISBN: 9783868026917

Dieses Buch richtet sich primär an Softwareentwickler in Unternehmen, die sich intensiv mit der Thematik der Universal-App-Entwicklung beschäftigen möchten. Neben technischen Aspekten stehen Prototyping, die Inbetriebnahme und verschiedene Möglichkeiten der App-Verteilung im Fokus. Praxisnahe Beispiele zeigen, wie sich Businessanforderungen herausarbeiten und in Form von Universal-Apps abbilden lassen. Zusätzlich erhält der Leser tiefe Einblicke in alle wichtigen APIs, wie etwa den Zugriff auf verschiedene Sensoren.

http://bit.ly/2jyObi4

FOLLOW ME!

ERFOLGREICHES SOCIAL MEDIA MARKETING MIT FACEBOOK, TWITTER UND CO.

von Anne Grabs, Elisabeth Vogl, Karim-Patrick Bannour

Buch 34,90€ /
E-Book 30,90€ /
Sofort verfügbar /
604 Seiten / 4., aktualisierte Auflage 2016 /
ISBN 9783836241243

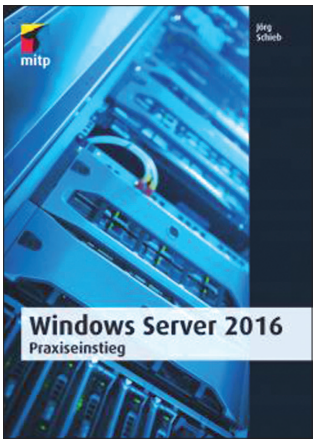
Der bewährte Begleiter durch die Welt des Social Media Marketings, jetzt in neuer, aktualisierter Auflage! Anne Grabs, Elisabeth Vogl und Karim-Patrick Bannour zeigen Ihnen, dass es für Unternehmen jeder Branche und jeder Größe lohnenswert ist, in Social Media aktiv zu werden. »Follow me!« liefert Ihnen praktische Tipps mit zahlreichen Best Practices sowie Praxiseinblicken und Erfahrungswerten von Social Media Managern.

http://bit.ly/2kLBWQV

WINDOWS SERVER 2016

WINDOWS SERVER 2016 OPTIMAL EINRICHTEN

von Jörg Schieb



Buch / E-Book /
ab 38,99€ / 576 Seiten
1. Auflage 2017 /
ISBN 9783958454774

Dieses Buch ist der perfekte Begleiter für einen praxisnahen Einstieg in die Administration mit Windows Server 2016 und zeigt, wie Sie das System bestmöglich für sich nutzen können. In einem ersten Teil wird alles erklärt, was für die ersten Schritte mit Windows Server 2016 wichtig ist: Sie erfahren, welche Kosten auf Sie zukommen, welche Voraussetzungen Ihr System mitbringen muss und wie Sie das System auf dem Server einrichten.

http://bit.ly/2jETbD3

DAS ESP8266-PROJEKTBUCH

HEIMAUTOMATION MIT DEM WLAN-CHIP

von Martin Mohr

24,90€ / 152 Seiten
ISBN 9783868021684 /
PDF 19,99€ / PDF-
ISBN: 9783868023558 /
EPUB 19,99 € / EPUB-
ISBN 9783868026931

Dieses Buch liefert einen praxisnahen Einstieg in die vielfältige Welt des ESP8266 und seine Entwicklungsumgebung, die auf der leicht bedienbaren Arduino IDE basiert. Den Kern des Werkes bilden fünf Projekte mit unterschiedlichem Schwierigkeitsgrad, die auf verschiedene Weise zeigen, wie der ESP8266 für IoT- oder Smart-Home-Projekte eingesetzt werden kann.

http://bit.ly/2kczkeZ

ON ERROR RESUME NEXT

Diskutieren Sie gerne über Sinn und Vorteil ihrer Lieblingsprogrammiersprache? Dann kann ich nur erwidern, Basic ist das Beste seit der Erfindung von geschnitten Brot. Nicht? Visual Basic ist nicht nur eine Sprache, es beschreibt auch ein Stück den Nutzertyp.

Zugeschrieben wird die Sprache gern dem Microsoft Gründer. Er hat sie aber auch nur verwendet, für sein erstes Betriebssystem. Es gibt nicht viel, was vor meiner Geburt in der IT erfunden wurde, Basic gehört dazu. Es war wohl ein schöner Herbsttag 1964 am Campus Dartmouth, als John G. Kemeny und Thomas E. Kurtz vor ihren Terminals saßen und die erste Hochsprache entwarfen. Der dabei entstandene GOTO Befehl wird auch heute noch fälschlicherweise immer mit BASIC in Verbindung gebracht. Die beiden hatten was Einfaches für Ihre Studenten im Sinn und der Name spiegelt das wieder.

BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE

Diese Allzweckwaffe fand viele Fans auch in der ersten Heimcomputerwelle. Ob Atari oder C64, alle nutzen und konnten Basic. Folglich hat Bill Gates dieses weit verbreitete Wissen clever genutzt, um für sein DOS von Start weg eine Entwicklerbasis aufweisen zu können. Basic war Marktführer.

Statt kryptischen Codes wie && lehnt sich die Syntax an die natürliche Sprache an und verwendet ein AND. Dabei gingen die Redmonder Entwickler sogar soweit, dass in frühen Office Versionen einen lokalisierte Visual Basic for Applications Syntax verwendet werden konnte. Auch folgender Deutscher VBA Code konnte den Siegeszug nicht stoppen

```
Funktion VorherigerGeschaeftstag(dt
Als Datum) Als Datum
  Dim wd as integer
  wd = Wochentag(dt)
  Prüfe Fall wd
        Fall 1
           VorherigerGeschaeftstag
           = dt - 2
        Fall Sonst
           VorherigerGeschaeftstag
           = dt - 1
   Ende Prüfe
Ende Funktion
```

Aber der Ruf war ruiniert. Visual Basic ist langsam und die Entwickler schreiben schmutzigen Code. Erst mit Visual Basic 6 und dem dazugehörigen Visual Studio erreicht die Desktop Programmierung seinen produktiven Höhepunkt. Eigentlich verdrahten zu dem Zeitpunkt die meisten Entwickler nur mehr vorhandene COM Objekte mit wenigen Zeilen Code. Es war auch der Höhepunkt der Drittanbieter Hersteller. Wer ein passende Datagrid wollte, hat vielleicht TrueDBGrid erworben.

Kaum einer ist auf die Idee gekommen das selbst zu kodieren. Rechnet sich schlicht nicht. Es war die Zeit der Menschen, die wussten wie ihre Geschäftsprozesse ticken und diese mit mäßigen Programmierskills digitalisierten. Das dabei genutzte Windows Forms findet sich heute in Geschäftsanwendungen noch weit verbreitet. Wer dagegen Access als den Kombiweg aus Datenbank und VBA wählte, wird diese Entscheidung eher bereut haben.

Eine weitere Sackgasse hat vermutlich Bill Gates persönlich mit seinem JavaScript Konkurrent Visual Basic Script (VBS) als Browser Engine gebaut. Technologisch zu dem Zeitpunkt durchaus überlegen, aber mit der IE Plattform mehr oder weniger gleichzeitig zu Grabe getragen.

Die Zeiten ändern sich, das Geschäft professionalisierte sich und Microsoft publizierte den Java Konkurrent .NET und damit Visual Basic .NET. Die Leichtigkeit war für alle Zeit dahin. Architektur, Design Patterns, Interfaces und Co sind keine Domäne von Business Know How Trägern. Code muss nun Clean sein. Auf der ewigen Suche nach Verbesserung auch testbar mit Unit Tests. Auf der Strecke blieben die vielen kleinen Programmierer, die einfach ihr Problem gelöst haben wollen. Basic ist Bäh geworden. Vermutlichen ahnen die meisten Besucher der Basta Konferenz nicht, wo sie sich befinden, nämlich auf den BASic TAgen.

Je elitärer C# in seinen Möglichkeiten wird, je weiter weg sind sie, die Semipros. „Ich will doch nur ein Formular, am Desktop, im Web", murmeln sie leise. Wer es laut sagt, wird per Shitstrom niedergemacht. Winforms ist veraltet und Webforms sowieso ganz furchtbar. UI Design Tools sind ohnehin für Mausschubser, wir die coolen Kids coden JavaScript auf der Kommandozeile.

Die Kampagne hat Erfolg. Es wurde ruhig um die VB Community. Zwar wird die Sprache noch immer bei jeder Microsoft Technologie unterstützt. Manchmal aber erst später, oft fehlt in der Doku der VB Parts oder die Beispiele. Wir warten auf die Rente, um dann mit den Foxpro Entwicklern über die gute alte Zeit zu philosophieren. Wobei philosophisch VB mein Lebensmotto gestaltet hat „On Error Resume Next".

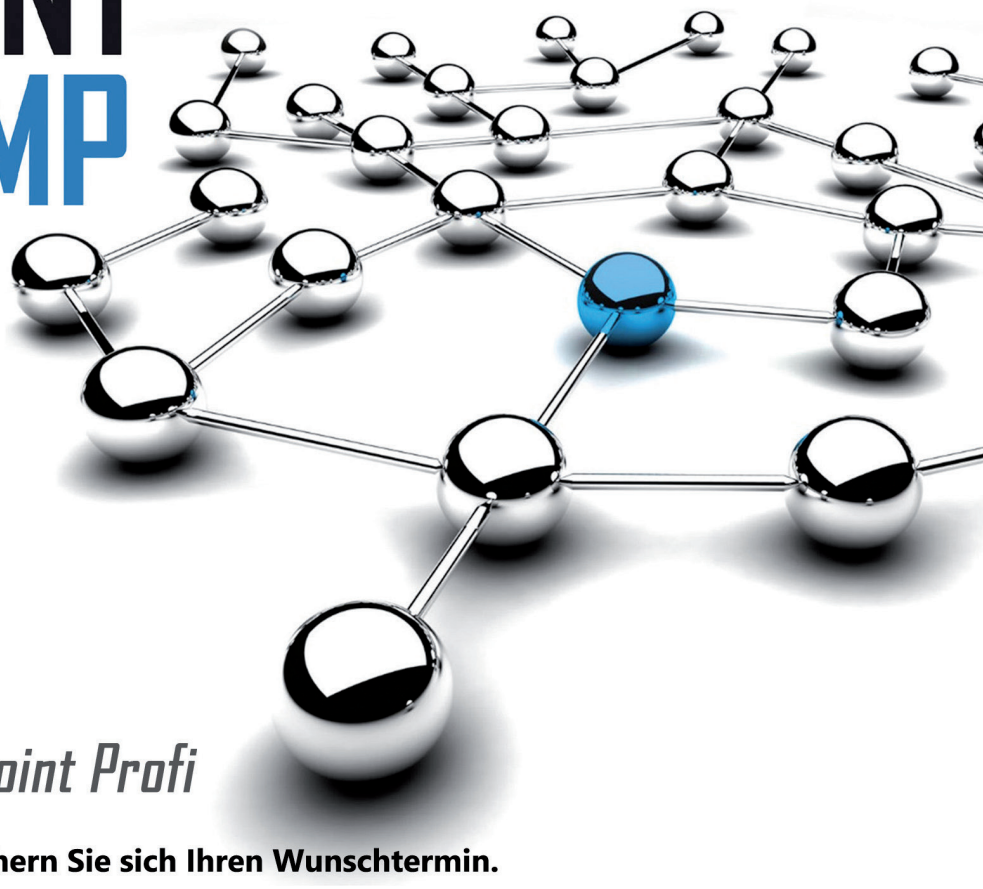
Damit könnte ich schließen und würde das im Genre „trauriges Familiendrama" ablegen. Aber man glaubt es kaum, Anfang Februar 2017 erwacht das MSDN Visual Basic Blog wieder zum Leben. Es gibt zwei Artikel, die sich mit Strategie und Zukunft befassen. Noch ist es zu früh von einem Happy End zu sprechen, aber die Überraschung weckt Hoffnung. Meine alternative Schusshandlung lautet in etwa so.

.NET Core erhält einen XAML Layer. XAML wird einfacher. Visual Studio enthält einen guten UI Designer, so dass per Drag&Drop aus der Werkzeugleiste ein Button und dann sein Click Event einfach implementiert werden können. Datenbindung funktioniert komplett per Drag&Drop, so dass man die Datenbank wieder verbinden kann. Es gibt Reports, Chart Controls, Drucken, Eingabevalidierung, lokale Datenbank und alles was 90% der Line Of Business Applications so brauchen, um einfach schnell per Prototyp eine Anwendung zu erhalten. Und wenn sie nicht gestorben sind, klicken sie noch heute ihre Visual Basic Anwendung zusammen.

Hannes Preishuber
Visual Basic Programmierer

SHAREPOINT CAMP

Server 2013



München
03. - 07. April

Berlin
27. - 31. März

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
20. - 24. März

Die besondere Camp Idee der ppedv bietet das optimale Lernkonzept mit mehr Praxisübungen, Industrie-Experten und besonderem Lernambiente - ab 2.499 €

Mehr Informationen unter:
http://xamarin.camp

ppedv

IT-Akademie
MINERVA

Erklimmen Sie die Spitze der Zertifizierung!

MCSA / MCSE Windows Server 2016

Zertifizierung kompakt in 9 / 12 Tagen

Inkl.
Exam Replay
Voucher

Ihr Zertifizierungsexperte MINERVA bringt Sie zum MCSA, MCSE, MCSD.

Fordern Sie jetzt Ihr kostenloses Angebot an:

http://minerva.ppedv.de | minerva@ppedv.de | +49 (0) 8677 - 988 966

powered by ppedv AG, Marktler Straße 15 b, 84489 Burghausen, Traunstein HRB 12703, St.Nr. 131-45412, Vorstand Hannes Preishuber.