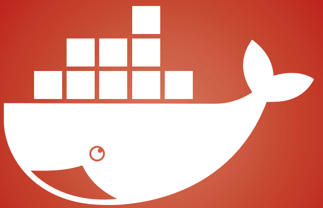
THE MICROSOFT JOURNAL FOR DEVELOPERS

APRIL 2017 VOL 32 NO 4

msdn magazine



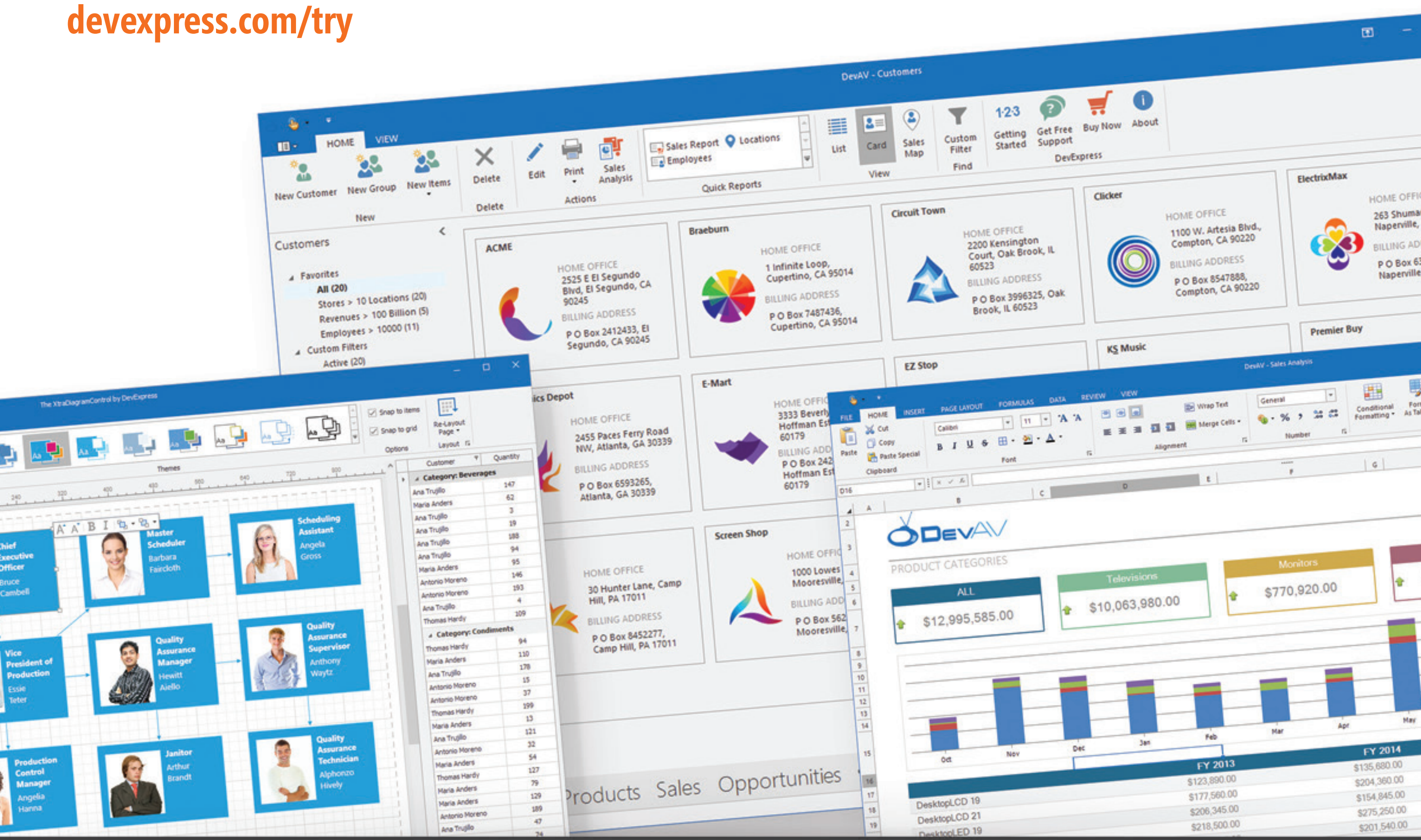
Container Development with Docker....................16, 20

DevExpress®

The King of the Desktop

Your peers have voted DevExpress Desktop Components best-in-class for 5 straight years.
We invite you to see why. Download your free 30-day trial today.

devexpress.com/try



Your Next Great Desktop App Starts Here

From apps that replicate the look and feel of Microsoft Office® to high-powered data mining and decision support systems for your enterprise, DevExpress desktop controls will help you build your best, without limits or compromise.

Download your free 30-day trial
an experience the DevExpress difference today.

devexpress.com/try

THE MICROSOFT JOURNAL FOR DEVELOPERS

APRIL 2017 VOL 32 NO 4

msdn magazine

Microsoft

Write Fast, Run Fast

with **Infragistics Ultimate** developer toolkit

UI Controls & Productivity Tools For Quickly Building High-Performing Web, Mobile, and Desktop Apps

Download a free trial at
Infragistics.com/Ultimate

INFRAGISTICS

To speak with our sales team or request a product demo call: 1.800.321.8588

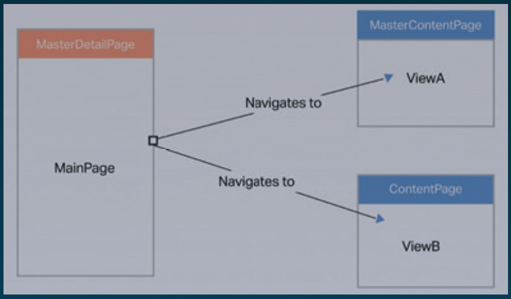
New Release

Infragistics Ultimate UI Controls for Xamarin

Lightning-fast controls with a RAD WYSIWYG design-time experience

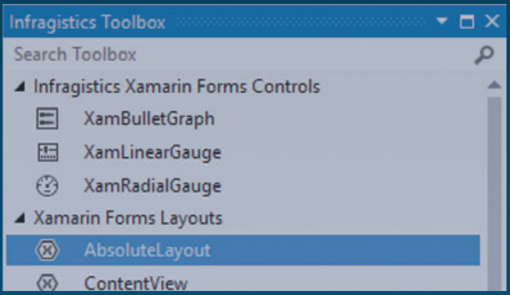
AppMap

Visually map out the entire flow of your app, including master-detail, tab, and child pages. Automatically generate all the Views, ViewModels and Navigation code with the click of a button! Generate a best-practices Prism architecture in a clean MVVM solution with iOS and Android projects. Hit F5 and your app is ready to run!



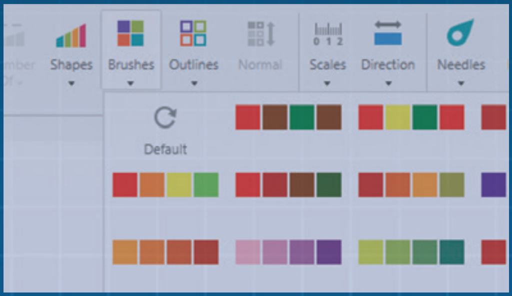
Xamarin.Forms Toolbox

Use the world's first NuGet-powered Toolbox to design your app views by dragging & dropping widgets directly from the Toolbox onto the XAML editor – no previous XAML knowledge required. The Toolbox gives you automatic namespace referencing, full support for all Infragistics Ultimate UI for Xamarin widgets, and the complete set of Xamarin.Forms widgets!



Control Configurators

Visually configure rich controls like data grids and charts right in the XAML editor and shave hours off your development time. Customize your UI widgets in a rich, design-time experience with pre-built styles and themes, built-in databinding, and WYSIWYG property editing – no coding required!



Get the Code! Download the Reference App

Infragistics.com/Xamarin

msdn magazine

APRIL 2017 VOLUME 32 NUMBER 4

General Manager Jeff Sandquist
Director Dan Fernandez
Editorial Director Mohammad Al-Sabt *mmeditor@microsoft.com*
Site Manager Kent Sharkey
Editorial Director, Enterprise Computing Group Scott Bekker
Editor in Chief Michael Desmond
Features Editor Sharon Terdeman
Features Editor Ed Zintel
Group Managing Editor Wendy Hernandez
Senior Contributing Editor Dr. James McCaffrey
Contributing Editors Dino Esposito, Frank La Vigne, Julie Lerman, Mark Michaelis, Ted Neward, David S. Platt
Vice President, Art and Brand Design Scott Shultz
Art Director Joshua Gould

1105 ENTERPRISE COMPUTING GROUP

President
Henry Allain

Chief Revenue Officer
Dan LaBianca

Chief Marketing Officer
Carmel McDonagh

ART STAFF
Creative Director **Jeffrey Langkau**
Associate Creative Director **Scott Rovin**
Senior Art Director **Deirdre Hoffman**
Art Director **Michele Singh**
Assistant Art Director **Dragutin Cvijanovic**
Senior Graphic Designer **Alan Tao**
Senior Web Designer **Martin Peace**

PRODUCTION STAFF
Print Production Coordinator **Lee Alexander**

ADVERTISING AND SALES
Chief Revenue Officer **Dan LaBianca**
Regional Sales Manager **Christopher Kourtoglou**
Account Executive **Caroline Stover**
Advertising Sales Associate **Tanya Egenolf**

ONLINE/DIGITAL MEDIA
Vice President, Digital Strategy **Becky Nagel**
Senior Site Producer, News **Kurt Mackie**
Senior Site Producer **Gladys Rama**
Site Producer **Chris Paoli**
Site Producer, News **David Ramel**
Director, Site Administration **Shane Lee**
Site Administrator **Biswarup Bhattacharjee**
Front-End Developer **Anya Smolinski**
Junior Front-End Developer **Casey Rysavy**
Executive Producer, New Media **Michael Domingo**
Office Manager & Site Assoc. **James Bowling**

LEAD SERVICES
Vice President, Lead Services **Michele Imgrund**
Senior Director, Audience Development & Data Procurement **Annette Levee**
Director, Audience Development & Lead Generation Marketing **Irene Fincher**
Director, Client Services & Webinar Production **Tracy Cook**
Director, Lead Generation Marketing **Eric Yoshizuru**
Director, Custom Assets & Client Services **Mallory Bastionell**
Senior Program Manager, Client Services & Webinar Production **Chris Flack**
Project Manager, Lead Generation Marketing **Mahal Ramos**

MARKETING
Chief Marketing Officer **Carmel McDonagh**
Vice President, Marketing **Emily Jacobs**
Marketing & Editorial Assistant **Megan Burpo**

ENTERPRISE COMPUTING GROUP EVENTS
Vice President, Events **Brent Sutton**
Senior Director, Operations **Sara Ross**
Senior Director, Event Marketing **Merikay Marzoni**
Events Sponsorship Sales **Danna Vedder**
Senior Manager, Events **Danielle Potts**
Coordinator, Event Marketing **Michelle Cheng**
Coordinator, Event Marketing **Chantelle Wallace**

1105 MEDIA INC
YOUR GROWTH. OUR BUSINESS.

Chief Executive Officer
Rajeev Kapur

Chief Operating Officer
Henry Allain

Chief Financial Officer
Craig Rucker

Chief Technology Officer
Erik A. Lindgren

Executive Vice President
Michael J. Valenti

Chairman of the Board
Jeffrey S. Klein

ID STATEMENT MSDN Magazine (ISSN 1528-4859) is published 13 times a year, monthly with a special issue in November by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. $35.00, International $60.00. Annual digital subscription rates payable in U.S. funds are: U.S. $25.00, International $25.00. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: MSDN Magazine, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. POSTMASTER: Send address changes to MSDN Magazine, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o *MSDN Magazine*, 4 Venture, Suite 150, Irvine, CA 92618.

LEGAL DISCLAIMER The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

CORPORATE ADDRESS 1105 Media, 9201 Oakdale Ave. Ste 101, Chatsworth, CA 91311 www.1105media.com

MEDIA KITS Direct your Media Kit requests to Chief Revenue Officer Dan LaBianca, 972-687-6702 (phone), 972-687-6799 (fax), dlabianca@1105media.com

REPRINTS For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International Phone: 212-221-9595. E-mail: 1105reprints@parsintl.com. www.magreprints.com/QuickQuote.asp

LIST RENTAL This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Jane Long, Merit Direct. Phone: 913-685-1301; E-mail: jlong@meritdirect.com; Web: www.meritdirect.com/1105

Reaching the Staff
Staff may be reached via e-mail, telephone, fax, or mail. A list of editors and contact information is also available online at Redmondmag.com.
E-mail: To e-mail any member of the staff, please use the following form: FirstinitialLastname@1105media.com
Irvine Office (weekdays, 9:00 a.m. – 5:00 p.m. PT)
Telephone 949-265-1520; Fax 949-265-1528
4 Venture, Suite 150, Irvine, CA 92618
Corporate Office (weekdays, 8:30 a.m. – 5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
9201 Oakdale Avenue, Suite 101, Chatsworth, CA 91311
The opinions expressed within the articles and other contentsherein do not necessarily express those of the publisher.

Microsoft

BPA WORLDWIDE

1906 100 2006 AMERICAN BUSINESS MEDIA A Century Moving Business Forward

WPA Western Publications Association

LEADTOOLS®

THE WORLD LEADER IN IMAGING SDKs

The leading **DICOM & PACS SDK**

Comprehensive DICOM & PACS SDK technology specially designed for modern, interconnected workplaces. Quickly develop powerful, multi-platform applications with support for DICOM data set, metadata, secure PACS communication, and medical-related image processing for 2D and 3D images.

SDKs for WEB

SDKs for MOBILE

SDKs for DESKTOP

SDKs for SERVER

.NET WinRT Windows API Java Linux iOS macOS Android JavaScript

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM

LEAD TECHNOLOGIES INCORPORATED

SALES@LEADTOOLS.COM
800.637.1840

The Things We Leave Behind

In its short history, the Internet of Things (IoT) has produced its share of drama, from the largest-ever distributed denial of service (DDoS) attack (msdn.com/magazine/mt790193) to the prospect of zombie automobiles taking orders from remote hackers (msdn.com/magazine/mt422336). Now, a recent presentation at the RSA Conference 2017 makes clear that the IoT has an ownership lifecycle problem. And once again, the auto industry is helping lead the way.

Charles Henderson leads IBM's rather amazingly named X-Force Red, a crack team of security pros tasked with challenging and verifying the security of deployed applications, networks, hardware and workforces. In that role, Henderson is involved in research, outreach and vulnerability testing for IBM. At the RSA Conference 2017 in San Francisco, Henderson related his personal experience trading in a beloved convertible for a new car at an auto dealership. What started as a simple transaction turned into a journey of discovery, as Henderson learned that there seemed to be no straightforward way to fully remove his personal information and access rights from the connected systems on his old car.

Two years later, the old convertible was *still* present in his smartphone app. Henderson enjoyed as much control over his old car's systems as the current, rightful owner.

It wasn't for lack of trying. Henderson detailed the steps he took in returning his convertible, taking care to clear his personal information from the car's infotainment and other systems. He performed a factory reset, wiped the Bluetooth settings and reset the garage door openers. But when he got home with the new car—a different model from the same brand and dealership of his old car—Henderson discovered something odd. The smartphone app used to locate his car and provide convenience functions like locking doors, starting the engine and beeping the horn still showed his old car right next to the new one. Figuring there must be a lag in processing the transfer, Henderson waited. And waited. And waited.

Two years later, the old convertible was *still* present in his smartphone app. Henderson enjoyed as much control over his old car's systems as the current, rightful owner. Over the next two years that followed, he researched the resale of several cars back to authorized dealers across four different manufacturers, and found that in every instance the dealer failed to properly control access after the sale.

"Cars are not disposable items," Henderson said. "Concepts of access revocation and resetting access only work if they're intuitive to that second owner."

The problem is only getting worse. As home smart hubs and other consumer-connected devices and appliances proliferate, the question begs: What happens to data, settings and access after you're done with connected hardware? Right now, there's no easy and obvious answer. Henderson pointed out that the mobile phone industry used to have this problem, with personally identifiable information (PII) like photos and contacts turning up on phones resold on the open market. To help boost the resale value of phones, vendors worked to create a consistent device reset experience.

Automobiles and really the whole universe of consumer IoT products need something similar. Henderson said the answer lies in the development and adoption of standards for clearing PII from devices, and in training users to look for and utilize factory reset functionality. And even then, it won't be easy.

"At the B2B level we still screw up access revocation on a daily basis," Henderson told the audience. "If we can't do it in business, how do we expect to do it in the home? That's a tough nut to crack."

It certainly is. But the alternative is to leave a trail of leaky, exposed and vulnerable connected hardware in our wakes.

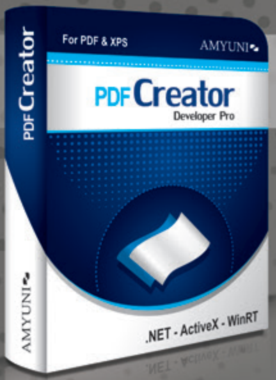
[illegible]

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

Switch to Amyuni PDF

AMYUNI

NEW v5.5



Create and Edit PDFs in .NET, COM/ActiveX, WinRT & UWP

- Edit, process and print PDF 1.7 documents
- Create, fill-out and annotate PDF forms
- Fast and lightweight 32- and 64-bit components for .NET and ActiveX/COM
- New Universal Apps DLLs enable publishing C#, C++, CX or Javascript apps to windows Store
- Updated Postscript/EPS to PDF conversion module

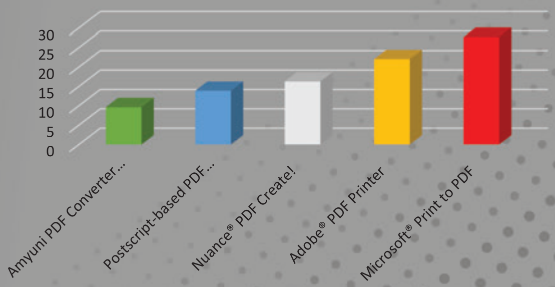
High Performance PDF Printer for Desktops and Servers

- Print to PDF in a fraction of the time needed with other tools. WHQL tested for all Windows platforms. Version 5.5 updated for Windows 10 support

Complete Suite of Accurate PDF Components

- All your PDF processing, conversion and editing in a single package
- Combines Amyuni PDF Converter and PDF Creator for easy licensing, integration and deployment.
- Includes our Microsoft WHQL certified PDF Converter printer driver
- Export PDF documents into other formats such as Jpeg, PNG, XAML or HTML5
- Import and Export XPS files using any programming environment

Benchmark Testing - Amyuni vs Others
Seconds required to convert a document to PDF



Other Developer Components from Amyuni®

- WebkitPDF: Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- PDF2HTML5: Conversion of PDF to HTML5 including dynamic forms
- Postscript to PDF Library: For document workflow applications that require processing of Postscript documents
- OCR Module: Free add-on to PDF Creator uses the Tesseract engine for character recognition
- Javascript engine: Integrate a full Javascript interpreter into your applications to process PDF files or for any other need

CERTIFIED FOR Windows Server® 2008

Windows Server 2012 Certified

Windows 7

Windows

All development tools available at
www.amyuni.com

AMYUNI Technologies

USA and Canada
Toll Free: 1866 926 9864
Support: 514 868 9227
sales@amyuni.com

Europe
UK: 0800-015-4682
Germany: 0800-183-0923
France: 0800-911-248

A Plan for Promotion

A few months ago, I started helping a software engineer, Suman, with his career. Suman had been a developer at his company for four years without a single promotion. Last week, he gave me an update: He'd just been promoted to senior developer, and earned a $23,000 raise.

The process of getting a promotion as a developer is simple, repeatable and universal. This is the process I asked Suman to follow (and that I've followed myself).

First, set a goal in the present tense, with a date attached, and start reviewing the goal daily. As an example, the goal could be, "As of June 2018, I am a senior software engineer at my current company." Constant review will sink the goal deep into your unconscious and impel you into action. Then, on a regular basis, take the *tiniest* uncomfortable step that would move you one step closer to this goal. This is because a promotion isn't a one-time event; it's a months-long series of small challenges and opportunities.

Make an Impression, Naturally

Next, when you volunteer for projects, be sure it helps or involves others on the team! This automatically publicizes your work and creates a quiet army of supporters. The projects don't have to be "sexy," but they should be high-impact, pressing needs for the team. The most boring, painful, "grunt work" projects are often the ones that *everybody* appreciates most when *someone* takes them on. But this is no time to "work on weaknesses"; pick projects in your comfort zone.

Another developer I know, Jaya, took this approach. As other developers gravitated to hot, new customer-facing features in the product, Jaya volunteered to work on the setup and configuration components. The goal: Demonstrate her skill as a developer, while showing that she wasn't high-maintenance or picky.

Jaya's success convinced her managers that she could do any job well, even the most terrible ones. Increasingly important projects followed, and Jaya received a promotion at her next performance review.

It's a career mistake to resent being assigned "meaningless" projects. When you do these projects *as if* they were meaningful, the rewards are more outsized. Even mediocre engineers can give their best to "great" projects; only great engineers can give their best to mediocre projects.

When choosing projects, consider the following:

- Pick projects based on the pressing business needs and revenue focus areas of the company. Avoid choosing based purely on technical beauty or architectural complexity. Budgets, not beauty, pay the bills.
- Pick team-based projects, ideally involving other teams' engineers, rather than projects that are done completely alone, behind-the-scenes.

Once you identify three to four projects, present your plan to your manager. Say something like, "I want to have a bigger impact on the team. So, getting promoted by next year is important to me. But before I do, I already want to be operating at that level. I think these projects will help me get there." Then invite the manager to provide input on the project ideas.

Even mediocre engineers can give their best to "great" projects; only great engineers can give their best to mediocre projects.

Usually, managers are so blown away by this proactive approach that they agree to the project plan. And once they consent to the plan, they'll feel more invested in helping you get promoted!

Uncover Strengths, Strategically

So far, you've taken steps toward raising your contributions and brought your manager into the effort. To maintain momentum, ongoing communication is key. Once a month, check in with your manager on how she thinks you're doing in your progress toward the promotion. Ask her, "What's one thing I've done well over the last month? What's one thing I could improve?" This forces the manager to articulate clearly (to herself) what you're doing well, instead of waiting until review time. And it helps you identify your own blind spots before it's too late.

In addition to brief monthly conversations, keep a record of your achievements every week, regardless of whether you share them with your manager. This isn't so much for your manager's benefit as it is for your own advancement. Each weekly summary is a concise, back-door resume that wraps up your achievements for the week. This exercise is an effective self-accountability mechanism, and serves as an important marketing tool at review time.

Finally, seek out an advisor from outside the company who can suggest more creative and effective ways to achieve the promotion. These people can also offer a broader, outsider perspective and help you avoid a bad case of tunnel vision.

Taken together, these efforts can put your promotion efforts in fast-forward and bend your career arc toward success. ■

KRISHNAN RANGACHARI *is an executive coach for software engineers. Visit RadicalShifts.com for his career success secrets.*

DevExpress®

DevExpress Spreadsheet for WPF & WinForms

with Chart, Pivot Table and Cell Editor support

Your users already know Microsoft® Excel® and with the new capabilities we've introduced in our Spreadsheet Control, you can create solutions that utilize a "spreadsheet-first" UX for a broad range of use-case scenarios. Imagine sales entry forms that are an actual invoice or an inventory management document with built-in ordering and data entry options.

Give our Spreadsheet a try today and see how easy it is to build Windows® apps that amaze.

Free 30-day trial

devexpress.com/spreadsheet

#UseTheBest

Query JSON Data in SQL Server 2016

Moving data around independent and autonomous systems is all that most software does these days and JSON is the ubiquitous language behind data transfer. Short for JavaScript Object Notation, JSON is a text-based way to lay out the state of an object so that it can be easily serialized and transferred across the wire from one system to the next, especially in heterogeneous systems.

JSON has become what, in the end, XML failed to be—the *lingua franca* of the Web. Personally, I wouldn't buy into much of the fact that JSON is easier to read than XML. On the other hand, JSON is a text format much more compact and lightweight than XML, editable by humans and quick to parse and understand for computers across a long list of software and hardware platforms.

A JSON string is a plain text string and any versions of any relational database management system (RDBMS), including SQL Server, let you store a string regardless of its content layout. SQL Server 2016, however, is the first version of the Microsoft database that lets you read existing tabular data as JSON, to save tabular data as JSON and, more important, to query within JSON strings as if the JSON content were actually a collection of individual columns.

For a structured and comprehensive overview of the JSON functions in SQL Server 2016, read the MSDN documentation at bit.ly/2llab1n. In addition, you can find an excellent executive summary of JSON in SQL Server 2016 in the Simple Talk article at bit.ly/26rprwv. The article offers a more business-oriented view of JSON in SQL Server 2016 and, in general, a scenario-based perspective of the use of JSON data in a relational persistence layer.

JSON Data in the Persistence Layer

Two verbs are key to understanding the purpose of JSON: transmit and serialize. Therefore, JSON is the format in which you lay out the state of a software entity so that it can be transmitted across process spaces with the certainty it'll be well understood on both ends. Great, but this is a column about JSON in SQL Server and, hence, in the persistence layer. So, let's start with the base question: When would you save data in SQL Server as JSON?

A relational database table is articulated on a fixed number of columns and each column has its own data type, such as strings of variable or fixed length, dates, numbers, Booleans and the like. JSON is not a native data type. A SQL Server column that contains JSON data from the database perspective is a plain string column. You can write JSON data to a table column as you would write a regular string and you can do that in any versions of SQL Server, as well as in any other RDBMS.

Where do you get the JSON strings you eventually store into a database? There are two main scenarios: First, those strings might come from a Web service or some other form of an external endpoint that transmits data (for example, a connected device or sensor). Second, JSON data might be a convenient way to group together related pieces of information so that they appear as a single data item. This typically happens when you deal with semi-structured data, such as data that represents a business event to store in an event-sourcing scenario or, more simply, in a business context that's inherently event-driven, such as real-time systems for domains such as finance, trading, scoring, monitoring, industrial automation and control, and so on. In all these cases, your storage can be normalized to a structured form serializing related information variable in length and format in a single data item that would fit in the string column of a relational table.

Two verbs are key to understanding the purpose of JSON: transmit and serialize.

As mentioned, the JSON content you might persist can come from an external source or can be generated through serialization from instances of C# objects:

```
foreach (var c in countries)
{
  // Serialize the C# object to JSON
  var json = JsonConvert.SerializeObject(c);

  // Save content to the database
  record.JsonColumn = json;
}
```

You can use Entity Framework (EF), as well, to save JSON data into one column of a database table.

SQL Server 2016 takes this one level further and lets you transform JSON data in table rows. This ability might save a lot of work and CPU cycles off your code as now you can push the raw JSON text to the database without first parsing it to C# objects in the application code and then passing through EF or direct ADO.NET calls. The key to achieve this goal is the new OPENJSON function:

```
declare @country nvarchar(max) = '{
  "id" : 101,
  "name": "United States",
  "continent": "North America"
}';
  INSERT INTO Countries
    SELECT * FROM OPENJSON(@country)
    WITH (id int,
      name nvarchar(100),
      continent nvarchar(100))
```

You can use the function to insert or update regular table rows from plain JSON text. The WITH clause lets you map JSON properties to existing table columns.

The Event Sourcing Scenario

In my December 2016 column, I discussed Event Sourcing as an emerging pattern to store the historical state of the application (msdn.com/magazine/mt790196). Instead of saving the latest-known good state, with Event Sourcing you save every single business event that alters the state and rebuild the latest state replaying the past events.

With JSON, features enabled using SQL Server in an Event Sourcing scenario become realistic.

The crucial aspect of an Event Sourcing implementation is how effectively you can save and retrieve the past events. Every event is different and might have a different schema, depending on the type and information available. At the same time, having a distinct (relational) store for each event type is problematic because events come asynchronously and might affect different entities and different segments of the state. If you keep them in different tables, rebuilding the state might become expensive because of cross-table JOINs. Hence, saving events as objects is the most recommended option and NoSQL stores do the work very well. Is it possible to do Event Sourcing with a relational database instead?

Saving the event as JSON is an option possible on any version of SQL Server, but reading JSON effectively, when large numbers of events are in store, might be unsustainable. With the native JSON features in SQL Server 2016, the landscape changes and using SQL Server in an Event Sourcing scenario becomes realistic. However, how would you query JSON from a database table?

Querying Data Out of JSON Content

So let's say you managed to have one or more columns of JSON data in a canonical relational table. Therefore, columns with primitive data and columns filled with JSON data live side by side. Unless the new functions of SQL Server 2016 are used, the JSON columns are treated as plain text fields and can be queried only with T-SQL string and text instructions such as LIKE, SUBSTRING and TRIM. For the purpose of the demo, I built a column called Countries—with a few tabular columns—and another named Serialized that contains the entire rest of the record serialized as JSON, as shown in **Figure 1**.

The JSON object serialized in the sample table looks like this:

```
{
  "CountryCode":"AD",
  "CountryName":"Andorra",
  "CurrencyCode":"EUR",
  "Population":"84000",
  "Capital":"Andorra la Vella",
  "ContinentName":"Europe",
  "Continent":"EU",
  "AreaInSqKm":"468.0",
  "Languages":"ca",
  "GeonameId":"3041565",
  "Cargo":null
```

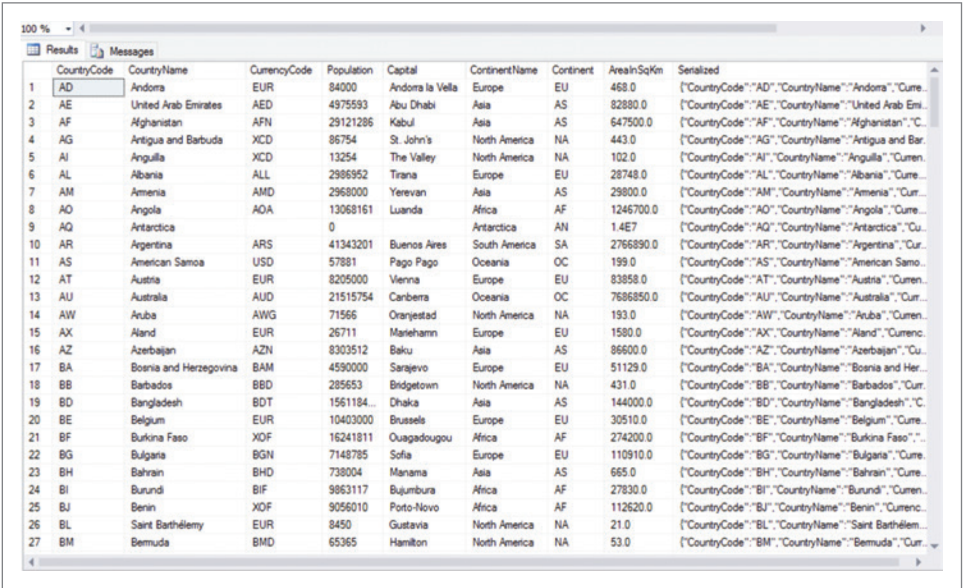
The following T-SQL query shows how to select only the countries that count more than 100 million inhabitants. The query mixes regular table columns and JSON properties:

```
SELECT CountryCode,
  CountryName,
  JSON_VALUE(Serialized, '$.Population') AS People
FROM Countries
WHERE ISJSON(Serialized) > 0 AND
  JSON_VALUE(Serialized, '$.Population') > 100000000
ORDER BY JSON_VALUE(Serialized, '$.AreaInSqKm')
```

The JSON_VALUE function takes the name of a JSON column (or a local variable set to a JSON string) and extracts the scalar value following the specified path. As shown in **Figure 2**, the $ symbol refers to the root of the serialized JSON object.

Because the JSON column is configured as a plain NVARCHAR column, you might want to use the ISJSON function to check whether the content of the column is real JSON. The function returns a positive value if the content is JSON.

JSON_VALUE always returns a string of up to 4,000 bytes, regardless of the selected property. If you expect a longer return value, then you should use OPENJSON instead. At any rate, you might want to consider a CAST to get a value of the proper type. Looking back at the previous example, let's say you want the number of people living in a country formatted with commas. (In general, this might not be a good idea because formatting data in the presentation layer gives your code a lot more flexibility.) The



	CountryCode	CountryName	CurrencyCode	Population	Capital	ContinentName	Continent	AreaInSqKm	Serialized
1	AD	Andorra	EUR	84000	Andorra la Vella	Europe	EU	468.0	{"CountryCode":"AD","CountryName":"Andorra","Curre...
2	AE	United Arab Emirates	AED	4975593	Abu Dhabi	Asia	AS	82880.0	{"CountryCode":"AE","CountryName":"United Arab Emi...
3	AF	Afghanistan	AFN	29121286	Kabul	Asia	AS	647500.0	{"CountryCode":"AF","CountryName":"Afghanistan","C...
4	AG	Antigua and Barbuda	XCD	86754	St. John's	North America	NA	443.0	{"CountryCode":"AG","CountryName":"Antigua and Bar...
5	AI	Anguilla	XCD	13254	The Valley	North America	NA	102.0	{"CountryCode":"AI","CountryName":"Anguilla","Curren...
6	AL	Albania	ALL	2986952	Tirana	Europe	EU	28748.0	{"CountryCode":"AL","CountryName":"Albania","Curre...
7	AM	Armenia	AMD	2968000	Yerevan	Asia	AS	29800.0	{"CountryCode":"AM","CountryName":"Armenia","Curr...
8	AO	Angola	AOA	13068161	Luanda	Africa	AF	1246700.0	{"CountryCode":"AO","CountryName":"Angola","Curre...
9	AQ	Antarctica		0		Antarctica	AN	1.4E7	{"CountryCode":"AQ","CountryName":"Antarctica","Cu...
10	AR	Argentina	ARS	41343201	Buenos Aires	South America	SA	2766890.0	{"CountryCode":"AR","CountryName":"Argentina","Cur...
11	AS	American Samoa	USD	57881	Pago Pago	Oceania	OC	199.0	{"CountryCode":"AS","CountryName":"American Samo...
12	AT	Austria	EUR	8205000	Vienna	Europe	EU	83858.0	{"CountryCode":"AT","CountryName":"Austria","Curren...
13	AU	Australia	AUD	21515754	Canberra	Oceania	OC	7686850.0	{"CountryCode":"AU","CountryName":"Australia","Curr...
14	AW	Aruba	AWG	71566	Oranjestad	North America	NA	193.0	{"CountryCode":"AW","CountryName":"Aruba","Curren...
15	AX	Aland	EUR	26711	Mariehamn	Europe	EU	1580.0	{"CountryCode":"AX","CountryName":"Aland","Currenc...
16	AZ	Azerbaijan	AZN	8303512	Baku	Asia	AS	86600.0	{"CountryCode":"AZ","CountryName":"Azerbaijan","Cu...
17	BA	Bosnia and Herzegovina	BAM	4590000	Sarajevo	Europe	EU	51129.0	{"CountryCode":"BA","CountryName":"Bosnia and Her...
18	BB	Barbados	BBD	285653	Bridgetown	North America	NA	431.0	{"CountryCode":"BB","CountryName":"Barbados","Curr...
19	BD	Bangladesh	BDT	1561184...	Dhaka	Asia	AS	144000.0	{"CountryCode":"BD","CountryName":"Bangladesh","C...
20	BE	Belgium	EUR	10403000	Brussels	Europe	EU	30510.0	{"CountryCode":"BE","CountryName":"Belgium","Curre...
21	BF	Burkina Faso	XOF	16241811	Ouagadougou	Africa	AF	274200.0	{"CountryCode":"BF","CountryName":"Burkina Faso","...
22	BG	Bulgaria	BGN	7148785	Sofia	Europe	EU	110910.0	{"CountryCode":"BG","CountryName":"Bulgaria","Curre...
23	BH	Bahrain	BHD	738004	Manama	Asia	AS	665.0	{"CountryCode":"BH","CountryName":"Bahrain","Curre...
24	BI	Burundi	BIF	9863117	Bujumbura	Africa	AF	27830.0	{"CountryCode":"BI","CountryName":"Burundi","Curren...
25	BJ	Benin	XOF	9056010	Porto-Novo	Africa	AF	112620.0	{"CountryCode":"BJ","CountryName":"Benin","Currenc...
26	BL	Saint Barthélemy	EUR	8450	Gustavia	North America	NA	21.0	{"CountryCode":"BL","CountryName":"Saint Barthélem...
27	BM	Bermuda	BMD	65365	Hamilton	North America	NA	53.0	{"CountryCode":"BM","CountryName":"Bermuda","Curr...

Figure 1 **The Sample Countries Database with a JSON Column**

SQL FORMAT function expects to receive a number and you receive an error if you pass the direct JSON value. To make it work, you must resort to an explicit CAST:

```
SELECT CountryCode,
  CountryName,
  FORMAT(CAST(
    JSON_VALUE(Serialized, '$.Population') AS int), 'NO')
    AS People
FROM Countries
WHERE ISJSON(Serialized) > 0 AND
  JSON_VALUE(Serialized,'$.Population') > 100000000
ORDER BY JSON_VALUE(Serialized, '$.AreaInSqKm')
```

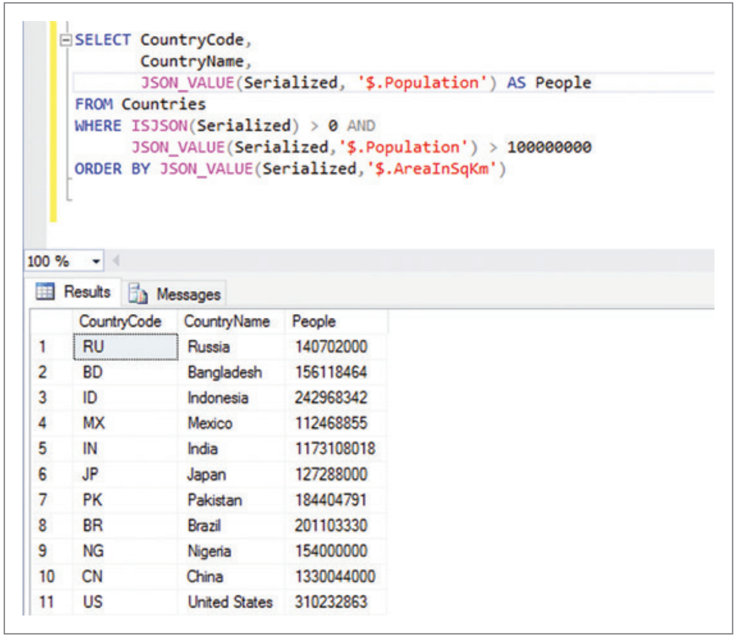
The JSON_VALUE can only return a single scalar value. If you have an array of a nested object that you want to extract, then you must resort to the JSON_QUERY function.

How effective is it to query over JSON data? Let's do some tests.

JSON parsing is faster than the deserialization of some special types, such as XML and spatial.

Indexing JSON Content in SQL Server 2016

As obvious as it might sound, querying the entire JSON string from the database and then parsing it in memory through a dedicated library such as Newtonsoft JSON, albeit always functional, might not be an effective approach in all cases. Effectiveness mostly depends on the number of records in the database and how long it might really take to get the data you need in the format you need. Probably for a query that your application runs occasionally, in-memory processing of JSON data might still be an option. In general, though, querying through JSON-dedicated functions and letting SQL Server do the parsing internally results in slightly faster code. The difference is even bigger if you add an index on JSON data.



```
SELECT CountryCode,
       CountryName,
       JSON_VALUE(Serialized, '$.Population') AS People
FROM Countries
WHERE ISJSON(Serialized) > 0 AND
      JSON_VALUE(Serialized,'$.Population') > 100000000
ORDER BY JSON_VALUE(Serialized,'$.AreaInSqKm')
```

	CountryCode	CountryName	People
1	RU	Russia	140702000
2	BD	Bangladesh	156118464
3	ID	Indonesia	242968342
4	MX	Mexico	112468855
5	IN	India	1173108018
6	JP	Japan	127288000
7	PK	Pakistan	184404791
8	BR	Brazil	201103330
9	NG	Nigeria	154000000
10	CN	China	1330044000
11	US	United States	310232863

Figure 2 **Results of a JSON Query**

You shouldn't create the index on the JSON column, however, as it would index the JSON value as a single string. You'll hardly be querying for the entire JSON string or a subset of it. More realistically, instead, you'll be querying for the value of a particular property in the serialized JSON object. A more effective approach is creating one or more computed columns based on the value of one or more JSON properties and then indexing those columns. Here's an example in T-SQL:

```
-- Add a computed column
ALTER TABLE dbo.Countries
ADD JsonPopulation
AS JSON_VALUE(Serialized, '$.Population')

-- Create an index
CREATE INDEX IX_Countries_JsonPopulation
ON dbo.Countries(JsonPopulation)
```

Again, you should be aware that JSON_VALUE returns NVARCHAR, so unless you add CAST the index will be created on text.

Interestingly, JSON parsing is faster than the deserialization of some special types, such as XML and spatial. You can find more information at bit.ly/2kthrrC. In summary, at least JSON parsing is better than fetching properties of other types.

JSON and EF

As a general remark, the JSON support in SQL Server 2016 is primarily exposed through the T-SQL syntax, as tooling is quite limited now. In particular, EF doesn't currently provide any facilities to query JSON data, except for the SqlQuery method in EF6 and FromSql in EF Core. However, this doesn't mean you can't serialize complex properties of C# classes (say, arrays) into JSON columns. An excellent tutorial for EF Core can be found at bit.ly/2kVEsam.

Wrapping Up

SQL Server 2016 introduces some native JSON capabilities so that you can more effectively query stored JSON data as a canonical rowset. This mostly happens when the JSON data is the serialized version of some semi-structured aggregate of data. Indexes built out of computed columns that reflect that value of one or more JSON properties definitely help improve the performance.

JSON data is stored as plain text and isn't considered a special type, such as XML and Spatial. However, this just enables you to use JSON columns in any SQL Server objects right away. The same can't be said for other complex types such as XML, CLR and Spatial that are still on the waiting list.

In this column, I focused on the JSON-to-rowset scenario. However, SQL Server 2016 also fully supports the rowset-to-JSON query scenario when you write a regular T-SQL query and then map results to JSON objects via the FOR JSON clause. For more information on this feature, see bit.ly/2fTKly7. ■

Dino Esposito *is the author of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2014) and "Modern Web Applications with ASP.NET" (Microsoft Press, 2016). A technical evangelist for the .NET and Android platforms at JetBrains, and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents@wordpress.com and on Twitter: @despos.*

Thanks to the following Microsoft technical expert for reviewing this article: *Jovan Popovic*

File Format APIs

Working with Files?

CREATE CONVERT PRINT
MODIFY COMBINE

FREE TRIAL

Aspose.Total

Manipulate Word, Excel, PDF, PowerPoint, Outlook and more than 100 other file formats in your applications without installing Microsoft Office.

DOC, XLS, PDF, PPT, MSG, BMP, PNG, XML and many more!

Platforms supported: .NET, Java, Cloud, Android, SharePoint, Reporting Services, and JasperReports

ASPOSE
File Format APIs

CONTACT US

US: +1 903 306 1676
EU: +44 141 628 8900
AU: +61 2 8006 6987

sales@asposeptyltd.com

Try for FREE at
www.aspose.com

JULIE LERMAN

Tips for Building Tests with EF Core and Its InMemory Provider

When creating automated tests against methods that trigger database interaction, there are times when you truly want to see what's happening in the database, and other times when the database interaction isn't at all relevant to your test's assertion. The new EF Core InMemory provider can prove useful in the latter case. In this article I'll provide an introduction to this handy tool and share some tips and tricks about creating automated tests with EF Core that I've discovered while learning to use it myself.

In cases where the database effort isn't important to the test result, unnecessary calls to the database can put a strain on performance or even cause inaccurate test results. For example, the amount of time it takes to talk to the database—or drop and recreate a test database—can hold up your tests. Another concern is if there's a problem with the database itself. Perhaps network latency or a momentary hiccup causes a test to fail only because the database is unavailable, not as a result of a failure in the logic the test is attempting to assert.

> Microsoft has created another provider—not to persist to a database, but to temporarily persist to memory.

We've long sought ways to minimize these side effects. Fakes and mocking frameworks are common solutions. These patterns allow you to create in-memory representations of the data store, but there's a lot involved in setting up their in-memory data and behavior. Another approach is to use a lighter-weight database for testing than you're targeting in production, such as a PostgreSQL or SQLite database, rather than, for example, a SQL Server database you use for your production data store. Entity Framework (EF) has always allowed for targeting different databases with a single model thanks to the various providers available. However, nuanced differences in database functionality can cause you to hit issues where this won't always work (though it's still a good option to keep in your toolbox). Alternatively, you could use an external tool, such as the open source EFFORT extension (github.com/tamasflamich/effort), which magically provides an in-memory representation of the data store without the setup needed for fakes or mocking frameworks. EFFORT works with EF 4.1 through EF6, but not EF Core.

Code download available at msdn.com/magazine/0417magcode.

There are already a number of database providers for EF Core. Microsoft includes the SQL Server and SQLite providers as part of the family of EntityFrameworkCore APIs. There are also providers for SQLCE and PostgreSQL, respectively maintained by MVPs Erik Eilskov Jensen and Shay Rojansky. And there are third-party commercially available providers. But Microsoft has created another provider—not to persist to a database, but to temporarily persist to memory. This is the InMemory provider: Microsoft. EntityFrameworkCore.InMemory, which you can use as a quick way to provide a stand-in for an actual database in many testing scenarios.

Readying the DbContext for the InMemory Provider

Because your DbContext will sometimes be used to connect to a true data store and sometimes to the InMemory provider, you want to set it up to be flexible with respect to providers, rather than dependent on any particular provider.

When instantiating a DbContext in EF Core, you have to include DbContextOptions that specify which provider to use and, if needed, a connection string. UseSqlServer and UseSqlite, for example, require that you pass in a connection string, and each provider gives you access to the relevant extension method. Here's how this looks if done directly in the DbContext class OnConfiguring method, where I'm reading a connection string from an application config file:

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {
  var settings = ConfigurationManager.ConnectionStrings;
  var connectionString = settings["productionDb"].ConnectionString;
  optionsBuilder.UseSqlServer(connectionString);
 }
```

A more flexible pattern, however, is to pass a pre-configured DbContextOptions object into the constructor of the DbContext:

```
public SamuraiContext(DbContextOptions<SamuraiContext> options)
    :base(options) { }
```

EF Core will pass those pre-configured options into the underlying DbContext and apply them for you.

With this constructor in place, you now have a way to specify different providers (and other options, such as a connection string) on the fly from the logic that's using the context.

If you're using some type of Inversion of Control (IoC) container in your application, such as StructureMap (structuremap.github.io) or the services built into ASP.NET Core, you have the ability to

ComponentSource® /update/2017/04

Experience the brand new look at **www.componentsource.com**

Aspose.Total for .NET | from $2,939.02

ASPOSE File Format APIs

Every Aspose .NET API in one package.

- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Work with charts, diagrams, images, project plans, emails, barcodes, OCR and OneNote files alongside many more document management features in .NET applications
- Common uses also include mail merging, adding barcodes to documents, building dynamic reports on the fly and extracting text from most document types

DevExpress DXperience 16.2 | from $1,439.99

DevExpress

The complete range of DevExpress .NET controls and libraries for all major Microsoft platforms.

- WinForms - New TreeMap control, Chart series types and Unbound Data Source
- WPF - New Wizard control and Data Grid scrollbar annotations
- ASP.NET - New Vertical Grid control, additional Themes, Rich Editor Spell Checking and more
- Windows 10 Apps - New Hamburger Sub Menus, Splash Screen and Context Toolbar controls
- CodeRush - New debug visualizer expression map and code analysis diagnostics

ActiveReports 11 | from $1,575.02

GrapeCity

Award-winning .NET reporting platform for HTML5, WPF, WinForms, ASP.NET & Windows Azure.

- Visual Studio-integrated report designer
- Extensible optional report server with built-in scalability
- Responsive HTML5 Report Portal
- Royalty-free redistribution
- Source data from JSON files, Web services and REST API using the built in JSON data provider

Help & Manual Professional | from $586.04

ec software

Help and documentation for .NET and mobile applications.

- Powerful features in an easy, accessible and intuitive user interface
- As easy to use as a word processor, but with all the power of a true WYSIWYG XML editor
- Single source, multi-channel publishing with conditional and customized output features
- Output to responsive HTML, CHM, PDF, MS Word, ePUB, Kindle or print
- Styles and Templates give you full design control

We accept purchase orders.
Contact us to apply for a credit account.

US Headquarters	European Headquarters	Asia / Pacific Headquarters
ComponentSource	ComponentSource	ComponentSource
650 Claremore Prof Way	2 New Century Place	7F Kojimachi Ichihara Bldg
Suite 100	East Street	1-1-8 Hirakawa-cho
Woodstock	Reading, Berkshire	Chiyoda-ku
GA 30188-5188	RG1 4ET	Tokyo, 102-0093
USA	United Kingdom	Japan

Sales Hotline - US & Canada:
(888) 850-9911
www.componentsource.com

VISA DISCOVER
GSA Schedule Contract GS-35F-0188R

configure the provider for the context in the code where you configure other application-wide IoC services. Here's an example that uses ASP.NET Core services in a typical startup.cs file:

```
public void ConfigureServices(IServiceCollection services) {
  services.AddDbContext<SamuraiContext>(
    options => options.UseSqlServer(
      Configuration.GetConnectionString("productionDb")));
  services.AddMvc();
}
```

In this case, SamuraiContext is the name of my class that inherits from DbContext. I'm using SQL Server again and I've stored the connection string in the ASP.NET Core appsettings.json file under the name productionDb. The service has been configured to know that any time a class constructor requires an instance of SamuraiContext, the runtime should not only instantiate SamuraiContext, but should pass in the options with the provider and connection string stated in this method.

When this ASP.NET Core app uses my SamuraiContext, by default, it will now do so with SQL Server and my connection string. But thanks to the flexibility I built into the SamuraiContext class, I can also create tests that use the same SamuraiContext but pass in a DbContextOptions object that specifies using the InMemory provider instead—or specifies any other options that are relevant to a particular test.

Deleting and recreating the SQL Server database might affect how long it takes to run the test.

In the next section I'll show two different tests that involve EF Core. The first, **Figure 1**, is designed to test that the correct database interaction occurred. This means I truly want the test to hit the database, so I'll construct DbContextOptions to use the SQL Server provider, but with a connection string that targets a test version of my database I can create and drop on the fly.

I use the EnsureDeleted and EnsureCreated methods to give me a totally fresh version of the database for the test, and these will work even if you don't have migrations. Alternatively, you could use EnsureDeleted and Migrate to recreate the database if you have migration files.

Next, I create a new entity (samurai), tell EF to begin tracking it and then note the temporary key value the SQL Server provider supplies. After calling SaveChanges, I verify that SQL Server has applied its own database-generated value for the key, assuring me that this object was, indeed, inserted into the database correctly.

Deleting and recreating a SQL Server database might affect how long it takes to run the test. You could use SQLite in this case and get the same results more quickly while ensuring that the test is still hitting an actual database. Also, note that, like the SQL Server provider, SQLite also sets a temporary key value when you add an entity to the context.

If you have methods that happen to use EF Core but you want to test them without hitting the database, this is where the InMemory provider is so handy. But keep in mind that InMemory is not a database and won't emulate all flavors of relational database behavior—for example, referential integrity. When this is important to your test, you may prefer the SQLite option or, as the EF Core docs suggest, the SQLite in-memory mode as explained at bit.ly/2l7M71p.

Here's a method I've written in an app that performs a query with EF Core and returns a list of KeyValuePair objects:

```
public List<KeyValuePair<int, string>> GetSamuraiReferenceList() {
  var samurais = _context.Samurais.OrderBy(s => s.Name)
    .Select(s => new {s.Id, s.Name})
    .ToDictionary(t => t.Id, t => t.Name).ToList();
  return samurais;
}
```

I want to test that the method truly returns a KeyValuePair list. I don't need it to query the database in order to prove this.

Following is a test to do that using the InMemory provider (which I've already installed into the test project):

```
[TestMethod]
  public void CanRetrieveListOfSamuraiValues() {
    _options = new DbContextOptionsBuilder<SamuraiContext>()
             .UseInMemoryDatabase().Options;
    var context = new SamuraiContext(_options);
    var repo = new DisconnectedData(context);
    Assert.IsInstanceOfType(repo.GetSamuraiReferenceList(),
                        typeof(List<KeyValuePair<int, string>>));
  }
```

This test doesn't even require any sample data to be available to the in-memory representation of the database because it's enough to return an empty list of KeyValuePairs. When I run the test, EF Core will be sure that when the GetSamuraiReferenceList executes its query, the provider will allocate resources in memory for EF to execute against. The query is successful and so is the test.

What if I want to test that the correct number of results are returned? This means I'll need to provide data to seed the InMemory provider. Much like a fake or mock, this requires creating the data and loading it into the provider's data store. When using fakes and mocks, you might create a List object and populate that, then query against the list. The InMemory provider takes care of the container. You just use EF commands to pre-populate it. The InMemory provider also takes care of much of the overhead and extra coding that are needed when using fakes or mocks.

As an example, **Figure 2** shows a method I'm using to seed the InMemory provider before my tests interact with it:

If my in-memory data is empty, this method adds in two new samurais and then calls SaveChanges. Now it's ready to be used by a test.

But how would my InMemory data store have data in it if I've just instantiated the context? The context is not the InMemory data store. Think of the data store as a List object—the context will

Figure 1 **Testing That a Database Insert Works as Expected**

```
[TestMethod]
  public void CanInsertSamuraiIntoDatabase() {
    var optionsBuilder = new DbContextOptionsBuilder();
    optionsBuilder.UseSqlServer
      ("Server = (localdb)\\mssqllocaldb; Database =
        TestDb; Trusted_Connection = True; ");
    using (var context = new SamuraiContext(optionsBuilder.Options)) {
      context.Database.EnsureDeleted();
      context.Database.EnsureCreated();
      var samurai = new Samurai();
      context.Samurais.Add(samurai);
      var efDefaultId = samurai.Id;
      context.SaveChanges();
      Assert.AreNotEqual(efDefaultId, samurai.Id);
    }
  }
```

Figure 2 **Seeding an EF Core InMemory Provider**

```
private void SeedInMemoryStore() {
    using (var context = new SamuraiContext(_options)) {
      if (!context.Samurais.Any()) {
        context.Samurais.AddRange(
          new Samurai {
            Id = 1,
            Name = "Julie",
          },
          new Samurai {
            Id = 2,
            Name = "Giantpuppy",
        );
        context.SaveChanges();
      }
    }
  }
```

create it on the fly, if needed. But once it's been created, it remains in memory for the lifetime of the application. If I'm running a single test method, there will be no surprises. But if I'm running a number of test methods, then every test method will use the same set of data and you may not want to populate it a second time. There's more to understand about this, which I'll be able to explain after you see a bit more code.

This next test is a bit contrived, but it's designed to demonstrate using a populated InMemory store. Knowing that I've just seeded the memory with two samurais, the test calls that same GetSamuraiReferenceList method and asserts that there are two items in the resulting list:

```
[TestMethod]
  public void CanRetrieveAllSamuraiValuePairs() {
    var context = new SamuraiContext(_options);
    var repo = new DisconnectedData(context);
    Assert.AreEqual(2, repo.GetSamuraiReferenceList().Count);
  }
```

You may have noticed that I didn't call the seed method or create the options. I've moved that logic to the test class constructor so I don't have to repeat it in my tests. The _options variable is declared for the full scope of the class:

```
private DbContextOptions<SamuraiContext> _options;

  public TestDisconnectedData() {
    _options =
      new DbContextOptionsBuilder<SamuraiContext>().UseInMemoryDatabase().Options;
    SeedInMemoryStore();
  }
```

Now that I've moved the seed method into the constructor, you might think (as I did) that it will get called only once. But that's not the case. Did you know that a test class constructor gets hit by every test method that's run? In all honesty, I had forgotten this until I noticed that tests were passing when run on their own, but failing when I ran them together. That was before I added in the check to see if the samurai data already existed in memory. Every method that triggered the seed method to be called would be seeding the same collection. This would happen whether I was calling the seed method in each test method or only once in the constructor. The check for pre-existing data protects me either way.

There's a nicer way to avoid the problem of conflicting in-memory data stores. InMemory allows you to provide a name for its data store.

If you want to move the creation of the DbContextOptions back into the test method, and do so for each method, specifying a unique name as a parameter of UseInMemory will assure that each method is using its own data store.

I refactored my test class by removing the class-wide _options variable and the class constructor. Instead, I use a method for creating the options for a named data store and seeding the particular data store that takes in the desired name as a parameter:

```
private DbContextOptions<SamuraiContext> SetUpInMemory(string uniqueName) {
  var options = new DbContextOptionsBuilder<SamuraiContext>()
                  .UseInMemoryDatabase(uniqueName).Options;
  SeedInMemoryStore(options);
  return options;
}
```

I modified the signature and first line of the SeedInMemoryStore to use the configured options for the unique data store:

```
private void SeedInMemoryStore(DbContextOptions<SamuraiContext> options) {
  using (var context = new SamuraiContext(options)) {
```

And each test method now uses this method along with a unique name to instantiate the DbContext. Here's the revised CanRetrieveAllSamuraiValuePairs. The only change is that I'm now passing in the new SetUpInMemory method along with the unique data store name. A nice pattern recommended by the EF team is to use the test name as the name of the InMemory resource:

```
[TestMethod]
  public void CanRetrieveListOfSamuraiValues() {
  using (var context = new
SamuraiContext(SetUpInMemory("CanRetrieveListOfSamuraiValues"))) {
    var repo = new DisconnectedData(context);
    Assert.IsInstanceOfType(repo.GetSamuraiReferenceList(),
                            typeof(List<KeyValuePair<int, string>>));
   }
 }
```

Other test methods in my test class have their own unique data store names. And now you see there are patterns for using a unique set of data, or sharing a common set of data across test methods. When your tests are writing data to the in-memory data store, the unique names allow you to avoid side effects on other tests. Keep in mind that EF Core 2.0 will always require a name to be supplied, as opposed to the optional parameter in EF Core 1.1.

There's one last tip I want to share about the InMemory data store. In writing about the first test, I pointed out that both the SQL Server and SQLite providers insert a temporary value to the Samurai's key property when the object is added to the context. I didn't mention that if you specify the value yourself, the provider won't overwrite that. But in either case, because I'm using the default database behavior, the database overwrites the value with its own generated primary key value. With the InMemory provider, however, if you supply a key property value, that will be the value that the data store uses. If you don't supply one, the InMemory provider uses a client-side key generator whose value acts as the data-store assigned value.

The samples I've used come from my EF Core: Getting Started course on Pluralsight (bit.ly/PS_EFCoreStart), where you can learn more about EF Core, as well as testing with EF Core. The sample code is also included as a download with this article. ■

Julie Lerman *is a Microsoft Regional Director, Microsoft MVP, software team mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework," as well as a Code First and a DbContext edition, all from O'Reilly Media. Follow her on Twitter: @julielerman and see her Pluralsight courses at juliel.me/PS-Videos.*

Thanks to the following Microsoft technical expert for reviewing this article: *Rowan Miller*

Bringing Docker to Windows Developers with Windows Server Containers

Taylor Brown

For the last couple of years Docker and containers have been one of the hottest topics in dev circles, and in enterprises, around the world. The release of Windows Server 2016 last fall added a lot to the conversation by opening containers to Windows developers. How did the world of Windows and Docker come together? It started during the gorgeous Puget Sound summer of 2014 as the Windows Base team embarked on a new project that would ultimately become Windows Server Containers. This is the story behind the code, and a glimpse into what it was like to build one of the top new features in Windows Server 2016.

History of Containers and the Root of Docker

In 2013, containers quickly started generating interest at the keyboard of Solomon Hykes, who at the time was the CTO and founder of a Platform-as-a-Service (PaaS) startup, DotCloud. Hykes took a set of relatively obscure and difficult-to-use Linux kernel features and brought them together under an open source tool he called Docker. He wasn't intentionally trying to become the king of containers, but was looking for a solution to a problem that plagued DotCloud: How could developers provide code that worked the same way on their servers as it did in their working environment?

Hykes took a set of relatively obscure and difficult-to-use Linux kernel features and brought them together under an open source tool he called Docker.

A real problem for services like DotCloud stemmed from the extensive and diverse set of software applications customers wanted to deploy—software built with different development processes, different patch cycles and requirements, written in different languages (both code and spoken), and with different dependencies. Hardware virtualization—virtual machines (VMs)—was the best tool available, but it presented challenges in shipping software from developer laptops to production. Either you had to use fully configured VMs from the developer, which made scalability and

This article discusses:
- History of containers and the root of Docker
- Windows Server Containers
- Hyper-V isolation
- How Docker and Windows Server Containers came together

Technologies discussed:
Windows Server 2016, Docker, Linux

DevExpress®

Your Next Great App Starts Here

DEPARTMENT

FULL NAME	ADDRESS	CITY	STATE	ZIPCODE	EMAIL
Engineering (Count=9)					
Bart Arnaz	13109 Old Myford ...	Irvine	CA	92602	barta@dx-email.com
Leah Simpson	6755 Newlin Ave	Whittier	CA	90601	leahs@dx-email.com
Arnie Schwartz	125 W Elm St	Brea	CA	92821	arnolds@dx-email.com
Billy Zimmer	1325 Prospect Dr	Redlands	CA	92373	williamz@dx-email.com
Samantha Piper	200 E Ave 43	Los Angeles	CA	90031	samanthap@dx-email.com
Maggie Boxter	3101 W Harvard St	Santa Ana	CA	92704	margaretb@dx-email.com
Terry Bradley	4595 Cochran St	Simi Valley	CA	93063	terryb@dx-email.com
	1635 Woods Drive	Los Angeles	CA	90069	bradf@dx-email.com
	200 N. Spring St	Los Angeles	CA	90012	stu@dx-email.com
Count=6)					
	1700 S Grandview ...	Alhambra	CA	91803	gretas@dx-email.com
	4600 N Virginia Rd.	Long Beach	CA	90807	sandraj@dx-email.com
	2211 Bonita Dr.	Glendale	CA	91208	cindys@dx-email.com
	501 N Main St	Los Angeles	CA		
	7570 McGroarty Ter	Tujunga	CA		
	12100 Mora Dr	Santa Fe Sprin...	CA		
	1120 Old Mill Rd.	San Marino	CA		
	7776 Torreyson Dr	West Hollywo...	CA		
	527 W 7th St	Los Angeles	CA		
	10385 Shadow Oak...	Chatsworth	CA		
	1100 Pico St	San Fernando	CA		
	309 Monterey Rd	South Pasadena	CA		

When Only the Best Will Do

With the DevExpress Universal Subscription, you'll deliver amazing, high-impact user experiences for Windows®, the web and your mobile world. Get started today and see why your peers consistently vote DevExpress products #1.

Download your free 30-day trial at **devexpress.com** and let's build great apps together.

UI Controls for DESKTOP / WEB / MOBILE

Unleash the UI Superhero in you and create solutions that fully address customer requirements today and leverage your code investments to build next generation touch-enabled solutions for tomorrow. Whether building an Office® inspired app or a data centric analytics dashboard, DevExpress Universal ships with everything you'll need to build your best, without limits or compromise.

And with industry recognized support services, we'll be with you every step of the way — making certain that our products fulfill their promise. Visit **devexpress.com/support** or email **support@devexpress.com** anytime, with any DevExpress product question.

Geoffrey Jones
Code21 Solutions Pty Ltd

We have been using DevExpress products for over 5 years and been delighted with the results. DevExpress have helped us to develop an enterprise-ready labour management solution with WinForms and ASP.NET modules. As a small team, it would not have been possible for us to develop an application with a polished UI, highly scalable architecture and flexible customization options without the assistance of DevExpress.

Office-Inspired Apps

Touch-Enabled Windows & Web Apps

HTML5 Mobile Apps

Reporting-Dashboard & Analytics Apps

Experience the DevExpress Difference Risk Free

We are so confident in our tools that we back them with a 60-day unconditional money-back guarantee. If in your first 60 days of ownership you find that our products do not meet your business needs, you can return them to us for a full, no questions asked refund.

To get started and to choose a subscription that's right for you, email **info@devexpress.com** or call **+1 (818) 844-3383**.

60 DAY MONEY-BACK GUARANTEE

Grid Controls

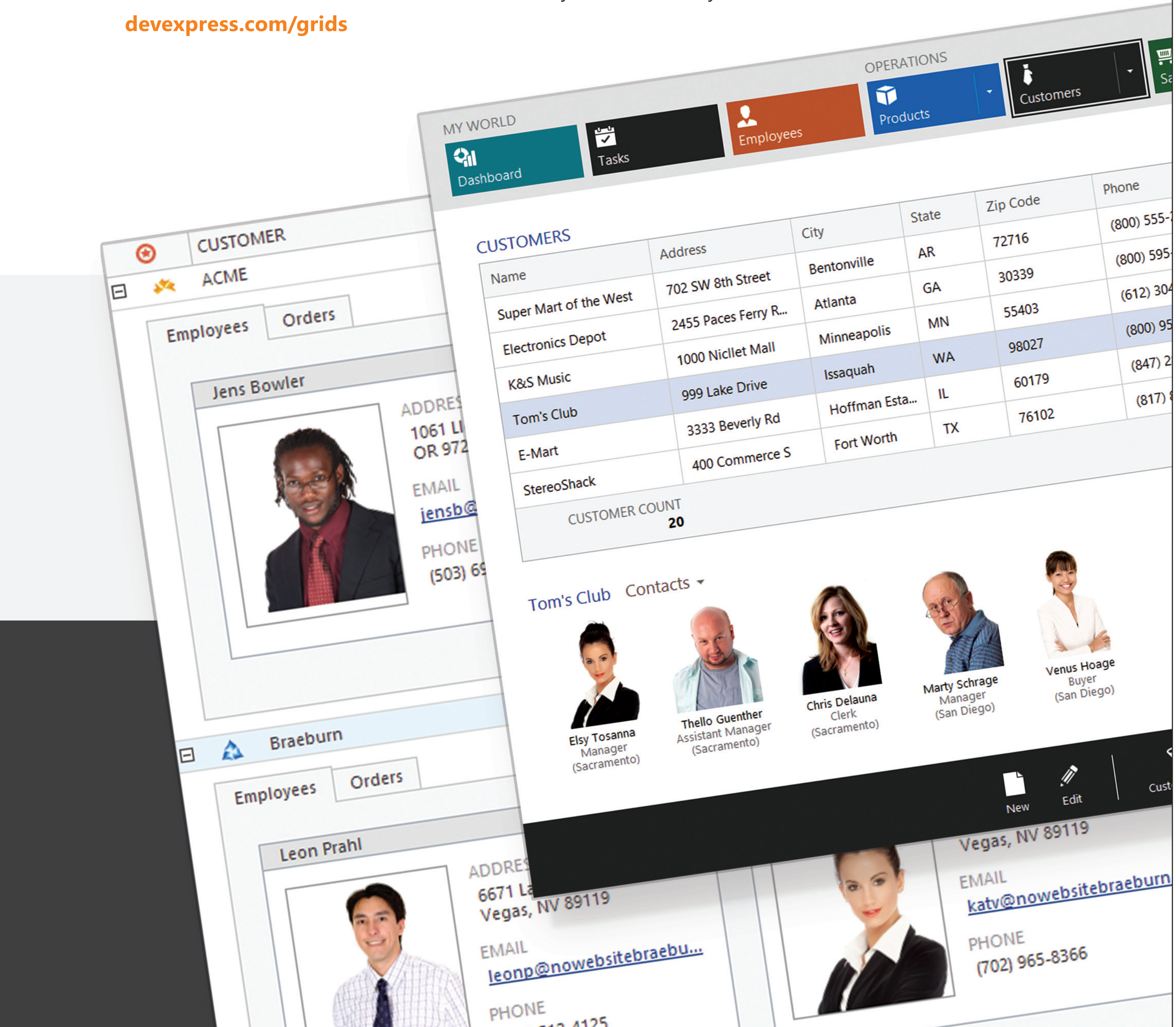
Blazing fast, feature-complete and just plain beautiful.

DevExpress Grid controls are Outlook® inspired record editing and data shaping components, allowing your end-users to manage and arrange information on-screen as business requirements dictate. Our data grids ship with dozens of industry leading features including integrated master-detail support and multiple data layout options such as Card and Windows Explorer views. And whether you target desktops or touch devices, all DevExpress Grid Controls support a broad range of user interaction options.

WIN WPF ASP MVC

Your Next Great App Starts @DevExpress

See our Grid controls in action and download your free 30-day trial.
devexpress.com/grids



CUSTOMERS

Name	Address	City	State	Zip Code	Phone
Super Mart of the West	702 SW 8th Street	Bentonville	AR	72716	(800) 555-
Electronics Depot	2455 Paces Ferry R...	Atlanta	GA	30339	(800) 595-
K&S Music	1000 Nicllet Mall	Minneapolis	MN	55403	(612) 304
Tom's Club	999 Lake Drive	Issaquah	WA	98027	(800) 95
E-Mart	3333 Beverly Rd	Hoffman Esta...	IL	60179	(847) 2
StereoShack	400 Commerce S	Fort Worth	TX	76102	(817)

CUSTOMER COUNT
20

Tom's Club Contacts

Elsy Tosanna
Manager
(Sacramento)

Thello Guenther
Assistant Manager
(Sacramento)

Chris Delauna
Clerk
(Sacramento)

Marty Schrage
Manager
(San Diego)

Venus Hoage
Buyer
(San Diego)

ACME

Jens Bowler
ADDRES
1061 LI
OR 972
EMAIL
jensb@
PHONE
(503) 69

Braeburn

Leon Prahl
ADDRES
6671 La
Vegas, NV 89119
EMAIL
leonp@nowebsitebraebu...
PHONE
4125

Vegas, NV 89119
EMAIL
katv@nowebsitebraeburn
PHONE
(702) 965-8366

Reporting

Inform and analyze with absolute ease.

The ever changing needs of the enterprise require that reporting platforms offer simple and straightforward end-user customization options so that consumers can freely manipulate report output for maximum clarity. Flexibility is key and ease-of-use paramount.

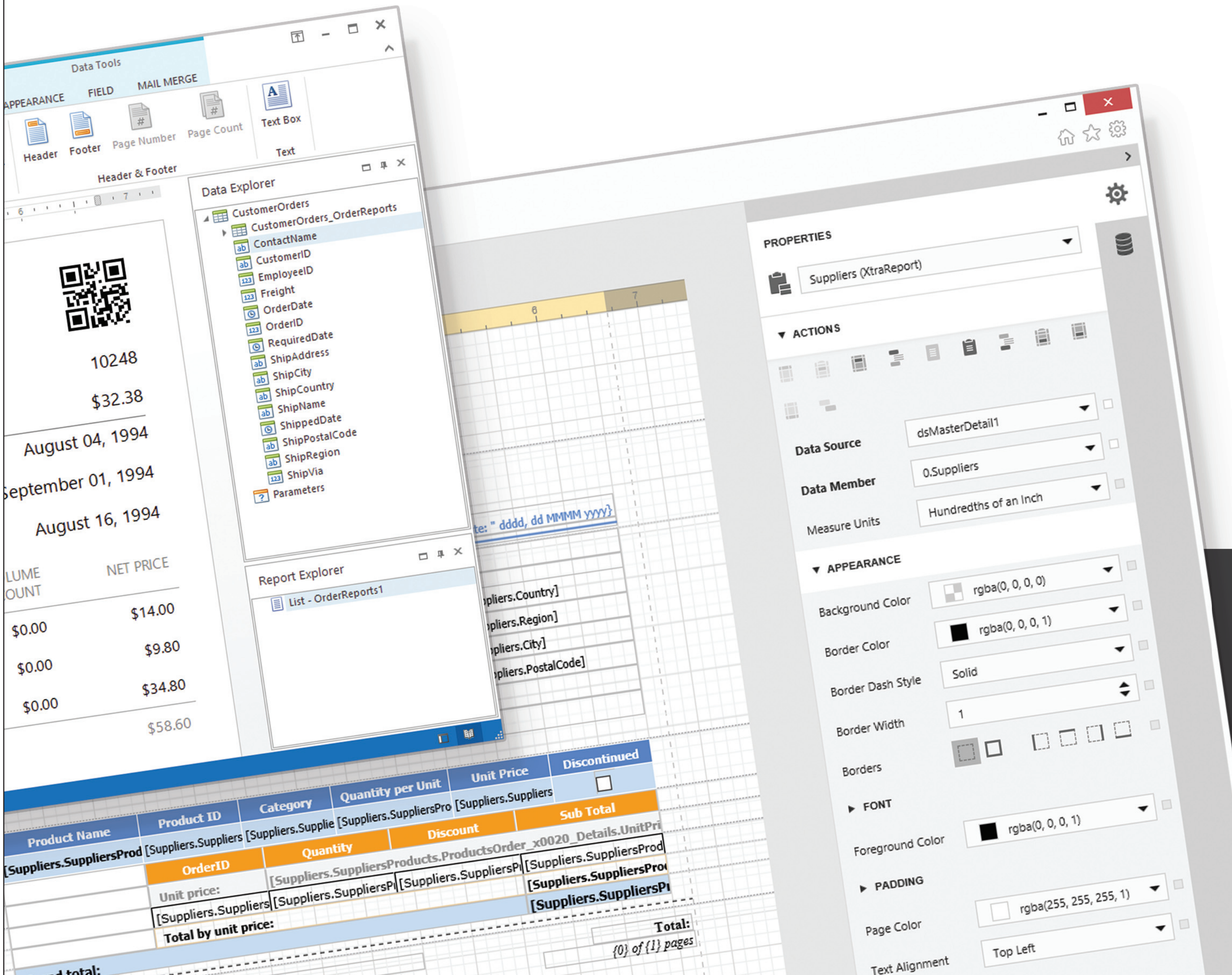
The DevExpress Reporting Suite for .NET was built with your end-users in mind and engineered so you can get the most out of your reporting and data analytics investments. Our Reporting Suite helps you overcome the limitations associated with traditional report tools by providing a fully integrated set of productivity tools, report wizards, pre-built report templates and end-user report designers for both Windows and the web.

WIN WPF ASP MVC

Your Next Great Report Starts @DevExpress

See how to solve your Windows® and web reporting requirements forever.

devexpress.com/reports



Suppliers (XtraReport)

Data Source	dsMasterDetail1
Data Member	0.Suppliers
Measure Units	Hundredths of an Inch
Background Color	rgba(0, 0, 0, 0)
Border Color	rgba(0, 0, 0, 1)
Border Dash Style	Solid
Border Width	1
Borders	
Foreground Color	rgba(0, 0, 0, 1)
Page Color	rgba(255, 255, 255, 1)
Text Alignment	Top Left

Spreadsheets (XLS, XLSx, CSV)

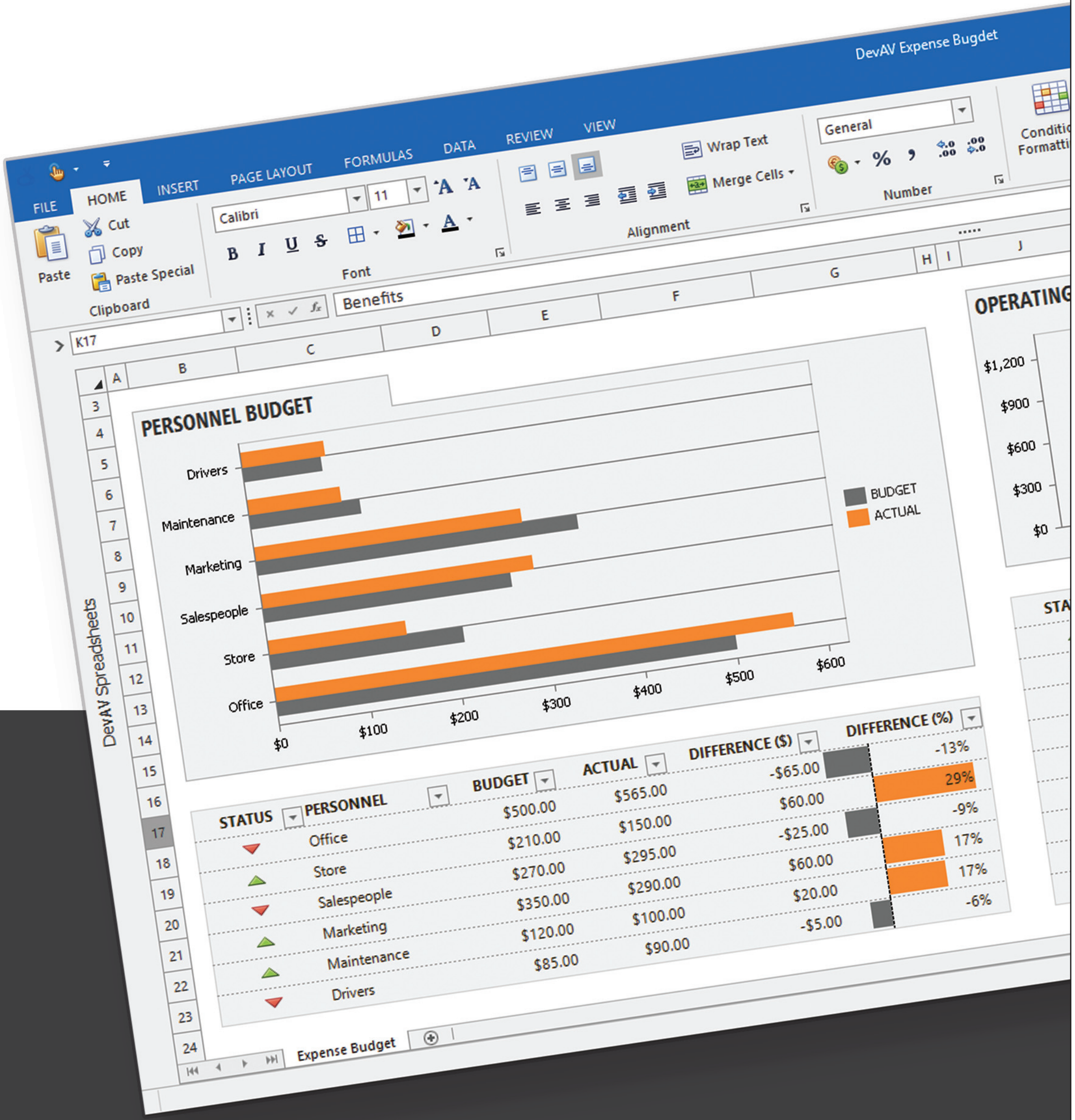
The power of Excel® at your fingertips.

It's no secret...spreadsheets are an important part of business and if you need to incorporate the UX and functionality end-users have come to expect from Microsoft Excel®, DevExpress Spreadsheet is the tool for you. The component library ships with dozens of easy-to-implement features including chart support and fully integrates with other DevExpress components like our Office® inspired Ribbon.

WIN WPF ASP MVC

Your Next Great Spreadsheet Starts @DevExpress

See how you can harness the power of spreadsheets in your next .NET app.
devexpress.com/spreadsheet



STATUS	PERSONNEL	BUDGET	ACTUAL	DIFFERENCE ($)	DIFFERENCE (%)
🔻	Office	$500.00	$565.00	-$65.00	-13%
🔺	Store	$210.00	$150.00	$60.00	29%
🔻	Salespeople	$270.00	$295.00	-$25.00	-9%
🔺	Marketing	$350.00	$290.00	$60.00	17%
🔺	Maintenance	$120.00	$100.00	$20.00	17%
🔻	Drivers	$85.00	$90.00	-$5.00	-6%

Word Processing (DOC, DOCx, RTF)

Create the perfect WYSIWYG document or report.

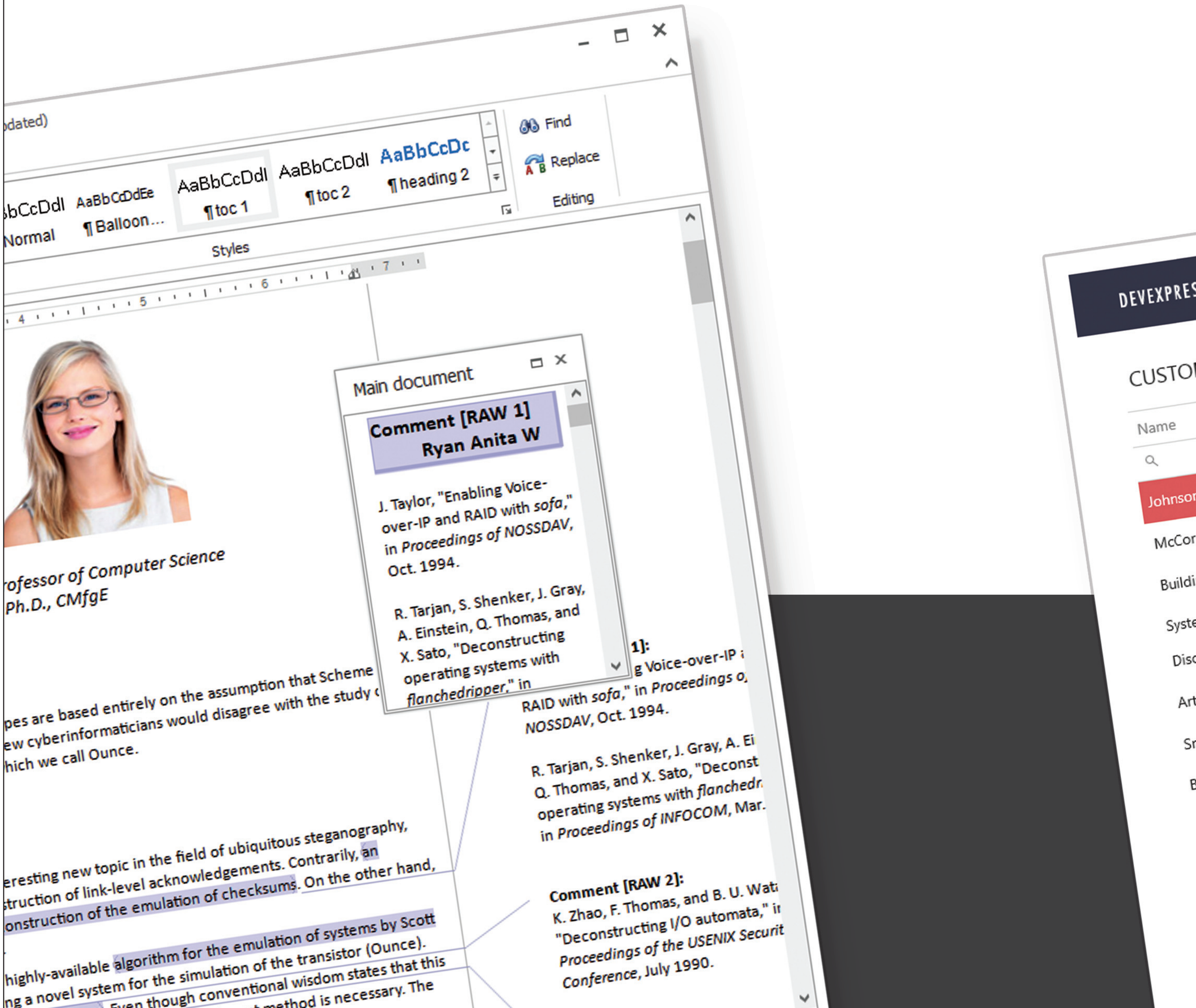
When rich text editing is a must and you need to replicate key features found inside Microsoft Word® (from mail merge and tables, to floating objects and document comments), look no further than the DevExpress Rich Text Editor Control. The library's comprehensive feature set includes printing and PDF export support and the availability of DevExpress Visual Studio project templates means you can incorporate our rich text editors in your next .NET app within minutes of install.

WIN WPF ASP MVC

Your Next Great Text Editor Starts @DevExpress

See how you can introduce Word® compatible rich text editing in your app.

devexpress.com/word



rofessor of Computer Science
Ph.D., CMfgE

pes are based entirely on the assumption that Scheme
ew cyberinformaticians would disagree with the study
hich we call Ounce.

eresting new topic in the field of ubiquitous steganography,
truction of link-level acknowledgements. Contrarily, an
onstruction of the emulation of checksums. On the other hand,

highly-available algorithm for the emulation of systems by Scott
ng a novel system for the simulation of the transistor (Ounce).
Even though conventional wisdom states that this
method is necessary. The

Comment [RAW 1]
Ryan Anita W

J. Taylor, "Enabling Voice-over-IP and RAID with *sofa*," in *Proceedings of NOSSDAV*, Oct. 1994.

R. Tarjan, S. Shenker, J. Gray, A. Einstein, Q. Thomas, and X. Sato, "Deconstructing operating systems with *flanchedripper*," in

1]:
g Voice-over-IP
RAID with *sofa*," in *Proceedings o*
NOSSDAV, Oct. 1994.

R. Tarjan, S. Shenker, J. Gray, A. Ei
Q. Thomas, and X. Sato, "Deconst
operating systems with *flanchedr*
in *Proceedings of INFOCOM*, Mar.

Comment [RAW 2]:
K. Zhao, F. Thomas, and B. U. Wat
"Deconstructing I/O automata," in
Proceedings of the USENIX Securit
Conference, July 1990.

Charting & Analytics

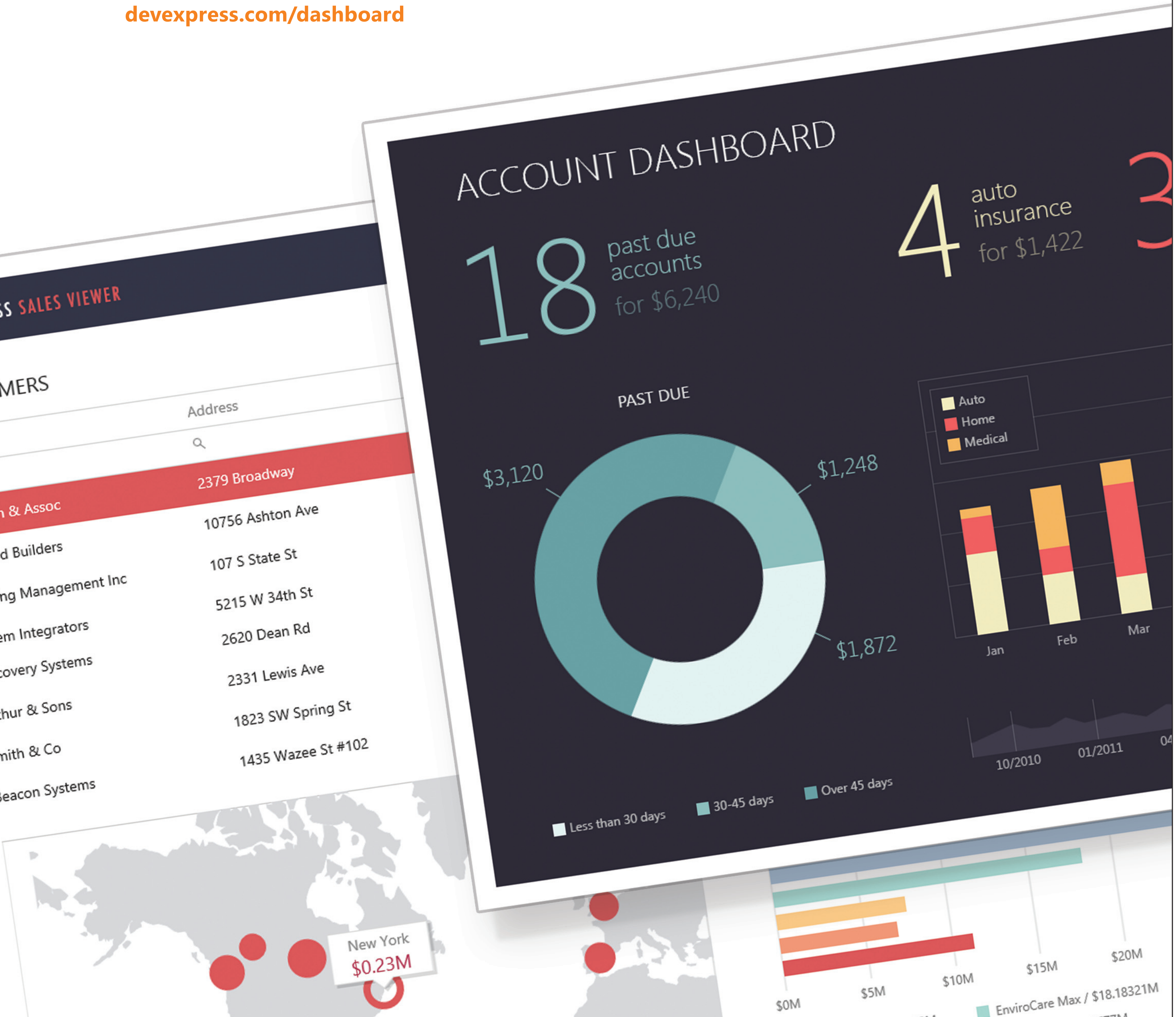
From zero to dashboard in record time.

With our integrated suite of charts, pivot grids, and multi-purpose dashboard widgets, you'll create information-rich decision support systems that are optimized for Windows, the web and your favorite mobile device. The UI components inside our subscriptions help you deliver adaptable, interactive and touch-enabled user experiences that can address a broad range of use-case scenarios. And perhaps best of all, the dashboards you build with DevExpress Universal can be distributed royalty-free.

WIN WPF ASP MVC

Your Next Great Dashboard Starts @DevExpress

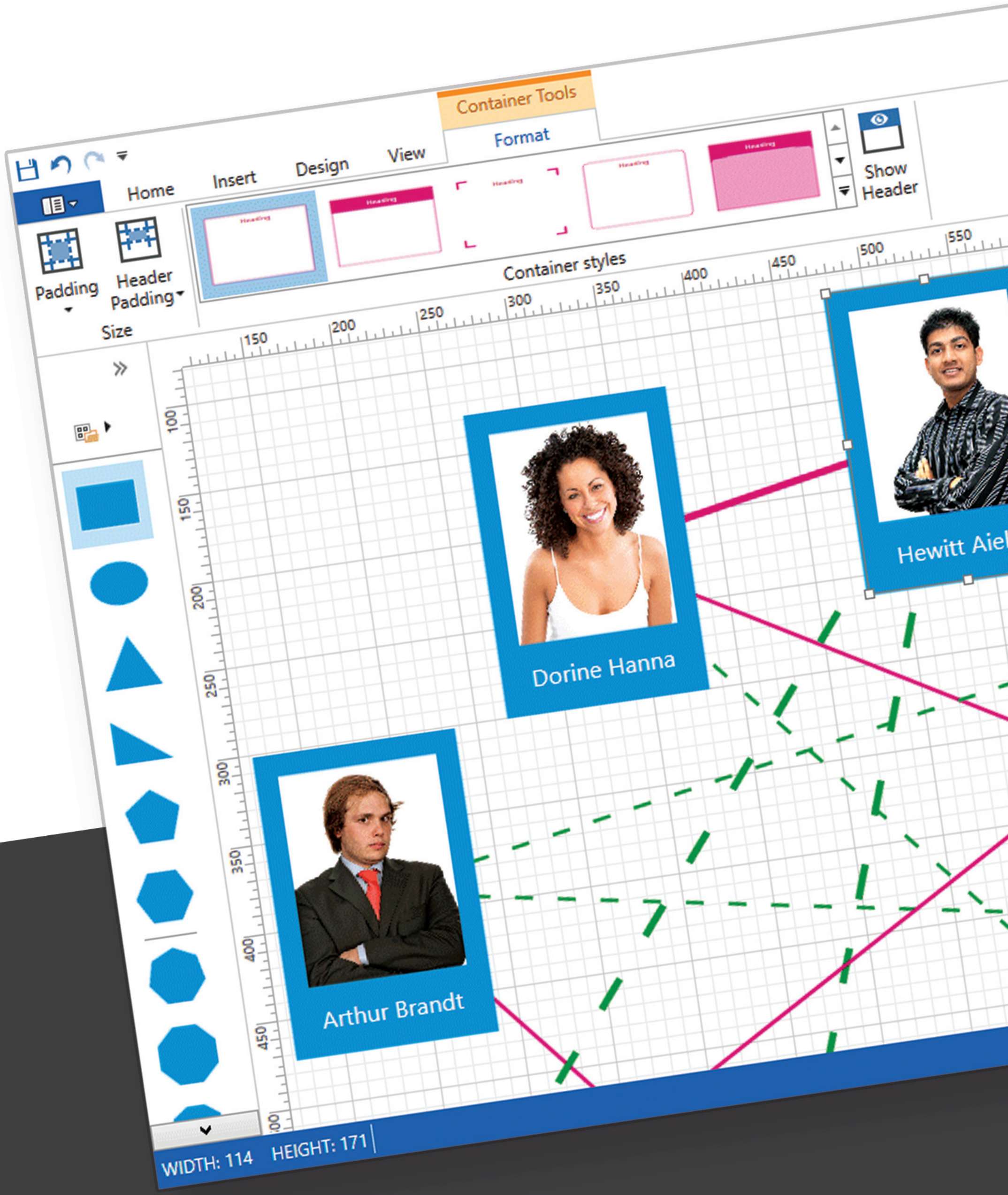
See how you can build future proof, enterprise-grade decision support systems.
devexpress.com/dashboard



READY · SET · GO

Download Your Free 30-Day Trial Today

devexpress.com/try



DevExpress®

management difficult, or you had to make deployment tools and scripts to take stock VMs and install the developer's applications, which isn't very flexible and can be fragile.

Hykes believed Docker was the answer to this problem and, looking back, he was on to something. However, he wasn't the first cloud service to look to containers; in fact, it was the needs of a different cloud service that kick-started the whole idea—Google. In 2006, a Linux kernel patch submitted by Rohit Seth, an engineer at Google, added support for grouping processes together under a common set of resource controls in a feature he called *cgroups*. Seth's description of that patch starts off with: "Commodity HW is becoming more powerful. This is giving opportunity to run different workloads on the same platform for better HW resource utilization" (bit.ly/2mhatrp). Although cgroups solved the problem of resource isolation, they didn't solve inconsistent distribution, which is why Docker uses not only cgroups but also another slice of Linux technology: namespaces.

Namespaces were introduced into the Linux kernel in 2002, providing a way to control what resources a process can see and what those resources are called. Namespaces are quite different from access controls because the process doesn't even know the resources exist or that it's using a version of them. A simple example of this is the process list: there could be 20 processes running on a server, yet a process running within a namespace might see only five of those processes with the rest hidden from view. Another example might be for a process to think it's reading from the root directory when in fact it's been virtualized from another separate location. It's the combination of cgroups and namespaces and Copy-on-Write (CoW) file-system technologies into an easy-to-use open source product that became the foundation of Docker.

Namespaces were introduced into the Linux kernel in 2002, providing a way to control what resources a process can see and what those resources are called.

By mid-2013, the Docker toolset that Hykes and his team built began to take off, becoming one of the top trending projects on GitHub and formally launching the Docker brand. Hykes' focus shifted from DotCloud to Docker and he ultimately spun off the DotCloud business while remaining the CTO of Docker Inc.

Windows Server Containers

During the same period that Docker was gaining notice in Linux circles, the Windows Base team had been looking at ways to isolate and increase the efficiency of Microsoft Azure services that executed customer or third-party code. A Microsoft research prototype code-named "Drawbridge" provided one avenue of investigation; the project had built a process isolation container leveraging a library OS (bit.ly/2aCOQxP). Unfortunately, Drawbridge had limitations relating to maintainability, performance and application compatibility, making it ill-suited as a general-purpose solution. Another even earlier prototype technology referred to as server silos initially seemed worth investigating. Silos expanded on the existing Windows Job Objects approach, which provides process grouping and resource controls (similar to cgroups in Linux) (bit.ly/2lK1AbI). What the server silos prototype added was an isolated execution environment that included file system, registry and object namespaces (similar to namespaces in Linux). The server silos prototype had been shelved years earlier in favor of VMs but would be reimagined as the foundation of Windows Server Containers.

The server silos prototype had been shelved years earlier in favor of virtual machines, but would be reimagined as the foundation of Windows Server Containers.

The server silo prototype code hadn't been looked at in years. It didn't even compile, let alone function, and it was prototype code written to prove the technique was viable in Windows, but far from production-ready. The team had a choice—start over from scratch or attempt to resurrect the prototype and start from there. We chose the latter. When the prototype was first developed, it was only a small team of developers proving that the technology was viable, but now the full force of the Windows engineering team was behind the project. Architects and engineers from across Windows were drafted to help. The storage team built the file system virtualization; the networking team built the network isolation; the kernel team built the memory management and scheduling abstractions; and so on.

Some big architectural questions remained; in particular, how would we handle system processes? In Linux, a container often runs just a single process that shares the system services in the kernel with the host and other containers. However, to improve serviceability and security, Windows has been moving code out of the kernel and into user mode processes for many years. This represented an issue for the team: Either we could share all the system services, requiring changes to all the system services to make them aware of containers, or we could start a new copy of the user mode system services in each container. This was a difficult decision—we worried about the density and startup time impact of starting new instances of all the user mode services in each container. On the other side, we worried about the complexity and ongoing cost of updating all the system services in Windows, both for us and for developers outside of Windows. In the end we landed on a mix of the two approaches—a select set of services was made container-aware, but most services run in each container.

The impact on density was minimal because the containers share read-only memory with each other and the host, so only the private memory is per-container. Startup time was a significant challenge, however, calling this decision into question many times; when we first demonstrated Windows Server Containers in the keynote of Build 2015, it took several seconds to start, in large part because of the startup time of the system services. However, the Windows Server performance team was on the case. They profiled, analyzed and worked with teams across Windows to make their services faster and reduce dependencies to improve parallelism. The result of this effort not only made container startup faster but actually improved Windows startup time, as well. (If your Xbox or Surface started booting faster last year, you can thank containers.) Container startup went from about seven to eight seconds to a sub-second startup time in less than a year, and this trajectory to reduce startup time continues even today.

Hyper-V Isolation

Often, the first question I get regarding Hyper-V isolation is something like, "Don't containers provide isolation already? So why do I need Hyper-V?" Containers do provide isolation and for most scenarios that isolation is likely completely sufficient. However, the risk is that if an attacker is able to compromise the kernel it could potentially break out of the container and impact other containers or the host. With kernel exploits being relatively common in Windows (typically several per year), the risk for services like Azure Automation or Azure Machine Learning that consume and execute end-user or third-party code on a shared infrastructure is too high to rely on just kernel isolation. Teams building and operating these types of services either had to manage the density and startup cost of full VMs or build different security and isolation techniques. What was needed was a general-purpose isolation mechanism that was hostile to intruders yet multi-tenant safe: Windows Server Containers with Hyper-V isolation.

The team was already hard at work on Windows Server Containers, and this provided a great experience and management model for teams building the services. By coupling the technology with the well-tested isolation of Hyper-V, we could provide the security required. However, we needed to solve the startup time and density challenges traditionally associated with VMs.

Hyper-V, like most virtualization platforms, was designed to run guests with a variety of OSes both old and new. With the goal of behaving as much like hardware as possible, to achieve these objectives the solution most virtualization platforms chose was emulating common hardware. As virtualization became commonplace, however, OSes were "enlightened" (specifically modified to operate well as a guest VM) such that much of the emulation was no longer required. A good example of this is Hyper-V Generation 2 VMs, which discard emulation in favor of improved startup time and performance, but still achieve the same objective of behaving the same as if the guest was running directly on hardware (bit.ly/2lPpdAg).

To improve serviceability and security Windows has been moving code out of the kernel and into user mode processes for many years.

For containers, we had a different need and different goals: We didn't need to run any older OSes and we knew exactly what the workload inside the VM was going to be—a container. So we built a new type of VM, one that was designed to run a container. To address the need for a fast startup time we built cloning technology. This was always a challenge for traditional VMs because the OS becomes specialized with things like hostnames and identity, which can't easily be changed without a reboot. But because containers have their own hostname and identity, that was no longer an issue. Cloning also helped with the density challenge, but we had to go further: We needed memory sharing.

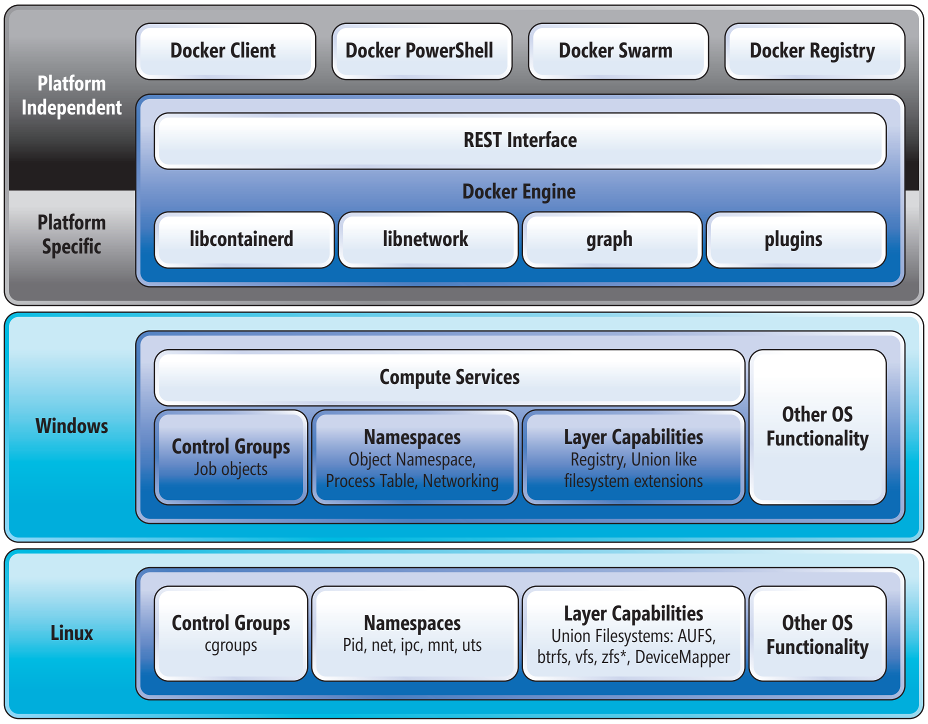


Figure 1 **Comparing the Basic Architecture of Containers and Docker Across Windows and Linux**

There are two approaches to sharing memory. You can look for memory that's common across multiple VMs and effectively de-duplicate (though memory randomization technology in most kernels makes this difficult). Or you can follow the same approach the kernel does by separating read-only (public) memory from read-write (private) memory. The latter typically requires that the memory manager in guest VMs interact with each other, which is counter to the isolation requirement. However, by changing the way the VMs boot and access files, we found a way where the host doesn't have to trust the guest and the guests don't have to trust each other. Instead of the VM booting from and accessing files from a virtual hard disk, it boots and accesses its files directly from the host file system. This means that the host can provide the same sharing of read-only (public) memory. This was the key to improving density by several orders of magnitude and it put us on a path to continue improving density for many years to come.

The other value we discovered with Hyper-V isolation is that by running a different kernel for the container for developers building containerized applications on their Windows 10 machines, we could still run the server kernel, ensuring their applications would work the same way in production as they do on the development machines. Thus, with the Windows 10 Anniversary Update, we enabled Windows Server Containers with Hyper-V isolation and worked with Docker on Docker for Windows to take full advantage of the new technology for developers.

Docker and Windows Server Containers

One question remained—how would users interact with this new platform technology? In the Linux world Docker had been garnering praise and was quickly becoming the de facto standard for container management. Why not enable users to use Windows Server Containers the same way? That fall I flew down to San Francisco to meet with Docker, unsure what the company would think of a Windows-based container and whether it would be interested in building on top of Windows at all. I was in for a surprise: Solomon thought the Windows container idea was great! But would the company build on top of it? That conversation changed the face of the project completely. Solomon simply said, "You know Docker is open source, you can add the code to make it work on Windows and we'll help," and we did just that. Since then, John Howard, a software engineer on the Hyper-V team, has become a maintainer of the Docker project and, in fact, has climbed to fourth all-time code contributor (bit.ly/2lAmaZX). **Figure 1** shows the basic architecture of containers and Docker across Windows and Linux.

In the end we landed on a mix of the two approaches—a select set of services was made container-aware, but most services run in each container.

Bringing It All Together

Four months ago at Microsoft Ignite, we launched Windows Server 2016 and announced the expansion of our partnership with Docker, which means it will provide its commercially supported version of the Docker Engine at no additional charge to Windows Server customers. Since then, it's been a whirlwind of activity. Customers like Tyco have been using Docker and Windows Server Containers to revolutionize the way they build software and to modernize existing applications, all with the same platform (bit.ly/2dWqIFM). Visual Studio 2017 has fully integrated tooling for Windows and Linux containers, including F5 debugging, and Visual Studio Code has Dockerfile and compose support baked right in. Both Azure and Amazon's container services have added support for Windows Server Containers and well more than 1 million Windows-based container images have been pulled from Docker Hub. To achieve end-to-end security and orchestration, Docker Datacenter is the platform for developers and sysadmins to build, ship and run distributed applications anywhere. With Docker, organizations shrink application delivery from months to minutes, frictionlessly move workloads between datacenters and the cloud and achieve 20 times greater efficiency in their use of computing resources.

What was needed was a general-purpose isolation mechanism that was hostile to intruders, yet multi-tenant safe.

When I took on containers I knew it was going to be a high-stress project. I knew it was going to take some long nights, some working weekends and a lot of effort, but it's worth it because it helped millions of developers build more apps faster. I also knew it was going to be a lot of fun and that it had the opportunity to really change the way people developed and ran applications on Windows. It's been more fun than I could have ever expected and, while it was also more work than I anticipated, I wouldn't trade this experience for anything. I recall one weekend early in the project, looking out the window of my office as I worked at a gorgeous, sunny summer day and thinking to myself, "I sure hope people are gonna use this stuff . . ." ■

Taylor Brown *is a principal program management lead in the Windows and Devices Group at Microsoft. As a member of the Base Windows engineering team he's responsible for Windows Server Developer strategy, as well as focusing specifically on container technologies, including Windows Server Containers. Brown started his career in Windows working on the 1394/Firewire stack for Windows 2003, then working on ACPI/power management for Windows Server 2003 SP1 before joining the newly formed virtual machine team. Since then he has contributed to every VM technology shipped by Microsoft including Virtual PC, Virtual Server and every version of Hyper-V, making him recognized as an industry expert in virtualization technologies. Reach him at taylorb@microsoft.com.*

Thanks to the following technical expert for reviewing this article: *David Holladay*

Modernizing Traditional .NET Apps with Docker

Elton Stoneman

The Microsoft .NET Framework has been a successful application platform for 15 years, with countless business-critical apps running on older versions of the Framework and older versions of Windows Server. These traditional apps still offer great business value, but they're likely to be difficult to maintain, upgrade, extend and manage. Equally, they may not justify the investment needed for a full rewrite. With Docker, a platform for running applications in lightweight containers, and Windows Server 2016, you can give traditional apps a new lease on life—adding features, increasing security and performance, and moving toward continuous deployment—without a lengthy and expensive rebuild project.

This article discusses:

- Migrating .NET apps to containers
- Pulling dependencies from Docker Hub
- Breaking a monolithic application into smaller services
- Adding self-service analytics
- Running Dockerized solutions on Microsoft Azure

Technologies discussed:

Microsoft .NET Framework, Windows Server 2016, Docker, SQL Server, Elasticsearch, Kibana

Code download available at:

msdn.com/magazine/0417magcode

In this article I'll take a monolithic ASP.NET WebForms app that connects to a SQL Server database, and modernize it by taking advantage of the Docker platform. I'll start by moving the whole app as is to Docker, without any code changes, and run the Web site and database in lightweight containers. Then I'll show a feature-driven approach to extending the app, improving performance and giving users self-service analytics. With the Docker platform you'll see how to iterate with new versions of the app, upgrade the components quickly and safely, and deploy the complete solution to Microsoft Azure.

Where Docker Fits in .NET Solutions

Docker is for server applications—Web sites, APIs, messaging solutions and other components that run in the background. You can't run desktop apps in Docker because there's no UI integration between the Docker platform and the Windows host. That rules out running Windows Forms or Windows Presentation Foundation (WPF) apps in containers (although you could use Docker to package and distribute those desktop apps), but Windows Communication Foundation (WCF), .NET console apps and all flavors of ASP.NET are great candidates.

To package an application to run in Docker, you write a small script called a Dockerfile that automates all the steps for deploying the app. Typically this includes Windows PowerShell commands for configuration and instructions to copy application content and set up any dependencies. You can unzip compressed archives or install

MSIs, too, but the packaging process is all automated, so you can't run an install process that has a Windows UI and needs user input.

When you're looking at a solution architecture to figure out which parts can run in Docker containers, keep in mind any component that can be installed and run without the Windows UI is a good candidate. This article focuses on .NET Framework apps, but you can run anything in a Windows container that runs on Windows Server, including .NET Core, Java, Node.js and Go apps.

Migrating .NET Apps to Containers

How you migrate to Docker depends on how you're currently running your app. If you have a fully configured app running in a Hyper-V VM, the open source Image2Docker tool can automatically generate a Dockerfile from the VM's disk. If you have a build process that publishes an MSI or a WebDeploy package, it's easy to write your own Dockerfile by using one of Microsoft's base images on Docker Hub.

The Dockerfile is like a deployment guide for the Web application, but instead of being a vague human document, it's a precise and actionable script.

Here's a complete Dockerfile that scripts the packaging of an ASP.NET WebForms app into a Docker image:

```
FROM microsoft/aspnet:windowsservercore-10.0.14393.693
SHELL ["powershell"]

RUN Remove-Website -Name 'Default Web Site'; \
    New-Item -Path 'C:\web-app' -Type Directory; \
    New-Website -Name 'web-app' -PhysicalPath 'C:\web-app' -Port 80 -Force

EXPOSE 80
RUN Set-ItemProperty -Path 'HKLM:\SYSTEM\CurrentControlSet\Services\
Dnscache\Parameters' \
    -Name ServerPriorityTimeLimit -Value 0 -Type DWord

COPY ProductLaunch.Web /web-app
```

Nine lines of script are all I need, and there are no application changes. This could be an ASP.NET 2.0 app, currently running on Windows Server 2003—with this Dockerfile I can build it into an image that immediately upgrades the app to Windows Server 2016 and the .NET Framework 4.5. I'll walk through each of those instructions:

- *FROM microsoft/aspnet* tells Docker which image to use as the starting point. In this case, it's a Microsoft image with IIS and ASP.NET installed on top of a specific version of Windows Server Core.
- *SHELL ["powershell"]* changes to a different shell for the rest of the Dockerfile, so I can run PowerShell cmdlets.
- *RUN Remove-Website* uses PowerShell to set up IIS, removing the default Web site and creating a new one with a known location for the application.
- *EXPOSE 80* opens port 80 explicitly to allow network traffic into the container as Docker containers are locked down by default.
- *RUN Set-ItemProperty* turns off the Windows DNS cache inside the image, so any DNS requests get served by Docker.
- *COPY ProductLaunch.Web* copies the published Web site project from the ProductLaunch.Web directory on the host into the image.

The Dockerfile is like a deployment guide for the Web application, but instead of being a vague human document, it's a precise and actionable script. To produce the packaged app I run the docker build command from the directory that contains the Dockerfile and the published Web site:

```
docker build --tag sixeyed/msdn-web-app:v1 .
```

This command builds a Docker image with the name sixeyed/msdn-web-app and the tag v1. The name contains my user account for the Hub (sixeyed), so I can share this image by signing in with my credentials and pushing it to the Hub. Tags are useful for versioning images, so when I package a new version of the application, the image name will stay the same, but the tag will be v2.

Now I can run a container from the image and that will start the application, but the sample app has a dependency on SQL Server so I need SQL Server running before I can start the Web site.

Pulling Dependencies from Docker Hub

Docker has a networking stack that lets containers reach each other over a virtual network, and also lets containers reach external hosts running on the physical network. If I had a SQL Server instance running on a machine in the network, the ASP.NET app in the container could use it—I'd just need to specify the server name in the connection string. Or I can run SQL Server in a container, and the Web app will be able to reach it by using the container name in the connection string.

There are more than half a million images on Docker Hub, which have been downloaded more than 9 billion times.

SQL Server Express is available on Docker Hub in an image maintained by Microsoft. To start a database container from that image, I run:

```
docker run --detach `
 --publish 1433:1433 `
 --env sa_password=MSDNm4g4z!n3 `
 --env ACCEPT_EULA=Y `
 --name sql-server `
 microsoft/mssql-server-windows-express
```

This starts a container in the background with the detach flag and publishes port 1433, so I can connect to the SQL instance in the container from outside, perhaps using SQL Server Management Studio on the host. The env options are key-value pairs, which Docker surfaces inside the container as system environment variables. The SQL Server image uses these values to confirm that the license agreement has been accepted, and to set the password for the sa user.

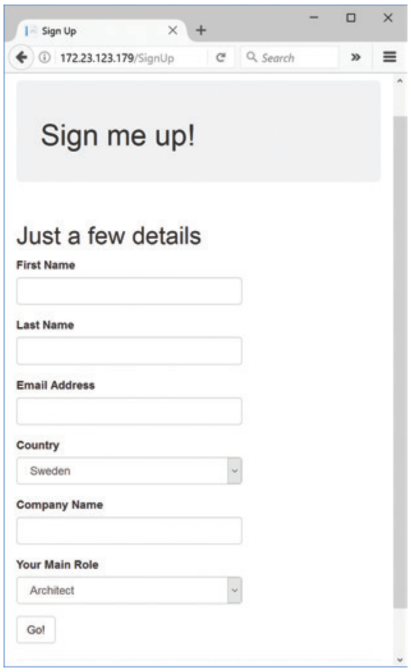
To run a container, Docker needs to have a copy of the image locally. Distribution is built into the Docker platform, so if you don't have the SQL Server Express image locally when you run this command, Docker will download it from the Hub. There are more than half a million images on Docker Hub, which have been downloaded more than 9 billion times. Docker started in the Linux world and the majority of those images are Linux apps, but there are a growing number of high-quality Windows apps you can download and drop straight into your solution.

SQL Server is running in a Docker container now, and my Web app uses sql-server as the hostname in the connection string so it will connect to the database running in Docker. I can start the WebForms application in the background and publish port 80 to make the Web site accessible:

```
docker run --detach `
 --publish 80:80 `
 sixeyed/msdn-web-app:v1
```

If an external machine sends a request on port 80 to my host, Docker receives the request and transparently forwards it to the ASP.NET app running in the container. If I'm working on the host, I need to use "docker inspect" to get the container's IP address and browse to the container to see the site, which is a simple product launch microsite. You can see the data capture page from the site running in Docker in **Figure 1**.

Run "docker ps" and you'll see a list of all running containers. One is a database and one is a Web application, but you manage them both in the same way—"docker top" shows you the processes running in the container; "docker logs" shows you the log output from the app; and "docker inspect" shows you which ports are open and a host of other information about the container. Consistency is a major benefit of the Docker platform. Apps are packaged, distributed and managed in the same way, no matter what technology they use.


Sign Up

172.23.123.179/SignUp

Sign me up!

Just a few details

First Name

Last Name

Email Address

Country: Sweden

Company Name

Your Main Role: Architect

Go!

Figure 1 **A Signup Page for a Site Running in Docker**

Splitting Features from Monolithic Apps

Now that the application is running on a modern platform, I can start to modernize the application itself. Breaking a monolithic application down into smaller services can be a significant project of work, but you can take a more targeted approach by working on key features, such as those that change regularly, so you can deploy updates to a changed feature without regression testing the whole application. Features with non-functional requirements that can benefit from a different design without needing a full re-architecture of the app can also be a good choice.

I'm going to start here by fixing a performance issue. In the existing code, the application makes a synchronous connection to the database to save the user's data. That approach doesn't scale well—lots of concurrent users would make a bottleneck of SQL Server. Asynchronous communication with a message queue is a much more scalable design. For this feature, I can publish an event from the Web app to a message queue and move the data-persistence code into a new component that handles that event message.

Consistency is a major benefit of Docker. Apps are packaged, distributed and managed in the same way, no matter what technology they use.

This design does scale well. If I have a spike of traffic to the Web site I can run more containers on more hosts to cope with the incoming requests. Event messages will be held in the queue until the message handler consumes them. For features that don't have a specific SLA, you can have one message handler running in a single container and rely on the guarantees of the message queue that all the events will get handled eventually. For SLA-driven features you can scale out the persistence layer by running more message-handler containers.

The source code that accompanies this article has folders for version 1, version 2 and version 3 of the application. In version 2, the SignUp.aspx page publishes an event when the user submits the details form:

```
var eventMessage = new ProspectSignedUpEvent
{
  Prospect = prospect,
  SignedUpAt = DateTime.UtcNow
};

MessageQueue.Publish(eventMessage);
```

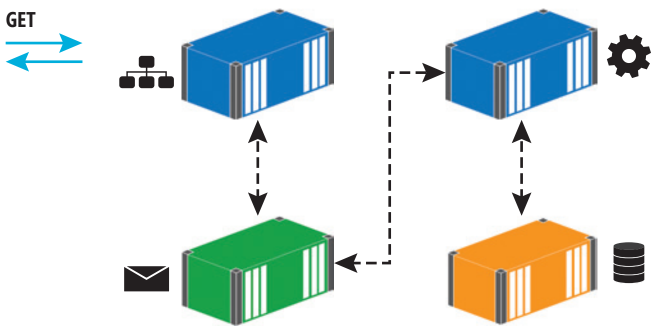


Figure 2 **The Modernized Application Has Many Working Parts**

/n software

We didn't invent the Internet...

...but our components help **you** power the apps that bring it to business.

TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- E-Business
  AS2, AS4, EDI/X12, OFTP …
- Credit Card Processing
  Authorize.Net, ACH, 3-D Secure …
- Shipping & Tracking
  FedEx, UPS, USPS …
- Accounting & Banking
  QuickBooks, OFX, SAP …
- Internet Business
  Amazon, PayPal, Google …
- Internet Protocols
  FTP, SMTP, IMAP, POP, WebDav …
- Secure Connectivity
  SSH, SFTP, SSL, Certificates …
- Encryption & Certificates
  X.509, OpenPGP, SHA, S/MIME …
- Network Management
  SNMP, MIB, LDAP, Monitoring …
- Compression
  Zip, Gzip, Jar, AES, 7Zip …

Our **Red Carpet Subscription** includes all product lines + updates for one year.

The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to any application, on any platform, anywhere, and you do the rest. For more than 20 years, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

connectivity powered by

To learn more please visit our website → www.nsoftware.com

Microsoft .NET | Java | C++ | Delphi | C++ Builder | ActiveX | Linux/Unix | macOS | Android | iOS | PHP | Xamarin | Node.js

Also in version 2 there's a shared messaging project that abstracts the details of the message queue, and a console application that listens for the event published by the Web app and saves the user's data to the database. The persistence code in the console app is directly lifted from the version 1 code in the Web app, so the implementation is the same but the design of the feature has been modernized.

The new version of the application is a distributed solution with many working parts, as shown in **Figure 2**.

The Docker platform treats distributed applications as first-class citizens.

There are dependencies between the components, and they need to be started in the correct order for the solution to work properly. This is one of the problems of orchestrating an application running across many containers, but the Docker platform deals with that by treating distributed applications as first-class citizens.

Orchestrating Applications with Docker Compose

Docker Compose is the part of the Docker platform that focuses on distributed applications. You define all the parts of your application as services in a simple text file, including the dependencies between them and any configuration values they need. This is part of the Docker Compose file for version 2, showing just the configuration for the Web app:

```
product-launch-web:
  image: sixeyed/msdn-web-app:v2
  ports:
    - "80:80"
  depends_on:
    - sql-server
    - message-queue
  networks:
    - app-net
```

Here, I'm specifying the version of the image to use for my Web application. I publish port 80 and then I explicitly state that the Web app depends on the SQL Server and message queue containers. To reach these containers, the Web container needs to be in the same virtual Docker network, so all the containers in the Docker Compose file are joined to the same virtual network, called app-net.

Elsewhere in the Docker Compose file I define a service for SQL Server, using the Microsoft image on Docker Hub, and I'm using the NATS messaging system for my message queue service, which is a high-performance open source message queue. NATS is available as an official image on Docker Hub. The final service is for the message handler, which is a .NET console application packaged as a Docker image, using a simple Dockerfile.

Now I can run the application using the Docker Compose command line:

```
docker-compose up -d
```

Then Docker Compose will start containers for each of the components in the right order, giving me a working solution from a single command. Anyone with access to the Docker images and the Docker Compose file can run the application and it will behave in the same way—on a Windows 10 laptop, or on a Windows Server 2016 machine running in the datacenter or on Azure.

For version 2, I made a small change to the application code to move a feature implementation from one component to another. The end-user behavior is the same, but now the solution is easily scalable, because the Web tier is decoupled from the data tier, and the message queue takes care of any spikes in traffic. The new design is easy to extend, as well, as I've introduced an event-driven architecture, so I can trigger new behavior by plugging in to the existing event messages.

Adding Self-Service Analytics

For my sample app, I'm going to make one more change to show how much you can do with the Docker platform, with very little effort. The app currently uses SQL Server as a transactional database, and I'm going to add a second data store as a reporting database. This will let me keep reporting concerns separate from transactional concerns, and also gives me free choice of the technology stack.

In version 3 of the sample code, I've added a new .NET console app that listens for the same event messages published by the Web application. When both console apps are running, the NATS message queue will ensure they both get a copy of all events. The new console app receives the events and saves the user data in Elasticsearch, an open source document store you can run in a Windows Docker container. Elasticsearch is a good choice here because it scales well, so I can cluster it across multiple containers for redundancy, and because it has an excellent user-facing front end available called Kibana.

I haven't made any changes to the Web application or the SQL Server message handler from version 2, so in my Docker Compose file I just add new services for Elasticsearch and Kibana, and for the new message handler that writes documents to the Elasticsearch index:

```
index-prospect-handler:
  image: sixeyed/msdn-index-handler:v3
  depends_on:
    - elasticsearch
    - message-queue
  networks:
    - app-net
```

Docker Compose can make incremental upgrades to an application, and it won't replace running containers if their definition matches the service in the Docker Compose file. In version 3 of the sample application, there are new services but no changes to the existing services, so when I run docker-compose up –d, Docker will run new containers for Elasticsearch, Kibana and the index message handler, but leave the others running as is—which makes for a very safe upgrade process where you can add features without taking the application offline.

This application prefers convention over configuration, so the host names for dependencies like Elasticsearch are set as defaults in the app, and I just need to make sure the container names match in the Docker Compose setup.

When the new containers have started, I can use "docker inspect" to get the IP address of the Kibana container, and browse to port 5601 on that address. Kibana has a very simple interface and in a few minutes I can build a dashboard that shows the key metrics for people signing up with their details, as shown in **Figure 3**.

Data Quality Made Easy. Your Data, Your Way.

NAME

Microsoft

Melissa provides the full spectrum of data quality to ensure you have data you can trust.

We profile, standardize, verify, match and enrich global **People Data** – name, address, email, phone, and more.

Our data quality solutions are available on-premises and in the Cloud – fast, easy to use, and powerful developer tools and plugins for the **Microsoft® Product Ecosystem**.

Microsoft® SQL Server® Integration Services · Microsoft Dynamics® CRM · Microsoft® .NET · Microsoft Azure · Microsoft® Excel

Start Your Free Trial

www.Melissa.com/msft-pd

Melissa Data is Now Melissa.
Why the change?
See for Yourself at the New www.Melissa.com

melissa™
1-800-MELISSA

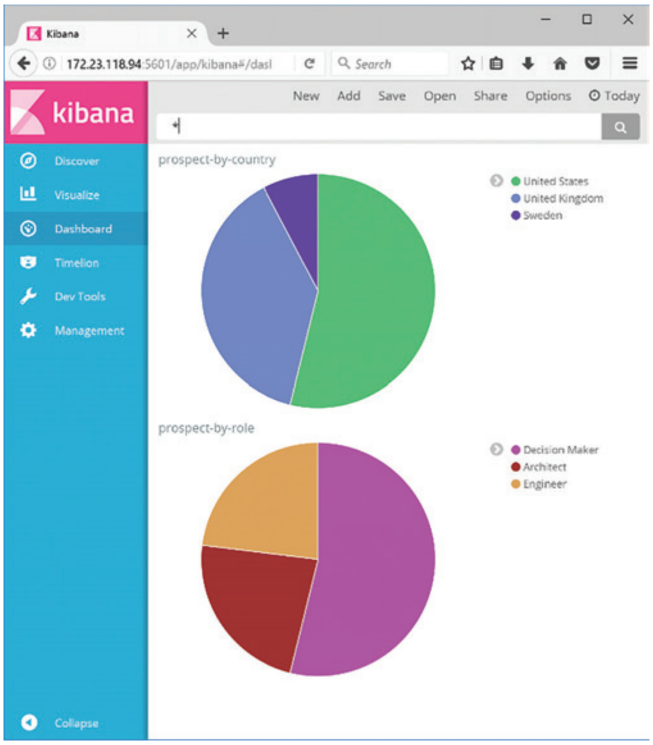


Figure 3 **A Kibana Dashboard**

Power users will quickly find their way around Kibana, and they'll be able to make their own visualizations and dashboards without needing to involve IT. Without any downtime I've added self-service analytics to the application. The core of that feature comes from enterprise-grade open source software I've pulled from Docker Hub into my solution. The custom component to feed data into the document store is a simple .NET console application, with around 100 lines of code. The Docker platform takes care of plugging the components together.

Running Dockerized Solutions on Azure

Another great benefit of Docker is portability. Applications packaged into Docker images will run the exact same way on any host. The final application for this article uses the Windows Server and SQL Server images owned by Microsoft; the NATS image curated by Docker; and my own custom images. All those images are published on the Docker Hub, so any Windows 10 or Windows Server 2016 machine can pull the images and run containers from them.

Now my app is ready for testing, and deploying it to a shared environment on Azure is simple. I've created a virtual machine (VM) in Azure using the Windows Server 2016 Datacenter with Containers option. That VM image comes with Docker installed and configured, and the base Docker images for Windows Server Core and Nano Server already downloaded. One item not included in the VM is Docker Compose, which I downloaded from the GitHub release page.

The images used in my Docker Compose file are all in public repositories on Docker Hub. For a private software stack, you won't want all your images publicly available. You can still use Docker Hub and keep images in private repositories, or you could use an alternative hosted registry like Azure Container Registry. Inside your own datacenter you can use an on-premises option, such as Docker Trusted Registry.

Because all my images are public, I just need to copy the Docker Compose file onto the Azure VM and run docker-compose up –d. Docker will pull all the images from the Hub, and run containers from them in the correct order. Each component uses conventions to access the other components, and those conventions are baked into the Docker Compose file, so even on a completely fresh environment, the solution will just start and run as expected.

If you've worked on enterprise software releases, where setting up a new environment is a manual, risky and slow process, you'll see how much benefit is to be had from Windows Server 2016 and the Docker platform. The key artifacts in a Docker solution—the Dockerfile and the Docker Compose file—are simple, unambiguous replacements for manual deployment documents. They encourage automation and they make it straightforward to build, ship and run a solution in a consistent way on any machine.

Next Steps

If you're keen to try Docker for yourself, the Image2Docker PowerShell module is a great place to start; it can build a Dockerfile for you and jump-start the learning process. There are some great, free, self-paced courses on training.docker.com, which provisions an environment for you. Then, when you're ready to move on, check out the Docker Labs on GitHub, which has plenty of Windows container walk-throughs.

In a Docker solution the Dockerfile and the Docker Compose file are simple, unambiguous replacements for manual deployment documents.

There are also Docker MeetUps all over the world where you can hear practitioners and experts talk about all aspects of Docker. The big Docker conference is DockerCon, which is always a sell-out; this year it's running in Texas in April and in Copenhagen in October. Last, check out the Docker Captains—they're the Docker equivalent of Microsoft MVPs. They're constantly blogging, tweeting and speaking about all the cool things they're doing with Docker, and following them is a great way to keep a pulse on the technology.■

Elton Stoneman *is a seven-time Microsoft MVP and a Pluralsight author who works as a developer advocate at Docker. He has been architecting and delivering successful solutions with Microsoft technologies since 2000, most recently API and Big Data projects in Azure, and distributed applications with Docker.*

Thanks to the following technical expert who reviewed this article: *Mark Heath*

RD

Rider

New .NET IDE

Cross-platform.
Killer code analysis.
Great for refactoring.

From the makers of ReSharper, IntelliJ IDEA, and WebStorm.

Learn more
and get early access
jetbrains.com/rider

JET
BRAINS

The New Azure App Service Environment

Christina Compy

The Azure App Service Environment (ASE) is a Premium feature offering of the Azure App Service. It gives a single-tenant instance of the Azure App Service that runs right in your own Azure virtual network (VNet), providing network isolation and improved scaling capabilities. While the original feature gave customers what they were looking for in terms of network control and isolation, it was not as "Platform as a Service (PaaS) like" as the normal App Service. This caused confusion among customers, who had some trouble managing the system. With the newly relaunched ASE, however, things now work the same as the multi-tenant App Service.

History

The Azure App Service is a multi-tenant application hosting service. If you want to run your HTTP listening applications in a PaaS service, the App Service is a very quick and easy way to go and has many developer-supporting features. You can do things like integrate with continuous integration (CI) systems, scale your apps out instantly with a flick of the mouse and much more. There are limits to the service, though, that blocked certain use cases.

This article discusses:

- Azure App Service and the App Service Environment (ASE)
- Customer concerns with the first version of the ASE
- Improvements in the newly relaunched ASE

Technologies discussed:

Microsoft Azure App Services and App Service Environment

The use cases that couldn't be met in the multi-tenant App Service largely centered around scale and app isolation. While you can scale your apps easily in the multi-tenant App Service, there are limits based on the price plan. The greatest number of instances you can scale an app to in the multi-tenant App Service is 20.

With respect to isolation, there's no way to lock down access to your apps in the multi-tenant App Service at a network level. The App Service has two features to access resources in other networks, Azure Virtual Network (VNet) Integration and Hybrid Connections, but has nothing that can lock apps down at a network level and no way to host completely Internet-isolated apps in the App Service. This means you couldn't host a line-of-business (LOB) application that you wanted available only on a private IP address on the multi-tenant App Service.

To resolve the scaling and isolation limitations, we provided the Premium ASE feature in 2015. It's an instance of the Azure App Service that runs in a customer's VNet, running the same code as the multi-tenant App Service but with some changes to deployment to use fewer resources.

With the first version of the ASE you could scale up to 50 instances and use larger dedicated workers. The ASE is capable of hosting Web apps, mobile apps, API apps and Functions. Because the ASE runs in a subnet in the customer's VNet, the apps in the ASE have easy access to resources that are available in the VNet itself or across ExpressRoute or site-to-site VPN connections. Also, as shown in **Figure 1**, because the ASE is in the customer's subnet, it can restrict access to its apps at a network level using network security groups (NSGs).

Among the benefits of this deployment model is a static IP address that can be used for both the inbound and outbound IP

DOMAINS | MAIL | **HOSTING** | eCOMMERCE | SERVERS

NEW!

NEXT LEVEL HOSTING!

NEW: Full performance scalability! Customize your web project's performance level yourself, according to your own needs. For example, with increasing traffic from newsletters, blogs, adding an online shop or resource intensive applications, you can easily adjust your site's performance level to match.

✓ Next Level Performance

- **NEW:** 2.5 GB RAM
- **NEW:** Simple scalability with up to 19 GB RAM in just a few clicks
- Performance monitoring in the 1&1 Control Panel
- Change your performance level with no downtime

✓ Next Level Speed

- **NEW:** Faster load times with HTTP/2
- **NEW:** PHP 7.1 + OPcache
- 1&1 CDN
- 24/7 Support

✓ Next Level Security

- Including SSL certificate
- DDoS protection
- Geo-redundancy

Starting at
~~$7.99~~ **$0.99 /month***

1 TRIAL TRY FOR 30 DAYS ★★★★★

1 MONTH FLEXIBLE PAYMENT OPTIONS ★★★★★

1 CALL SPEAK WITH AN EXPERT 24/7 ★★★★★

1 (844) 296 - 2559

1&1 ®

1and1.com

*Discount based on 12 month contract for Basic package at $0.99/month, then $7.99/month. © 1&1 Internet Inc. 2017 All rights reserved. 1&1 and the 1&1 logo are trademarks of 1&1 Internet SE, all other trademarks are the property of their respective owners. 1&1 Internet Inc, 701 Lee Road, Chesterbrook, PA 19087.

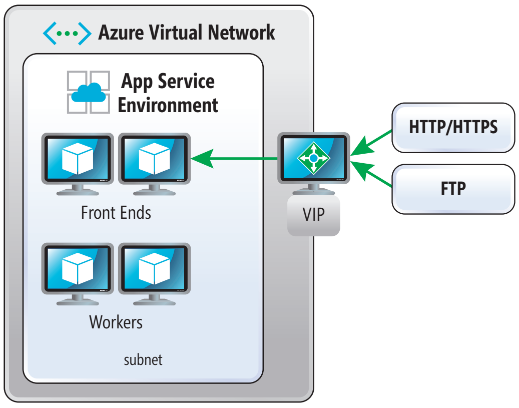


Figure 1 **App Service Environment High-Level Networking Model**

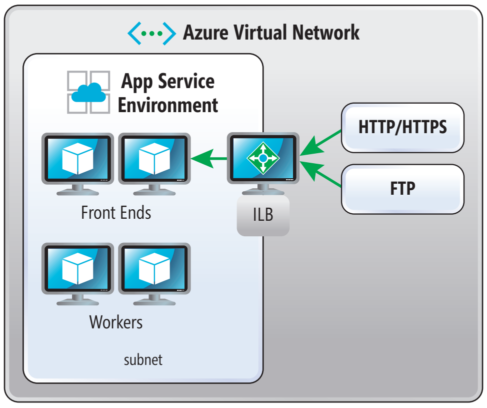


Figure 2 **App Service Environment High-Level Networking with an Internal Load Balancer**

address for the apps in the ASE. The nature of the multi-tenant app service is that the inbound and outbound addresses are shared by multiple tenants. While it is possible to set up IP SSL for an app and get an IP address assigned to that app, there is no way to lock down the outbound address.

The nature of the multi-tenant app service is that the inbound and outbound addresses are shared by multiple tenants.

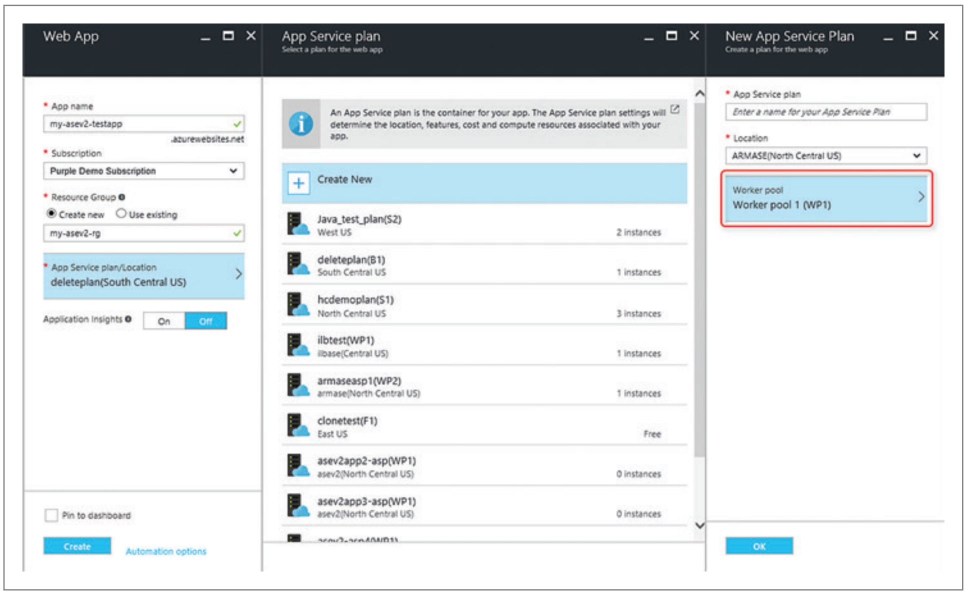
Hosting the ASE in a VNet is a great first start, but it still didn't completely solve the isolation problem when the ASE was first released. The ASE still needed a public virtual IP (VIP) for HTTP/S and publishing access. It also deployed only in classic VNets, which was a problem for many customers. To solve those problems, support was added in June 2016 for Resource Manager VNets and also for internal load balancers (ILBs), as shown in **Figure 2**.

The addition of ILB support meant that customers could now host intranet sites in the cloud. You could take an LOB application that you didn't want to be Internet-accessible and deploy it into your ILB-enabled ASE. The ILB sits on one of the VNet IP addresses, so it's accessible only from within the VNet or from hosts that have access to the VNet over a VPN.

The ILB-enabled ASE opened the door to other possibilities, such as Web application firewall (WAF)-fronted applications and two-tier applications. For WAF-fronted ASE applications, a customer could use a WAF virtual device to act as the Internet endpoint for its ILB ASE-hosted apps, which adds an additional security layer for Internet-accessible apps. In a two-tier application, the Web-accessible app could be hosted in either the multi-tenant app service or from another ASE, and the back-end-secured API apps could then be hosted in the ILB ASE. If you used the multi-tenant App Service for such a purpose, you'd then use the VNet Integration feature to securely access your API apps.

When the ASE was originally designed and planned, the assumption was that it would cater to IT professionals who'd want to control this personal deployment of the App Service as if it was a system they ran in their own datacenters. With that in mind, ASE was designed to be flexible. An ASE has two role types to manage: the front ends that act as the HTTP endpoints for applications and the workers that host the apps. You can scale out the quantity of either, as well as change the size of the virtual machine (VM) used for that role type.

There were certain consequences to thinking this way—the ASE roles were treated as resources that system administrators would independently manage, but it turned out that customers didn't want to be or have system administrators for their cloud services. They wanted the ASE to remain as easy to use as the multi-tenant App Service. Having to manage the resource pools and their apps was too confusing and affected feature adoption.



* App name: my-asev2-testapp .azurewebsites.net
* Subscription: Purple Demo Subscription
* Resource Group: (● Create new) ○ Use existing — my-asev2-rg
* App Service plan/Location: deleteplan(South Central US)

Application Insights: On [✓ Off]

An App Service plan is the container for your app. The App Service plan settings will determine the location, features, cost and compute resources associated with your app.

Java_test_plan(S2)	West US	2 instances
deleteplan(B1)	South Central US	1 instances
hcdemoplan(S1)	North Central US	3 instances
ilbtest(WP1)	ilbase(Central US)	1 instances
armaseasp1(WP2)	armase(North Central US)	1 instances
clonetest(F1)	East US	Free
asev2app2-asp(WP1)	asev2(North Central US)	0 instances
asev2app3-asp(WP1)	asev2(North Central US)	0 instances

* App Service plan: Enter a name for your App Service Plan
* Location: ARMASE(North Central US)

Worker pool: Worker pool 1 (WP1)

Figure 3 **Creating an App Service Plan in ASEv1**

FREE TRIAL

DocuVieware

Universal HTML5 Viewer & Document Management Kit

supports nearly 100 file formats

zero-footprint solution

super-easy integration

fast & crystal clear rendering

fully customizable UI
look & feel

mobile devices optimization

annotations, thumbnails
bookmarks & text search

HOT FEATURES IN THE NEW MAJOR VERSION

- Office Open XML (.docx) support
- TWAIN acquisition
- PDF form fields interactions
- Custom snap-ins
- Higher speed and lower memory usage

Download the **Free Trial** and check the **Online Demos** at www.docuvieware.com

Powered by GdPicture.NET

DocuVieware is an ORPALIS imaging technologies product

The New ASE

After the initial version of the ASE (which I'll refer to as ASEv1) was released, there was substantial feedback from customers who tried it out and found that it didn't fit their business needs for one reason or another. The primary reasons they gave concerned:

- The complexity of managing the ASEv1 roles, as well as their apps, was aggravating and non-intuitive.
- Adding more capacity to the ASE took too long. Because ASEv1 was built to be run by system administrators for their tenants, the concern had not been with how fast the roles were provisioned. The reality was that when ASEv1 was used, the person who scaled out the ASE roles and the one who deployed the app were typically one and the same and the delay was a problem.
- The system management model forced customers to be far more aware of the ASE architecture and behavior than they wanted to be.

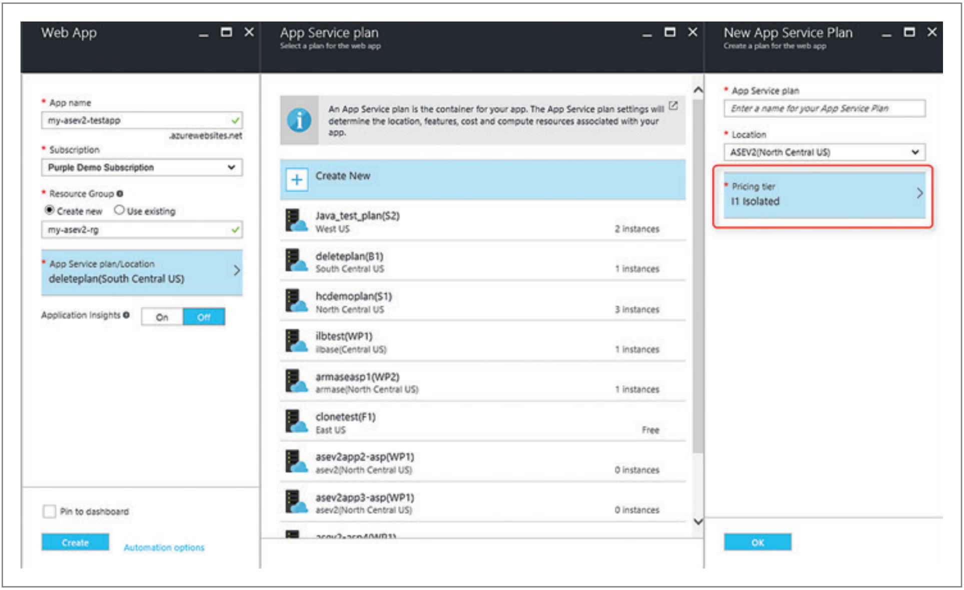
This brings us to the new version of the ASE, which I'll call ASEv2. The team took the feedback to heart and for ASEv2 we focused on making the UX the same as it was in the multi-tenant App Service, without losing the benefits that ASEv1 provided.

Creating an App Service Plan The App Service plan (ASP) is the scaling container that holds all apps. All apps are in ASPs and when you scale the ASP, you're also scaling all of the apps in the ASP. This is true for the multi-tenant App Service and for the ASE. This means that to create an app you need to either choose or create an ASP. To create an ASP in ASEv1 you needed to pick an ASE as your location and then select a worker pool, as shown in **Figure 3**. If the worker pool you wanted to deploy into didn't have enough capacity, you'd have to add more workers to it before you could create your ASP in it.

With ASEv2, when you create an ASP you still select the ASE as your location, but instead of picking a worker pool, you use the pricing cards just as you do outside of the ASE. There are no more worker pools to manage. When you create or scale your ASP, the necessary workers are automatically added.

ASEv2 includes upgraded VMs that are used to host your apps. The workers for ASEv2 are built on the Dv2-series VMs and outperform the workers used in the multi-tenant app service. To distinguish between ASPs that are in an ASE and those in the multi-tenant service, a new pricing SKU was created. The name of this SKU is Isolated, as shown in **Figure 4**. When you pick an Isolated SKU it means you want the associated ASP to be created in an ASEv2.

Creating an ASE One of the other issues that hindered ASE adoption was its lack of visibility. Many customers didn't even know that the ASE feature existed. To create an ASE, you had to look for the ASE creation flow, which was completely separate from app creation. In ASEv1 customers had to add workers to their worker pools in order to create ASPs. Now that workers are added automatically when ASPs are created or scaled,



Web App

* App name: my-asev2-testapp .azurewebsites.net
* Subscription: Purple Demo Subscription
* Resource Group: (●) Create new ( ) Use existing — my-asev2-rg
* App Service plan/Location: deleteplan(South Central US)
Application Insights: On [✓ Off]
Pin to dashboard

App Service plan — Select a plan for the web app

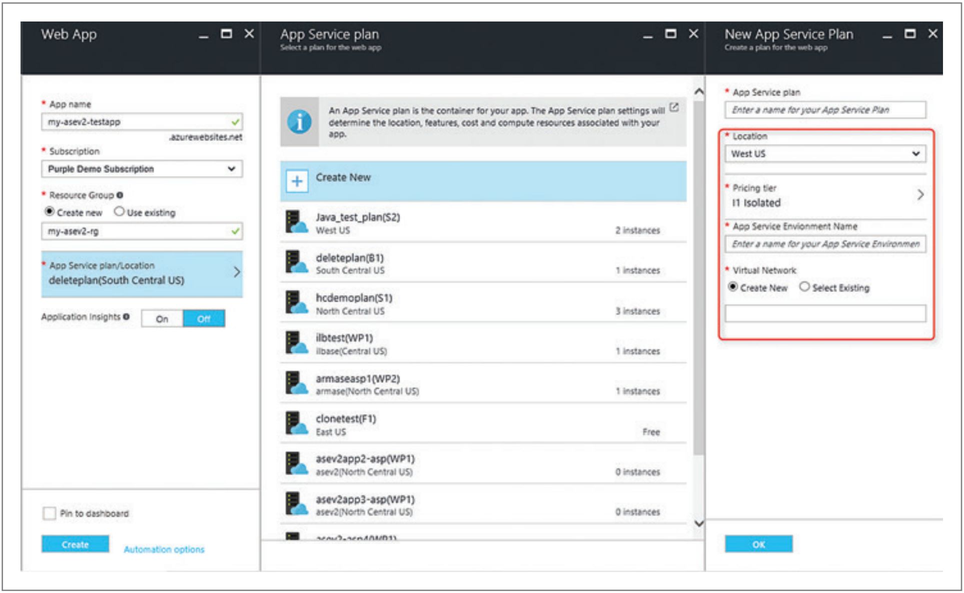
An App Service plan is the container for your app. The App Service plan settings will determine the location, features, cost and compute resources associated with your app.

Java_test_plan(S2)	West US	2 instances
deleteplan(B1)	South Central US	1 instances
hcdemoplan(S1)	North Central US	3 instances
ilbtest(WP1)	ilbase(Central US)	1 instances
armaseasp1(WP2)	armase(North Central US)	1 instances
clonetest(F1)	East US	Free
asev2app2-asp(WP1)	asev2(North Central US)	0 instances
asev2app3-asp(WP1)	asev2(North Central US)	0 instances

New App Service Plan — Create a plan for the web app

* App Service plan: Enter a name for your App Service Plan
* Location: ASEV2(North Central US)
* Pricing tier: I1 Isolated

Figure 4 **Creating an App Service Plan in ASEv2**



Web App

* App name: my-asev2-testapp .azurewebsites.net
* Subscription: Purple Demo Subscription
* Resource Group: (●) Create new ( ) Use existing — my-asev2-rg
* App Service plan/Location: deleteplan(South Central US)
Application Insights: On [✓ Off]
Pin to dashboard

App Service plan — Select a plan for the web app

An App Service plan is the container for your app. The App Service plan settings will determine the location, features, cost and compute resources associated with your app.

Java_test_plan(S2)	West US	2 instances
deleteplan(B1)	South Central US	1 instances
hcdemoplan(S1)	North Central US	3 instances
ilbtest(WP1)	ilbase(Central US)	1 instances
armaseasp1(WP2)	armase(North Central US)	1 instances
clonetest(F1)	East US	Free
asev2app2-asp(WP1)	asev2(North Central US)	0 instances
asev2app3-asp(WP1)	asev2(North Central US)	0 instances

New App Service Plan — Create a plan for the web app

* App Service plan: Enter a name for your App Service Plan
* Location: West US
* Pricing tier: I1 Isolated
* App Service Envionment Name: Enter a name for your App Service Environmen
* Virtual Network: (●) Create New ( ) Select Existing

Figure 5 **Creating an App Service Environment from the App Service Plan Creation Flow**

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

AUSTIN, TX
MAY 15-18, 2017
HYATT REGENCY

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

- Visual Studio / .NET
- JavaScript / HTML5
- Angular
- Native Mobile & Xamarin
- Software Practices
- Database and Analytics
- ASP.NET Core
- Web API
- ALM / DevOps
- Cloud Computing
- UWP
- Unit Testing

➤ **NEW!** Post-Con, Hands-On Labs, Friday, May 19. ONLY **$645** through April 12! SPACE IS LIMITED!

Register by April 12 and Save $200!*

Use promo code VSLAPRTI

*Available on 4 day packages; some restrictions apply.

REGISTER NOW

EVENT PARTNER Microsoft

GOLD SPONSOR Syncfusion

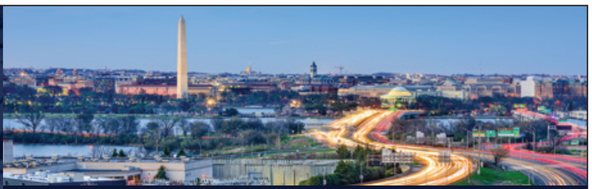
SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA YOUR GROWTH. OUR BUSINESS.

vslive.com/austin

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

WASH, DC
JUNE 12-15, 2017
MARRIOTT MARQUIS



INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

- Visual Studio / .NET
- JavaScript / HTML5
- Angular
- Native Mobile & Xamarin
- Software Practices
- Database and Analytics
- ASP.NET Core
- Web API
- ALM / DevOps
- Cloud Computing
- UWP
- Unit Testing

Register by April 21 and Save $300!*

Use promo code VSLAPRTI
*Available on 4 day packages; some restrictions apply.

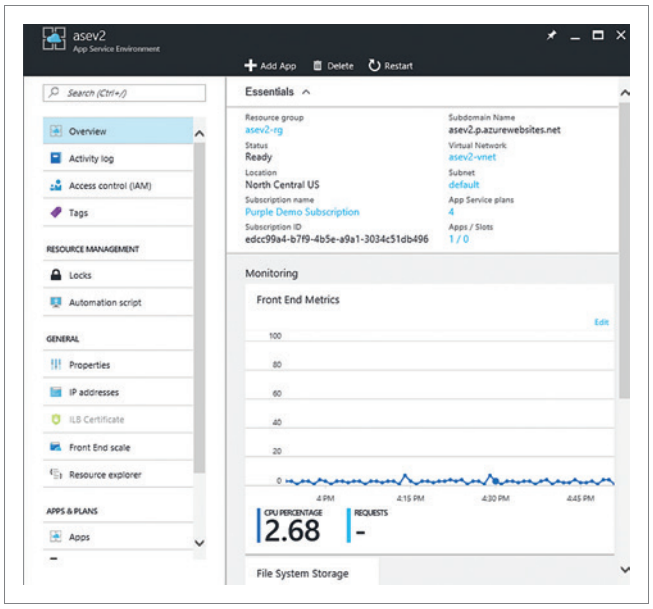
REGISTER NOW

EVENT PARTNER
Microsoft

SUPPORTED BY
Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY
1105 MEDIA INC. YOUR GROWTH. OUR BUSINESS.

vslive.com/dc



Resource group
asev2-rg
Status
Ready
Location
North Central US
Subscription name
Purple Demo Subscription
Subscription ID
edcc99a4-b7f9-4b5e-a9a1-3034c51db496
Subdomain Name
asev2.p.azurewebsites.net
Virtual Network
asev2-vnet
Subnet
default
App Service plans
4
Apps / Slots
1 / 0

Figure 6 **App Service Environment Version 2 Portal Page**

the ASEv2 creation experience can be placed squarely in the ASP creation flow, as shown in **Figure 5**.

To create a new ASEv2 during the ASP creation experience, you simply select a non-ASE region and then select one of the new Isolated SKU cards. When you do this, the ASE creation UI is displayed, which enables you to create a brand new ASEv2 in either a new or pre-existing VNet.

Time to scale The new ASP creation flow became possible only because the process for provisioning new workers was accelerated. ASEv2 automatically provisions new workers for your ASPs when you create or scale an ASP. The only way to make this a reasonable customer experience was to reduce the time required to create and scale out. To make this work, as much as possible is preloaded onto the VHDs used for provisioning the role instances, minimizing supplementary required reboots. Moving to the Dv2 workers was also helpful as they have faster cores and use SSDs. Both of those practices make install and reboots faster.

The new ASP creation flow became possible only because the process for provisioning new workers was accelerated.

System Management In ASEv1 the customer had to manage the front ends, workers and the update domain workers. The front-end roles handle HTTP traffic and send traffic to the workers. The workers are the VMs that host your apps. The update domain workers act as standby hosts in case of upgrades or worker failures. With ASEv1 the customer had to know how these components all worked together and scale the resource pools appropriately. When workers had to be scaled out to handle more ASP instances, users had to add more front ends and update domain workers.

ASEv2, in contrast, hides away the infrastructure. Now users simply scale out their App Service plans and the infrastructure is added as needed. When an ASP needs more workers, the workers are added. Front ends and update domain workers are added automatically as the quantity of workers is scaled out. If customers have unusual needs that require more aggressive front-end scaling, they can change the rate at which front ends are added to their ASE.

As you can see in the ASEv2 portal page in **Figure 6**, things are far simpler now. There's no longer a need for the worker pool or front-end UI pages. With all scaling now automatic, there are no more scale or autoscale controls. And because the IP addresses used by the ASE are pretty important to know about, the UI consolidates that information.

The release of ASEv2 is by no means the end of the ASE feature development efforts. There will continue to be a steady stream of improvements, but they will not impact the UX to the extent the changes made with ASEv2 did.

Additional Benefits Due to the changes made to the system architecture, ASEv2 has a few additional benefits over ASEv1. With ASEv1, the maximum default scale was 50 workers. There were a number of system-architecture reasons why that limit was set, but these issues were addressed in creating the new ASEv2 experience. With ASEv2 the maximum default scale is now 100, which means you can have up to 100 ASP instances hosted in ASEv2. This can be anything from 100 instances of an ASP to 100 individual ASPs, with anything in between.

Moreover, ASEv2 now uses Dv2-based dedicated workers. These new dedicated workers are much faster than the A series VMs on which ASEv1 depended. They have faster CPUs, which improves throughput, and SSDs, which improves file-access performance. As in the multi-tenant app service, the choices for dedicated workers when creating an ASP are single core, dual core or quad core. The new ASE dedicated workers, however, have double the memory of their multi-tenant counterparts and come with 3.5GB, 7GB or 14GB of RAM, respectively.

Wrapping Up

ASEv1 was a great first step toward enabling customers to have network isolation for their App Service-hosted applications. ASEv2 built on that experience a far more PaaS-like capability that's not just easier to use, but is also much more powerful.

All of the changes that have been noted here for the ASE have been vetted by a large number of MVPs and customers. Even before development started, the team wanted to validate its approach with people who had already tried using an ASE. As a result of this input-heavy approach, we are confident that the new ASE experience will be considered a substantial improvement and look forward to its success in the field. ■

Christina Compy *is originally an aerospace engineer who worked on the Hubble Space Telescope and has been working in the software industry for more than 20 years. She has been a program manager at Microsoft since 2013 and works on enterprise-focused capabilities.*

Thanks to the following Microsoft technical expert for reviewing this article: *Stefan Shackow*

Develop Hosted Web Apps for UWP

Sagar Bhanudas Joshi

Many developers and companies create Web interfaces for their products and services for easy discoverability and access. Another platform is the apps platform. Native apps provide richer UX and functionality than Web apps. Now, Project Westminster provides developers a way to convert modern Web sites into native apps.

The Project Westminster bridge enables Web developers to bring their responsive Web applications to the Universal Windows Platform (UWP) by leveraging existing code (see the "Project Westminster in a Nutshell" Windows Developer blog post at bit.ly/2jyhVQo). The idea of this "bridge" is to help reuse existing Web site code and add a layer for UWP-specific code to form the integration points of the Web app with an underlying Windows platform. The blog post discusses a few coding practices to ensure consistent UX.

This article discusses:

- File downloads scenario in hosted Web apps
- Session management and the back button
- Voice commands and Live Tiles integration

Technologies discussed:

Project Westminster, Windows Runtime, Hosted Web Apps

Code download available at:

bit.ly/2k5FlJh

How can a Web developer integrate a modern Web application to work with, let's say, Cortana? The answer is through the common denominator, JavaScript. Windows exposes the app framework functionality (Windows Runtime APIs) through JavaScript and the Windows namespace. The resulting apps are referred to as Hosted Web Apps.

In this article, I'll share what I've learned in working with Independent Software Vendors (ISVs) and partners to port their apps to the UWP through the Project Westminster tool. The focus here is to share details on how to best align Web apps for delivering the best UX while running as a platform app.

Getting Started

To get started with the Project Westminster tool, you first need to convert your Web site to the UWP (see the Channel 9 video, "Creating Hosted Web Apps with Project Westminster," at bit.ly/2jp4srs).

As a preliminary test—to be sure the Web site renders as expected and the presentation looks consistent—you should test it on the Microsoft Edge browser. This will help identify and fix any rendering issues before coding integrations for Windows functionality begin.

Most Web sites have a few common features, which enable users to either complete a specific task (like filling out forms) or take away information for their reference (like downloading a manual). In such scenarios, it's critical that these functionalities remain intact and the newly generated hosted Web app's experience is consistent with the original Web site. Some of these scenarios might

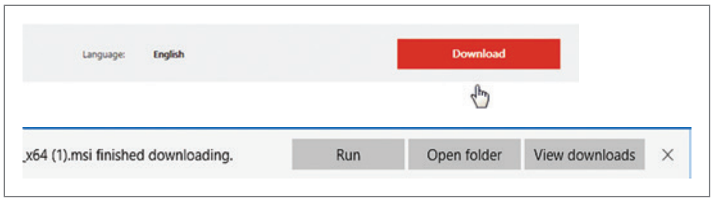


Figure 1 **Default Download Experience**

need to be tested, reviewed and re-factored through code to align with users' expectations. A few key scenarios are considered in the next sections to demonstrate different classes or objects available to developers while working with the Project Westminster tool and related code workflows. While porting the app to the UWP, it's a good idea to consider integrating a few native features of the platform for a richer UX and an enhanced app experience. Here are some of the commonly implemented features by app developers:

- Contacts
- Cortana integration
- Live tiles
- Toast notifications
- Camera and microphone
- Photos library
- Rich graphics and Media Windows Runtime Stack
- Sharing content with other apps

In this article, the focus is on integrating UWP features, such as Cortana and Live Tiles, for example, which help with activation of apps through voice-based commands and information delivery to the user. Thus, it enhances the overall UX with support of the UWP apps infrastructure. The latest Windows 10 features for developers Web page (bit.ly/2iKufs6) provides a quick overview of more integration opportunities. The Web page, which is being converted to an app, provides information on the following:

1. Web application functionality features
   a. File downloads
   b. Session management or single sign-on
   c. Can go to previous page through back button in a stately way
2. UWP integration features
   a. Live Tiles
   b. Cortana

These features need to be considered while porting and refactoring to deliver a predictable experience for the app.

File Download Scenario

Most Web applications today enable file downloads for a variety of content. As shown in **Figure 1**, the default experience of file download in the browser is like clicking a button or hyperlink; the browser begins to download it and also saves the file to (mostly) rootdir:\users\username\Downloads.

This happens partly because the browser is a native Win32 application running with full trust privileges and writes the file directly to the Downloads folder.

Now, consider the same scenario when the Web site is running in the context of an app (WWAHost.exe, to be specific) and the download button is clicked. What next? Most likely, nothing will happen and it will appear that the code simply doesn't work. It might appear that the button isn't responding or, perhaps, the file download has started, but where is it being saved?

WWAHost.exe is an app container for Web sites, which has a subset of features, compared to the browser. This subset of features

Figure 2 **File Download JavaScript Code (Larger Files)**

```
(function() {


  if (typeof Windows !== 'undefined' &&
    typeof Windows.UI !== 'undefined' &&
    typeof Windows.ApplicationModel !== 'undefined') {


  function WinAppSaveFileBGDownloader() {
    // This condition is only true when running inside an app.
    // The else condition is effective when running inside browsers.

    // This function uses the Background Downloader class to download a file.
    // This is useful when downloading a file more than 50MB.

    // Downloads continue even after the app is suspended/closed.


    if (typeof Windows !== 'undefined' &&
      typeof Windows.UI !== 'undefined' &&
      typeof Windows.ApplicationModel !== 'undefined') {

        var fileSavePicker = new Windows.Storage.Pickers.FileSavePicker();

        // Example: You can replace the words EXTENSION and ext with the word PNG.

        fileSavePicker.fileTypeChoices.insert(
          "EXTENSION file", [".ext"]);
            // Insert appropriate file format through code.
        fileSavePicker.defaultFileExtension = ".ext";
          // Extension of the file being saved.
        fileSavePicker.suggestedFileName = "file.ext";
          // Name of the file to be downloaded.
        fileSavePicker.settingsIdentifier = "fileSavePicker1";

        var fileUri =
          new Windows.Foundation.Uri("<URL of the file being downloaded>");


        fileSavePicker.pickSaveFileAsync().then(function (fileToSave) {
          var downloader =
            new Windows.Networking.BackgroundTransfer.BackgroundDownloader();
          var download = downloader.createDownload(
            fileUri,
            fileToSave);

          download.startAsync().then(function (download) {
            // Any post processing.

            console.log("Done");


          });

        });
      }



      else {
        // Use the normal download functionality already implemented for browsers,
        // something like <a href="<URL>" />.
      }

    }

}

})();
```

TEXT CONTROL

THE TOP 5 EMR/EHR HEALTHCARE SOFTWARE VENDORS INTEGRATED TEXT CONTROL REPORTING

The Text Control Reporting Framework combines powerful reporting features with an easy-to-use, MS Word compatible word processor.

Our award-winning developer libraries are completely independent from MS Word or any other third-party application and can be seamlessly integrated into your business application.

www.textcontrol.com

Visual Studio
Partner
WE'RE
CHANGING
THE WAY
YOU LOOK AT
REPORTING
Microsoft .NET
WINDOWS FORMS
WPF
ASP.NET MVC
ASP.NET AJAX
CLOUD WEB API

includes the rendering capability (mostly presentation through HTML) and Standard script execution.

What you're working with now is an app. In the apps world, for developers, the file download should be explicitly coded for—otherwise, the remote file is just a URL and it's unclear what the app/container would do with a remote file URL (I can discuss what might be happening under the hood through specialized debugging tools, however, I won't go into that for now).

To handle these kinds of scenarios, you need to invoke the appropriate UWP APIs. These APIs can co-exist with the code that makes those functionalities work when running the Web site using a browser. **Figure 2** shows the code involved.

You can write the code in **Figure 2** on the button click or wherever the file download functionality is expected. The code snippet uses the BackgroundDownloader class (bit.ly/2jQeuBw), which has lots of benefits, such as persisting downloads in background post-app suspension and the ability to download large files.

The code in **Figure 2** actually creates the same experience as a browser would, such as prompting the user to select a file location (fileSavePicker variable), initiating the download (downloader variable) and writing it to the selected location.

Alternatively, you can use the code in **Figure 3** for smaller file downloads, which will run as long as the app is running:

The code in **Figure 3** uses the Windows.Web.Http.HttpClient to handle the file download operation (bit.ly/2k5iX2E). If you compare the code snippets in **Figure 2** and **Figure 3**, you'll notice that the latter requires a bit of advanced coding, which gives you more control over the file download. This way, you're also able to explore the multiple ways to save a stream to a file using a UWP app.

Also, in the case where the app also processes the files that were downloaded, it's best to use the FutureAccessList property to cache and access the location where the file was saved (bit.ly/2k5fXn6). This is important because the user can choose to save the downloaded files anywhere on the system (such as in D:\files\ or in C:\users\myuser\myfiles\) and the app container, being a sandbox process, might not be able to directly access the file system without user-initiated actions. This class can be useful to avoid extra steps for the user to open the same file again. To use this class, it's required to open the file using the FileOpenPicker at least once.

This should help streamline file download functionality in the hosted Web apps and deliver an intuitive UX. (**Tip:** Make sure to add the file URL domain to the ApplicationContentURI at bit.ly/2jsN1WS to avoid runtime exceptions.)

Figure 3 **File Download JavaScript Code (Smaller Files)**

```
// Use the Windows.Web.Http.HttpClient to download smaller files and
// files are needed to download when the app is running in foreground.

(function() {


  if (typeof Windows !== 'undefined' &&
    typeof Windows.UI !== 'undefined' &&
    typeof Windows.ApplicationModel !== 'undefined') {


  function WinAppSaveFileWinHTTP() {


    var uri = new Windows.Foundation.Uri("<URL of the file being downloaded>");

    var fileSavePicker = new Windows.Storage.Pickers.FileSavePicker();
    fileSavePicker.fileTypeChoices.insert("EXT file",
      [".ext"]); //insert appropriate file format through code.
    fileSavePicker.defaultFileExtension = ".ext";
      // Extension of the file being saved.
    fileSavePicker.suggestedFileName = "file.ext";
      // Name of the file to be downloaded. Needs to be replaced programmatically.
    fileSavePicker.settingsIdentifier = "fileSavePicker1";


    fileSavePicker.pickSaveFileAsync().then(function (fileToSave) {

      console.log(fileToSave);
      var httpClient = new Windows.Web.Http.HttpClient();

      httpClient.getAsync(uri).then(function (remoteFile) {

        remoteFile.content.readAsInputStreamAsync().then(
           function (stream) {

          fileToSave.openAsync(
            Windows.Storage.FileAccessMode.readWrite).then(
            function (outputStream) {

            Windows.Storage.Streams.RandomAccessStream.copyAndCloseAsync(
              stream, outputStream).then(function (progress, progress2) {

          // Monitor file download progress.
            console.log(progress);
            var temp = progress2;
          });
        });

        });
      });

    });

  }
}

})();
```

Session Management

It's common for most modern apps today to persist data between users' multiple sessions. As an example, imagine having to log into a frequently used app every time the app is launched. It would be time-consuming if the app is launched multiple times in a day. There are classes within the Windows Runtime framework available to the Web apps, which are converted to UWP apps.

Figure 4 **Cortana Commands XML**

```
<?xml version="1.0" encoding="utf-8"?>
<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.1">
  <CommandSet xml:lang="en-us" Name="AdventureWorksCommandSet_en-us">
    <CommandPrefix> Adventure Works, </CommandPrefix>
    <Example> Show trip to London </Example>

    <Command Name="showTripToDestination">
      <Example> Show trip to London </Example>
      <ListenFor RequireAppName=
        "BeforeOrAfterPhrase"> show trip to {destination} </ListenFor>
      <Feedback> Showing trip to {destination} </Feedback>
      <Navigate/>
    </Command>

    <PhraseList Label="destination">
      <Item> London </Item>
      <Item> Dallas </Item>
    </PhraseList>

  </CommandSet>
<!-- Other CommandSets for other languages -->
</VoiceCommands>
```

Figure 5 **Voice Commands Handler Code**

```
<!-- In HTML page :

<meta name="msapplication-cortanavcd" content="http://<URL>/vcd.xml" />

-->

(function () {


  if (typeof Windows !== 'undefined' &&
    typeof Windows.UI !== 'undefined' &&
    typeof Windows.ApplicationModel !== 'undefined') {


    Windows.UI.WebUI.WebUIApplication.addEventListener("activated", activatedEvent);

  }

  function activatedEvent (args) {

    var activation = Windows.ApplicationModel.Activation;
    // Check to see if the app was activated by a voice command.
    if (args.kind === activation.ActivationKind.voiceCommand) {

      // Get the speech recognition.
      var speechRecognitionResult = args.result;
      var textSpoken = speechRecognitionResult.text;
      // Determine the command type {search} defined in vcd.
      switch (textSpoken) {

         case "London":
           window.location.href =
             'https://<mywebsite.webapp.net>/Pages/Cities.html?value=London';
        break;

      case "Dallas":
        window.location.href =
          'https://<mywebsite.webapp.net>/Pages/Cities.html?value=Dallas';
        break;

                 ...
        <other cases>
                 ...
      }

    }
  }

})();
```

Consider a scenario in which certain parameters need to be persisted between sessions by the same user. For example, consider a string identifying the user and other miscellaneous information such as last session time and other cached items.

The right class to use in this scenario is the Windows.Storage.ApplicationData class. This class has many properties and methods, however, consider the localSettings property for this scenario.

The settings are stored in a key-value pair system. Look at the following sample code for the implementation:

```
function getSessionData()
{
var applicationData = Windows.Storage.ApplicationData.current;
var localSettings = applicationData.localSettings;

// Create a simple setting.
localSettings.values["userID"] = "user998-i889-27";

// Read data from a simple setting.
var value = localSettings.values["userID"];

}
```

Ideally, the code to write data into the settings can be written at checkpoints within the app where it's necessary to capture certain information and then be sent to a remote Web API or service. The code to read the setting can usually be written in the App_activated event handler where the app is launching and the information persisted from the previous app session can be accessed. Note that the name of each setting has a 255-character limit and each setting has a limit of 8K bytes in size.

Voice (Cortana) Integration Scenario

One of the scenarios that your Web site (now an app) can implement—post porting to the UWP—is to integrate Cortana, a voice-based interactive digital assistant on Windows devices.

Consider that the Web site is a travel-based portal and now, ported to a Windows 10 app, one of the use cases where Cortana can be integrated is to show details of a person's trip. The user can then issue commands such as, "Show trip to London" (or whatever destination), which needs to be configured by app developers, as shown in **Figure 4**.

Cortana can help achieve tasks with a lot of first-party apps (such as Calendar) and expose APIs to interface with custom apps. Basically, here's basically how Cortana works with apps:

1. Listens to commands as registered in the voicecommands.xml file.
2. Activates the relevant app, matching the commands spoken/typed in the Windows Search bar.
3. Passes the speech to/text commands to app as variables to let the app process the input from user.

Note: The file name is indicative. It can be named as desired with an XML extension. The Windows Developer article, "Using Cortana to Interact with Your Customers (10 by 10)," at bit.ly/2iGymdE provides a good overview of configuring commands in the XML file and its schema.

The XML code in **Figure 4** will help Cortana identify that the Adventure Works app needs to launch when issued commands such as the following:

```
'Adventure Works, Show trip to London'
'Adventure Works, Show trip to Dallas'
```

Now, let's take a look at how the Web code would handle the activation and navigation to the relevant page within the app (or

Figure 6 **Back-Button Code**

```
(function() {


  if (typeof Windows !== 'undefined' &&
    typeof Windows.UI !== 'undefined' &&
    typeof Windows.ApplicationModel !== 'undefined') {

   Windows.UI.Core.SystemNavigationManager.getForCurrentView().
     appViewBackButtonVisibility =
     Windows.UI.Core.AppViewBackButtonVisibility.visible;
        Windows.UI.Core.SystemNavigationManager.getForCurrentView().
        addEventListener("backrequested", onBackRequested);


function onBackRequested(eventArgs) {


      window.history.back();
      eventArgs.handled = true;
    }
  }

  })();
```

Web site) based on the input. You create a separate JavaScript file for coding this scenario.

The code in **Figure 5** should be referred in the <body> section of the caller page/referencing page just before the DOMContentLoaded event triggers. Hence, it's best to add <script src> just before the <body> element ends in the referencing page. (**Note:** Because the app is "activated" by Cortana using the "activationKind" as "voiceCommand," it's important to register the event handler for this activation. To register app lifecycle events where the app isn't a native WinJS or C# one, the namespace Windows.UI.WebUI.WebUIApplication is provided to help subscribe to and handle specific events.)

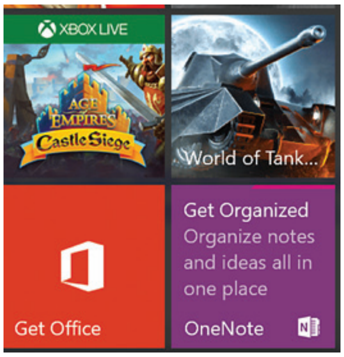

XBOX LIVE
AGE of EMPIRES Castle Siege
World of Tank...
Get Organized
Organize notes and ideas all in one place
Get Office
OneNote

Figure 7 **Live Tiles Examples**

In the code in **Figure 5**, Cortana APIs will receive the user voice input and populate the SpeechRecognition property of the activation classes. This should help retrieve the converted text in the code and help the app perform relevant actions. In this snippet, you use a switch-case statement to evaluate the textSpoken variable and route the user to the Cities.html page with the value of city appended as querystring.

Clearly, this is just one of the scenarios and, given the routing configuring of each Web site (MVC, REST and so on), the cases will change accordingly. The app is now ready to talk to Cortana.

The (Missing) Back Button

After discussing some of the most-advanced functionalities, let's look at one of the basic—but important—aspects of the app experience: navigation. Ease of browsing through the app and intuitive navigation hierarchy helps users to predictively experience an app.

This scenario is special because when you port the responsive Web site to a UWP app, the UI behaves as expected. However, you need to provide the app container with more pointers about handling the navigation within the app. The default UX goes like this:

1. User launches the app.
2. The user browses various sections of the app by clicking hyperlinks or menus on the pages.
3. At one point, the user wishes to go to the previous page and clicks the hardware or software back button provided by the Windows OS (there is no back button in the app yet).
4. The app exits.

Point No. 4 is an unexpected result and needs to be fixed. The solution is simple, which includes instructing the app container window to go back through regular JavaScript APIs. **Figure 6** shows the code for this scenario.

The initial few lines enable the framework-provided back button, which appears on the top of the app and then registers an event handler for the tap/click event. All that you do in the code is access the "window" DOM object and instruct it to go one page back. There's one thing to remember: When the app is at the bottom of the navigation stack, there's no further pages available in the history and the app will exit at this point. Additional code needs to be written if custom experience is required to be baked into the app.

Live Tiles

Live Tiles is a feature of the UWP apps that displays crisp information about updates in the app or anything that might be of interest to the user without having to launching the app. A quick example can be viewed by hitting the Start menu on a Windows device. A few Live Tiles would be evident for apps such as News, Money and Sports.

Here are just a few use-case examples:

- For an e-commerce app, a tile can display recommendations or status of your order.
- In a line-of-business app, a tile can display a mini image of your organization's reports.
- In a gaming app, Live Tiles can display offers, achievements, new challenges and so on.

Figure 7 shows two examples of tiles from some of the Microsoft first-party apps.

One of the easiest ways to integrate Live Tiles into your hosted Web app is to create a Web API and set up the app code (shown in **Figure 8**) to poll it every few minutes. The job of the Web API is to send back XML, which will be used to create the tile content for the Windows app. (**Note:** It's important that the end user pins the app to the Start menu to experience Live Tiles.)

Figure 8 **Simple Live Tiles Code**

```
function enableLiveTile()
  {
    if (typeof Windows !== 'undefined' &&
      typeof Windows.UI !== 'undefined' &&
      typeof Windows.ApplicationModel !== 'undefined') {

      {
        var notification = Windows.UI.Notifications;
        var tileUpdater =
          notification.TileUpdateManager.createTileUpdaterForApplication();
        var recurrence = notification.PeriodicUpdateRecurrence.halfHour;
        var url = new Windows.Foundation.Uri("<URL to receieve the XML for tile>");
        tileUpdater.startPeriodicUpdate(url, recurrence);

      }
      }
  }
```

Figure 9: **Another Option for Live Tiles Creations**

```
function createLiveTile() /* can contain parameters */
{

  var notifications = Windows.UI.Notifications,
  tile = notifications.TileTemplateType.tileSquare310x310ImageAndText01,
  tileContent = notifications.TileUpdateManager.getTemplateContent(tile),
  tileText = tileContent.getElementsByTagName('text'),
  tileImage = tileContent.getElementsByTagName('image');

  // Get the text for live tile here [possibly] from a remote service through xhr.

  tileText[0].appendChild(tileContent.createTextNode('Demo Message')); // Text here.
  tileImage[0].setAttribute('src','<URL of image>');

  var tileNotification = new notifications.TileNotification(tileContent);
  var currentTime = new Date();
  tileNotification.expirationTime = new Date(currentTime.getTime() + 600 * 1000);
    notifications.TileUpdateManager.createTileUpdaterForApplication().
    update(tileNotification);

}
```

The most important class here is the TileUpdateManager from the Windows.UI.Notifications namespace. This class not only creates a template internally to send to the tile, but also polls the specified URL for tile XML content through startPeriodicUpdate method. The duration of the poll can be set using the PeriodicUpdateRecurrence enumeration to period pull the XML content for Tiles. This approach is more server-driven where the Web API sends the XML code and tile template to the client. This is feasible when the developer has control over the app and the service layers.

Now consider a scenario in which the app receives information from third-party Web APIs such as weather or market research data. In such scenarios, mostly the Web APIs would send standard HTTP responses in terms of body, headers and so on. Here, you can parse the Web API response and then form an XML tile of content in the client-side UWP app code in JavaScript. This gives the app developer more control over the type of templates to display the data. You can also mention the expiration time for the tile through the TileNotification class. The code for this is shown in **Figure 9**.

Note the TileTemplateType class provides the functionality of creating a square tile template 310 x 310px in size with an image and text. Also, the expiration time for the tile is set to 10 minutes through code, which means that after this time, the Live Tile would revert to the default app tile provided in the app package, unless a new notification arrives in the app in form of push notifications. More information about available tile templates can be found at bit.ly/2k5PDJj.

Wrapping Up

There are a few things to consider while planning the migration from a Web app to a UWP app:

1. Test your app for layout and rendering with modern browsers (Microsoft Edge is an example).
2. If your Web app is dependent on an ActiveX control or plug-in, ensure an alternative way to make the functionality work when running on modern browsers or as a UWP app.
3. Use the SiteScan tool at bit.ly/1PJBcpi to surface recommendations related to libraries and functionality.
4. Identify URLs of external resources that your Web site references. These will be required to be added to the ApplicationContentUriRules section of the Appx.manifest file.

Additionally, there are many deeper integrations that can be achieved through the JavaScript Windows object and help light up the app functionality through richer experiences. Contacts, Camera, Microphone, Toast notifications and many features open a window of opportunity for blending your Web site with the app persona. The code in this article has been converted into a project template and made available for developers through GitHub at bit.ly/2k5FlJh. ■

Sagar Bhanudas Joshi *has worked with developers and ISVs on the Universal Windows Platform and Microsoft Azure for more than six years. His role includes working with startups to help them architect, design, and provide on-board solutions and applications to Azure, Windows, and the Office 365 platform. Joshi lives and works in Mumbai, India. Find him on Twitter: @sagarjms.*

Thanks to the following Microsoft technical experts who reviewed this article: *Sandeep Alur and Ashish Sahu*

dtSearch®

Instantly Search Terabytes of Data

across a desktop, network, Internet or Intranet site with dtSearch enterprise and developer products

Over 25 search features, with easy multicolor hit-highlighting options

dtSearch's document filters support popular file types, emails with multilevel attachments, databases, web data

Developers:

- APIs for .NET, Java and C++
- SDKs for Windows, UWP, Linux, Mac and Android
- See dtSearch.com for articles on faceted search, advanced data classification, working with SQL, NoSQL & other DBs, MS Azure, etc.

Visit dtSearch.com for

- hundreds of reviews and case studies
- fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

dtSearch.com 1-800-IT-FINDS

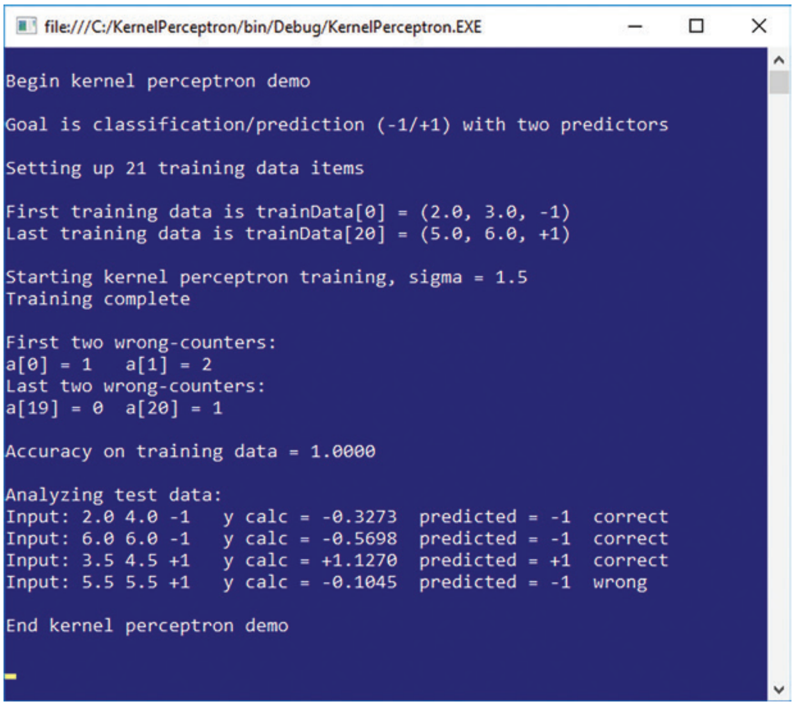
Kernel Perceptrons Using C#

A kernel perceptron is a machine learning (ML) classifier that can be used to make binary predictions. For example, a kernel perceptron could predict the sex of a person (male = -1, female = +1) based on age, income, height and weight. Kernel perceptrons are an advanced variation of ordinary perceptrons and can handle more complex data.

A good way to see where this article is headed is to take a look at the demo program in **Figure 1** and the associated data in **Figure 2**. The goal of the demo program is to predict the class, -1 or +1, of dummy input data that has just two predictor variables, x0 and x1. For example, the first training data item is (2.0, 3.0, -1), which means that if the input values are x0 = 2.0 and x1 = 3.0, the correct class is -1.

The 21 training data points have a circular geometry, which means that simple linear classification techniques, such as ordinary perceptrons or logistic regression, are ineffective. Such data is called non-linearly separable.

The demo program uses the training data to create a prediction model using a kernel perceptron. Unlike many prediction models that consist of a set of numeric values called weights, a kernel prediction model creates a set of counter values, one for each training data item. The demo program displays the counter values for the first two data items (1 and 2) and the last two data items (0 and 1).



file:///C:/KernelPerceptron/bin/Debug/KernelPerceptron.EXE

```
Begin kernel perceptron demo

Goal is classification/prediction (-1/+1) with two predictors

Setting up 21 training data items

First training data is trainData[0] = (2.0, 3.0, -1)
Last training data is trainData[20] = (5.0, 6.0, +1)

Starting kernel perceptron training, sigma = 1.5
Training complete

First two wrong-counters:
a[0] = 1   a[1] = 2
Last two wrong-counters:
a[19] = 0  a[20] = 1

Accuracy on training data = 1.0000

Analyzing test data:
Input: 2.0 4.0 -1    y calc = -0.3273  predicted = -1  correct
Input: 6.0 6.0 -1    y calc = -0.5698  predicted = -1  correct
Input: 3.5 4.5 +1    y calc = +1.1270  predicted = +1  correct
Input: 5.5 5.5 +1    y calc = -0.1045  predicted = -1  wrong

End kernel perceptron demo
```

Figure 1 **Kernel Perceptron Demo**

After training, the kernel perceptron model predicts all 21 data items correctly, which is to be expected. The trained model is applied to four new test data items that were not used during training. The first test item has inputs (2.0, 4.0) and a class of -1. The prediction model correctly predicts that item. Overall, the model correctly predicts three of the four test items for 0.75 accuracy.

Behind the scenes, the kernel perceptron uses a function called a radial basis function (RBF) kernel. The kernel function requires a parameter called sigma. The value of sigma must be determined by trial and error, and sigma = 1.5 in the demo. Note that by setting sigma to 0.5, the demo program would achieve 100 percent accuracy. I used sigma = 1.5 to point out that for most data sets you won't achieve 100 percent accuracy.

Code download available at msdn.com/magazine/0417magcode.

This article assumes you have intermediate or higher programming skills, but doesn't assume you know anything about kernel perceptrons. The demo program is coded using C#, but you should have no trouble refactoring the code to another language such as Python or JavaScript if you wish. All the key demo code, except for the hardcoded data and some display statements, is presented in this article. The complete demo source code, including data, is available in the accompanying download.

Understanding Kernel Functions

In order to understand kernel perceptrons you must understand kernel functions in general. There are many kernel functions, but the most common one, and the type used by the demo program, is called the RBF kernel. The RBF kernel function accepts two vectors (that is, arrays) and a parameter named sigma, and returns a single value between 0.0 and 1.0 that's a measure of similarity between the two arrays. A return value of exactly 1.0 means the two arrays are identical. The less similar the two arrays are, the smaller the value of RBF is, approaching but never quite reaching 0.0.

The equation for RBF is:

K(x1, x2) = exp(- || x1 - x2 ||^2 / (2 * sigma^2))

Here, K stands for kernel, x1 and x2 are two arrays that have the same length, sigma is a free parameter with a value like 1.0 or 1.5, the || indicates Euclidean distance, and exp means Euler's number (e) raised to a power.

The RBF is best explained by example. Suppose x1 = (2.0, 1.0, 3.0) and x2 = (0.0, 1.0, 5.0), and sigma is 1.0. First, you compute the squared Euclidean distance:

```
|| x1 - x2 ||^2 = (2.0 - 0.0)^2 + (1.0 - 1.0)^2  + (3.0 - 5.0)^2
                = 4.0 + 0.0 + 4.0
                = 8.0
```

Next, you divide the squared distance by 2 times sigma squared:

8.0 / (2 * (1.0)^2) = 8.0 / 2.0 = 4.0

Last, you take Euler's number (e = approximately 2.71828) and raise it to the negative of the previous result:

K = e^(-4.0) = 0.0183

A kernel perceptron is a machine learning (ML) classifier that can be used to make binary predictions.

Notice that if x1 equals x2, the squared Euclidean distance will be 0, and then dividing by 2 times sigma squared will be 0, and e raised to the 0th power is 1.0. The less similar x1 and x2 are, the larger the squared difference will be, and e raised to a negative large value gets very small, approaching 0.0. The larger the value of sigma is, the larger the value of K is, but K still varies from 1.0 (equal vectors) down to 0.0 exclusive (very dissimilar vectors). The order of array parameters doesn't matter, so K(x1, x2) = K(x2, x1).

The demo program defines the RBF kernel function as:

```
static double Kernel(double[] d1, double[] d2, double sigma) {
  double num = 0.0;
  for (int i = 0; i < d1.Length - 1; ++i)
    num += (d1[i] - d2[i]) * (d1[i] - d2[i]);
  double denom = 2.0 * sigma * sigma;
  double z = num / denom;
  double result = Math.Exp(-z);
  return result;
}
```

The function assumes that the last cell of each array holds the class label (+1 or -1), so the last cell isn't included in the calculation. There are several other kernel functions that can be used with a kernel perceptron, including the linear kernel, polynomial kernel, Fisher kernel, and sigmoid kernel. Calculating kernel functions is relatively easy. What's not obvious, however, is that kernel functions have some remarkable mathematical properties that allow simple classifiers, like an ordinary perceptron, to be transformed into classifiers that can handle non-linearly separable data.

Understanding Ordinary Perceptrons

An ordinary perceptron can perform binary classification for simple, linearly separable data. An ordinary perceptron consists of a set of weights, where the number of weights equals the number of predictor values, and an additional special weight called the bias. For example, suppose you have a binary classification problem where there are three predictor variables. And suppose the perceptron weights are w0 = 1.5, w1 = 1.1 and w2 = -3.0, and three predictor variables have values x0 = 4.0, x1 = 3.0, x2 = 5.0 and the bias is b = 2.5. The prediction is calculated as the sign (positive or negative) of the sum of the products of weight and input values, plus the bias:

```
sum = (w0)(x0) + (w1)(x1) +(w2)(x2) + b
    = (1.5)(4.0) + (1.1)(3.0) + (-3.0)(5.0) + 2.5
    = 6.0 + 3.3 + (-15.0) + 2.5
    = -3.2
prediction = sign(sum)
           = -1
```

It's that simple. But where do the weights and bias values come from? You take training data that has known input and correct class values and then use a mathematical optimization algorithm to find values for the weights and bias so that the calculated class values closely match the known correct class values.

Note that in perceptron literature, the bias value is often considered to be a regular weight associated with a dummy input value of 1.0. Using that scheme, the preceding example would have inputs x = (1.0, 4.0, 3.0, 5.0) and weights w = (2.5, 1.5, 1.1, -3.0), yielding the following (which is the same result as before):

```
sum = (2.5)(1.0) + (1.5)(4.0) + (1.1)(3.0) + (-3.0)(5.0)
    = -3.2
```

In other words, a perceptron bias value can be explicit or implicit. You should not underestimate the confusion this can cause.

Now, unfortunately, ordinary perceptrons can classify only linearly separable data, which makes them very limited in practice. However, by using a kernel function in conjunction with a perceptron, it's possible to create a classifier that can work with non-linearly separable data, such as the demo data shown in **Figure 2**.

The Kernel Perceptron Algorithm

At first glance, the kernel perceptron algorithm appears unrelated to the ordinary perceptron algorithm, but in fact, the two are deeply

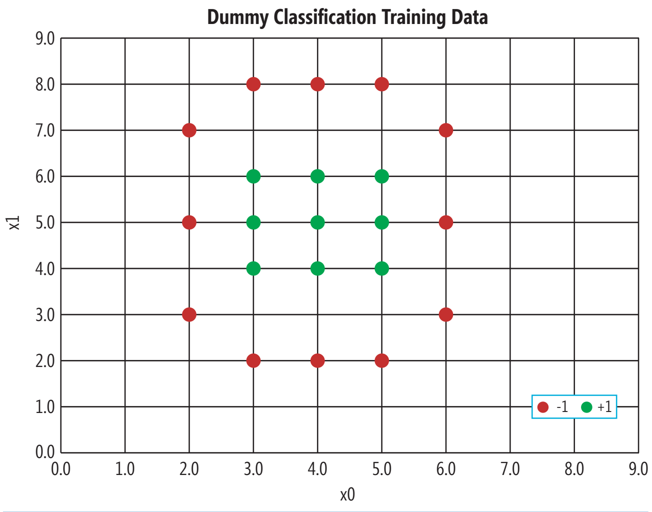


Figure 2 **Kernel Perceptron Training Data**

related. Suppose there are just four training data items: td[0] = (2.0, 3.0, -1), td[1] = (3.0, 2.0, +1), td[2] = (3.0, 5.0, +1), td[3] = (4.0, 3.0, -1). And suppose the trained kernel perceptron model gave you wrong counter values of a = (1, 2, 3, 4). (I use "a" for the wrong counters because in research literature they're usually given the symbol Greek sigma, which resembles lowercase English "a.")

To predict the class of new data item x = (3.0, 6.0), you compute the weighted sum (by a-value and training item correct class) of the kernel function applied to the new data item and each of the four training items:

```
K(td[0], x) = 0.1083
K(td[1], x) = 0.0285
K(td[2], x) = 0.8007
K(td[3], x) = 0.1083

sum = (1)(0.1083)(-1) + (2)(0.0285)(+1) + (3)(0.8007)(+1) +
(4)(0.1083)(-1)
      = +1.9175
prediction = sign(sum)
             = +1
```

Stated somewhat loosely, the kernel perceptron looks at the similarity of the data item to be classified and all training data items, and aggregates the similarity values—using the wrong counters as weights—into a single value that indicates predicted class.

Behind the scenes, the kernel perceptron uses a function called a radial basis function (RBF) kernel.

So, to make a kernel perceptron prediction you need the training data and the associated wrong counter values. When you train an ordinary perceptron, you use an iterative process. In each iteration, you use the current values of the weights and bias to calculate a predicted class. If the predicted class is incorrect (doesn't match the class in the training data) you adjust the weights a bit so that the predicted class is closer to the known correct class value.

In a kernel perceptron, you use a similar iterative training process, but instead of adjusting weight values when a calculated class is wrong, you increment the wrong counter for the current training item. Quite remarkable! The math proof of why this works is stunningly beautiful and can be found in the Wikipedia entry for kernel perceptrons.

Expressed in high-level pseudo-code, the kernel perceptron training algorithm is:

```
loop a few times
  for each curr training item, i
    for each other training item, j
      sum += a[j] * K(td[i], td[j], sigma) * y[j];
    end-for
    pred = sign(sum)
    if (pred is wrong) increment a[i]
  end-for
end-loop
```

The two parameters needed in the training algorithm are the number of times to iterate and the value of sigma for the kernel function. Both of these values must be determined by a bit of trial and error. The demo program iterates 10 times and uses 1.5 for sigma.

Note that the ordinary perceptron training algorithm uses training data to generate the weights and bias values, and then conceptually discards the training data. The kernel perceptron training algorithm generates wrong counter values that are mathematically related to weights, but the algorithm must keep the training data in order to make predictions.

The Demo Program Structure

The overall structure of the demo program, with a few minor edits to save space, is presented in **Figure 3**. I used a static method style rather than an object-oriented programming style for

Figure 3 **Kernel Perceptron Demo Program Structure**

```
using System;
namespace KernelPerceptron
{
  class KernelPerceptronProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Begin demo ");
      int numFeatures = 2;

      Console.WriteLine("Goal is classification(-1/+1) ");
      Console.WriteLine("Setting up 21 training items ");

      double[][] trainData = new double[21][];
      trainData[0] = new double[] { 2.0, 3.0, -1 };
      trainData[1] = new double[] { 2.0, 5.0, -1 };
      . . .
      trainData[20] = new double[] { 5.0, 6.0, +1 };
      int numTrain = trainData.Length;

      double[][] testData = new double[4][];
      testData[0] = new double[] { 2.0, 4.0, -1 };
      . .
      testData[3] = new double[] { 5.5, 5.5, +1 };

      int[] a = new int[trainData.Length];
      int maxEpoch = 10;
      int epoch = 0;
      double sigma = 1.5;  // for the kernel function

      Console.WriteLine("Starting train, sigma = 1.5 ");
      ...
      Console.WriteLine("Training complete ");

      double trainAcc = Accuracy(trainData, a,
        trainData, sigma, false);  // silent
      Console.WriteLine("Accuracy = " +
        trainAcc.ToString("F4"));

      Console.WriteLine("Analyzing test data: ");
      double testAcc = Accuracy(testData, a,
        trainData, sigma, true);  // verbose

      Console.WriteLine("End kernel perceptron demo ");
      Console.ReadLine();

    } // Main

    static double Kernel(double[] d1, double[] d2,
      double sigma) { . . }

    static double Accuracy(double[][] data, int[] a,
      double[][] trainData, double sigma, bool verbose)
    { . . }
  } // Program
} // ns
```

GROUPDOCS
Document Manipulation APIs

Manipulating Files?

APIs to view, export, annotate, compare, sign, automate and search documents in your applications.

GroupDocs.Total

GroupDocs.**Viewer**
GroupDocs.**Annotation**
GroupDocs.**Conversion**
GroupDocs.**Comparison**
GroupDocs.**Signature**
GroupDocs.**Assembly**
GroupDocs.**Metadata**
GroupDocs.**Search**
GroupDocs.**Text**

Try for Free

.NET Libraries   Java Libraries   Cloud APIs

Contact Us:
US: +1 903 306 1676
EU: +44 141 628 8900
AU: +61 2 8006 6987
sales@asposeptyltd.com

Visit us at www.groupdocs.com

GROUPDOCS
Document Manipulation APIs

simplicity. The Main method has all the control logic. There are two helper methods, Kernel and Accuracy.

To code the demo program, I launched Visual Studio and created a new C# console application program and named it KernelPerceptron. I used Visual Studio 2015, but the demo program has no significant .NET Framework dependencies so any recent version will work.

In order to understand kernel perceptrons you must understand kernel functions in general.

After the template code loaded into the editor window, I right-clicked on file Program.cs in the Solution Explorer window and renamed the file to KernelPerceptronProgram.cs, then allowed Visual Studio to automatically rename class Program for me. At the top of the template-generated code, I deleted all unnecessary using statements, leaving just the one that references the top-level System namespace.

The Main method sets up the training and test data like so:

```
int numFeatures = 2;
double[][] trainData = new double[21][];
trainData[0] = new double[] { 2.0, 3.0, -1 };
. .
trainData[20] = new double[] { 5.0, 6.0, +1 };
int numTrain = trainData.Length;
double[][] testData = new double[4][];
testData[0] = new double[] { 2.0, 4.0, -1 };
. .
testData[3] = new double[] { 5.5, 5.5, +1 };
```

The demo uses two predictor variables (also called features in ML terminology) for simplicity, but kernel perceptrons can handle any number of predictor variables. The data is hardcoded but in a non-demo scenario you'd likely load the data from a text file. The demo uses -1 and +1 to represent the two possible classes. This encoding is typical for perceptrons, but classes can be encoded as 0 and 1 instead (though this encoding would require some changes to the code logic).

Training is prepared with these statements:

```
int[] a = new int[numTrain];  // "Wrong counters"
double sigma = 1.5;  // For the kernel function
int maxEpoch = 10;
int epoch = 0;
```

Array a holds the wrong counters for each training item, and sigma is the free parameter for the RBF kernel function. The value of sigma and the maxEpoch loop control were determined by trial and error. Next, all possible kernel function values are pre-calculated and stored into a matrix:

```
double[][] kernelMatrix = new double[numTrain][];
for (int i = 0; i < kernelMatrix.Length; ++i)
  kernelMatrix[i] = new double[numTrain];
for (int i = 0; i < numTrain; ++i) {
  for (int j = 0; j < numTrain; ++j) {
    double k = Kernel(trainData[i], trainData[j], sigma);
    kernelMatrix[i][j] = kernelMatrix[j][i] = k;
  }
}
```

The idea is that during training, the kernel similarity between all pairs of training items will have to be used several times so it makes sense to pre-calculate these values.

The training loop is:

```
while (epoch < maxEpoch) {
  for (int i = 0; i < numTrain; ++i) {
    // Get "desired" correct class into di
    for (int j = 0; j < numTrain; ++j) {
      // Get "other" desired class into dj
      // Compute y = weighted sum of products
    }

    if ((di == -1 && y >= 0.0) || (di == 1 && y <= 0.0))
      ++a[i]; // increment wrong counter
  }
  ++epoch;
}
```

The desired, correct class (-1 or +1) is pulled from the current training item with this code:

```
int di = (int)trainData[i][numFeatures];
```

I use di here to stand for desired value. Two other common variable names are t (for target value) and y (which just stands for general output).

The inner nested for loop that calculates the weighted sum of kernel values is the heart of the kernel perceptron-learning algorithm:

```
double y = 0.0; // Weighted sum of kernel results
for (int j = 0; j < numTrain;; ++j) {
  int dj = (int)trainData[j][numFeatures];
  double kern = kernelMatrix[i][j];
  y += a[j] * dj * kern;
}
```

The demo code calculates the kernel for all pairs of training items. But when an associated wrong counter has value 0, the product term will be 0. Therefore, an important optional optimization when the number of training items is large is to skip the calculation of the kernel when the value of a[i] or a[j] is 0. Equivalently, training items that have a wrong counter value of 0 can be removed entirely.

I don't compute an explicit predicted class because it's easier to check if the predicted class is wrong directly:

```
if ((di == -1 && y >= 0.0) || (di == 1 && y <= 0.0))
  ++a[i]; // wrong counter for curr data
```

You have to be careful to check y >= 0.0 or y <= 0.0 rather than y > 0.0 or y < 0.0 because the first time through the training loop, all wrong counter values in the a array are zero and so the weighed sum of products will be 0.0.

After the training loop terminates, the kernel perceptron is effectively defined by the training data, the wrong-counter array and the RBF kernel parameter sigma.

Making Predictions

Helper function Accuracy makes predictions. The method's definition starts with:

```
static double Accuracy(double[][] data, int[] a,
  double[][] trainData, double sigma, bool verbose)
{
  int numFeatures = data[0].Length - 1;
  double[] x = new double[numFeatures];
  int numCorrect = 0;
  int numWrong = 0;
...
```

The parameter named data holds an array-of-arrays style matrix of data to evaluate. Parameter array a holds the wrong counter values generated by training.

Inside the body of the function, the key code is exactly like the training code, except instead of incrementing a wrong counter for

Extreme Performance Linear Scalability

NCache™

Distributed Cache

- In-Memory App Data Caching
- ASP.NET Sessions & View State
- Runtime Data Sharing

NosDB

NoSQL Database

- Schema-Free JSON Documents
- Multiple Shards & Data Replication
- Powerful SQL, LINQ, ADO.NET

100% Native .NET

Open Source

sales@alachisoft.com **alachisoft.com** US: +1 (925) 236 3830 the current training data item on an incorrect prediction, a single numWrong variable or numCorrect variable is incremented:

```
if ((di == -1 && y >= 0.0) || (di == 1 && y <= 0.0))
  ++numWrong;
else
  ++numCorrect;
```

After all data items under investigation have been examined, the percentage of correct predictions is returned:

```
return (1.0 * numCorrect) / (numCorrect + numWrong);
```

The Accuracy method has a verbose parameter, which if true, causes diagnostic information to be displayed:

```
if (verbose == true)
{
  Console.Write("Input: ");
  for (int j = 0; j < data[i].Length - 1; ++j)
    Console.Write(data[i][j].ToString("F1") + " ");
  // Etc.
}
```

Using the code from the Accuracy method, you could write a dedicated Predict method that accepts an array of x values (without a known correct class), the training data, the wrong-counter array, and the RBF kernel sigma value, which returns a +1 or -1 prediction value.

Wrapping Up

Kernel perceptrons aren't used very often. This is due in large part to the fact that there are more powerful binary classification techniques available, and that there is a lot of mystery surrounding ML kernel methods in general. If you do an Internet search for kernel perceptrons, you'll find many references that show the beautiful mathematical relationships between ordinary perceptrons and kernel perceptrons, but very little practical implementation information. In my opinion, the primary value of understanding kernel perceptrons is that the knowledge makes it easier to understand more sophisticated ML kernel methods (which I'll present in a future column).

An ordinary perceptron can perform binary classification for simple, linearly separable data.

One of the weaknesses of kernel perceptrons is that to make a prediction, the entire training data set (except for items that have a wrong counter value of 0) must be examined. If the training set is huge, this could make real-time predictions unfeasible in some scenarios.

Kernel perceptrons are arguably the simplest type of kernel methods. The kernel trick can be applied to other linear classifiers. In fact, applying the kernel trick to a maximum margin linear classifier is the basis for support vector machine (SVM) classifiers, which were popular in the late 1990s. ■

Dr. James McCaffrey *works for Microsoft Research in Redmond, Wash. He has worked on several Microsoft products including Internet Explorer and Bing. Dr. McCaffrey can be reached at jammc@microsoft.com.*

Thanks to the following Microsoft technical experts who reviewed this article: *Ani Anirudh and Chris Lee*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

6 LOCATIONS TO CHOOSE FROM
JOIN US

ROCK YOUR CODE TOUR 2017

AUSTIN MAY 15-18
Austin
May 15-18
See pages 68-71

WASH. DC JUNE 12-15
Washington, DC
June 12-15
See pages 50-51

REDMOND AUG 14-18
Redmond
August 14-18
See pages 52-53

CHICAGO SEPT 18-21
Chicago
September 18-21
See pages 60-61

ANAHEIM OCT 16-19
Anaheim
October 16-19

ORLANDO NOV 13-17
Orlando
November 12-17

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

vslive.com

TURN THE PAGE FOR MORE EVENT DETAILS

Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

WASH, DC
JUNE 12-15, 2017
MARRIOTT MARQUIS

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

Development Topics include:

- Visual Studio / .NET
- JavaScript / HTML5
- Angular
- Native Mobile & Xamarin
- Software Practices
- Database and Analytics
- ASP.NET Core
- Web API
- ALM / DevOps
- Cloud Computing
- UWP
- Unit Testing

Register by April 21 for Best Savings

Save $300!*

Use Code VSLDC5

*Some restrictions apply. Discount off of 4 day packages only.

REGISTER NOW

EVENT PARTNER Microsoft

SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA INC. YOUR GROWTH. OUR BUSINESS.

DC AGENDA AT-A-GLANCE

ALM / DevOps	Cloud Computing	Database and Analytics	Native Client	Software Practices	Visual Studio / .NET Framework	Web Client	Web Server

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, June 12, 2017 *(Separate entry fee required)*		
7:30 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries		
9:00 AM	6:00 PM	**M01** Workshop: Distributed Cross-Platform Application Architecture *- Jason Bock & Rockford Lhotka*	**M02** Workshop: Practical ASP.NET DevOps with VSTS or TFS *- Brian Randell*	**M03** Workshop: SQL Server 2016 for Developers *- Andrew Brust & Leonard Lobel*
6:45 PM	9:00 PM	Dine-A-Round		

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, June 13, 2017			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:00 AM	**KEYNOTE:** To Be Announced - **Saurabh Pant,** *Program Manager, Microsoft*			
9:15 AM	10:30 AM	**T01** Go Mobile With C#, Visual Studio, and Xamarin *- Kevin Ford*	**T02** Build Better JavaScript Apps with TypeScript *- Brian Noyes*	**T03** What's New in Azure IaaS v2 *- Eric D. Boyd*	**T04** Tactical DevOps with VSTS *- Brian Randell*
10:45 AM	12:00 PM	**T05** Conquer the Network - Making Resilient and Responsive Connected Mobile C# Apps *- Roy Cornelissen*	**T06** Getting Started with Aurelia *- Brian Noyes*	**T07** Cloud Oriented Programming *- Vishwas Lele*	**T08** PowerShell for Developers *- Brian Randell*
12:00 PM	1:30 PM	Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** Lessons Learned from Real World Xamarin.Forms Projects *- Nick Landry*	**T10** Getting to the Core of .NET Core *- Adam Tuliper*	**T11** To Be Announced	**T12** New SQL Server 2016 Security Features for Developers *- Leonard Lobel*
3:00 PM	4:15 PM	**T13** Xamarin vs. Cordova *- Sahil Malik*	**T14** Assembling the Web - A Tour of WebAssembly *- Jason Bock*	**T15** Go Serverless with Azure Functions *- Eric D. Boyd*	**T16** No Schema, No Problem! Introduction to Azure DocumentDB *- Leonard Lobel*
4:15 PM	5:30 PM	Welcome Reception			

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, June 14, 2017			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**W01** To Be Announced	**W02** Creating Reactive Applications With SignalR *- Jason Bock*	**W03** Bots are the New Apps: Building Bots with ASP.NET WebAPI & Language Understanding *- Nick Landry*	**W04** What's New in Visual Studio 2017 *- Robert Green*
9:30 AM	10:45 AM	**W05** Write Once Run Everywhere, Cordova, Electron, and Angular2 *- Sahil Malik*	**W06** Explore Web Development with Microsoft ASP.NET Core 1.0 *- Mark Rosenberg*	**W07** Exploring C# 7 New Features *- Adam Tuliper*	**W08** Get Started with Git and GitHub *- Robert Green*
11:00 AM	12:00 PM	General Session: To Be Announced			
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**W09** Distributed Architecture: Microservices and Messaging *- Rockford Lhotka*	**W10** Angular 101: Part 1 *- Deborah Kurata*	**W11** A Busy Developer's Intro to Windows Containers *- Vishwas Lele*	**W12** Continuous Delivery 3.0, The Next Step *- Marcel de Vries*
3:00 PM	4:15 PM	**W13** Agile Failures: Stories From The Trenches *- Philip Japikse*	**W14** Angular 101: Part 2 *- Deborah Kurata*	**W15** Use Docker to Develop, Build and Deploy Applications, a Primer *- Mark Rosenberg*	**W16** Mobile DevOps Demystified with Xamarin, VSTS and HockeyApp *- Roy Cornelissen*
4:30 PM	5:45 PM	**W17** Top 10 Ways to Go from Good to Great Scrum Master *- Benjamin Day*	**W18** SASS and CSS for Developers *- Robert Boedigheimer*	**W19** Microservices with Azure Container Service & Service Fabric *- Vishwas Lele*	**W20** A/B Testing, Canary Releases and Dark Launching, Implementing Continuous Delivery on Azure *- Marcel de Vries*
6:45 PM	10:30 PM	Visual Studio Live! Monuments by Moonlight Tour			

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, June 15, 2017			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**TH01** Let's Write a Windows 10 App: A Basic Introduction to Universal Apps *- Billy Hollis*	**TH02** JavaScript for the C# (and Java) Developer *- Philip Japikse*	**TH03** Entity Framework Core for Enterprise Applications *- Benjamin Day*	**TH04** Overcoming the Challenges of Mobile Development in the Enterprise *- Roy Cornelissen*
9:30 AM	10:45 AM	**TH05** Implementing the MvvM Pattern in Your Xamarin Apps *- Kevin Ford*	**TH06** Integrating AngularJS & ASP. NET MVC *- Miguel Castro*	**TH07** Power BI: Analytics for Desktop, Mobile and Cloud *- Andrew Brust*	**TH08** Exploring Microservices in a Microsoft Landscape *- Marcel de Vries*
11:00 AM	12:15 PM	**TH09** Building Cross-Platform Business Apps with CSLA. NET *- Rockford Lhotka*	**TH10** Debugging Your Website with Fiddler and Chrome Developer Tools *- Robert Boedigheimer*	**TH11** Big Data with Hadoop, Spark and Azure HDInsight *- Andrew Brust*	**TH12** End-to-End Dependency Injection & Testable Code *- Miguel Castro*
12:15 PM	1:15 PM	Lunch			
1:15 PM	2:30 PM	**TH13** Creating Great Looking Android Applications Using Material Design *- Kevin Ford*	**TH14** Building Single Page Web Applications Using Aurelia.js and the MVVM Pattern *- Ben Hoelting*	**TH15** Using Cognitive Services in Business Applications *- Michael Washington*	**TH16** Extend and Customize the Visual Studio Environment *- Walt Ritscher*
2:45 PM	4:00 PM	**TH17** Take the Tests: Can You Evaluate Good and Bad Designs? *- Billy Hollis*	**TH18** Tools for Modern Web Development *- Ben Hoelting*	**TH19** Introduction to Azure Machine Learning Studio (for the Non-Data Scientist) *- Michael Washington*	**TH20** Windows Package Management with NuGet and Chocolatey *- Walt Ritscher*

Speakers and sessions subject to change

vslive.com/dcmsdn

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

REDMOND
AUGUST 14-18, 2017
MICROSOFT HEADQUARTERS

JOIN US AT MICROSOFT HEADQUARTERS THIS SUMMER

- ➤ Rub elbows with blue badges
- ➤ Experience life on campus
- ➤ Enjoy lunch in the Commons and visit the Company Store
- ➤ Networking event on Lake Washington, Wednesday, August 16
- ➤ And so much more!

EVENT PARTNER
Microsoft

SUPPORTED BY
Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY
1105 MEDIA INC
YOUR GROWTH. OUR BUSINESS.

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

Development Topics include:

- AngularJS
- ASP.NET Core
- Azure
- Analytics
- DevOps
- .NET Framework
- Software Practices
- SQL Server
- Visual Studio 2017
- Web API
- UWP
- Xamarin

Register NOW and Save $400!

Use promo code VSLRED2

REGISTER NOW

ROCK YOUR CODE TOUR • 2017

REDMOND
AUG 14-18

"I liked that there was representation of Android and iOS, as well as, Microsoft. I prefer working with Microsoft tech/code but can't ignore Android and iOS." – Chris Nacey, Site Crafting

"This is the first conference that I have attended @Microsoft Headquarters and it truly has been a fantastic experience. Definitely exceeded my expectations and I look forward to coming back." – David Campbell, G3 Software

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

vslive.com/redmondmsdn

Understanding C# foreach Internals and Custom Iterators with yield

This month I'm going to explore the internals of a core construct of C# that we all program with frequently—the foreach statement. Given an understanding of the foreach internal behavior, you can then explore implementing the foreach collection interfaces using the yield statement, as I'll explain.

Although the foreach statement is easy to code, I'm surprised at how few developers understand how it works internally. For example, are you aware that foreach works differently for arrays than on IEnumberable<T> collections? How familiar are you with the relationship between IEnumerable<T> and IEnumerator<T>? And, if you do understand the enumerable interfaces, are you comfortable implementing them using yield?

What Makes a Class a Collection

By definition, a collection within the Microsoft .NET Framework is a class that, at a minimum, implements IEnumerable<T> (or the nongeneric type IEnumerable). This interface is critical because implementing the methods of IEnumerable<T> is the minimum needed to support iterating over a collection.

The foreach statement syntax is simple and avoids the complication of having to know how many elements there are.

The foreach statement syntax is simple and avoids the complication of having to know how many elements there are. The runtime doesn't directly support the foreach statement, however. Instead, the C# compiler transforms the code as described in the next sections.

foreach with Arrays: The following demonstrates a simple foreach loop iterating over an array of integers and then printing out each integer to the console:

```
int[] array = new int[]{1, 2, 3, 4, 5, 6};

foreach (int item in array)
{
  Console.WriteLine(item);
}
```

Code download available at itl.tc/MSDN.2017.04.

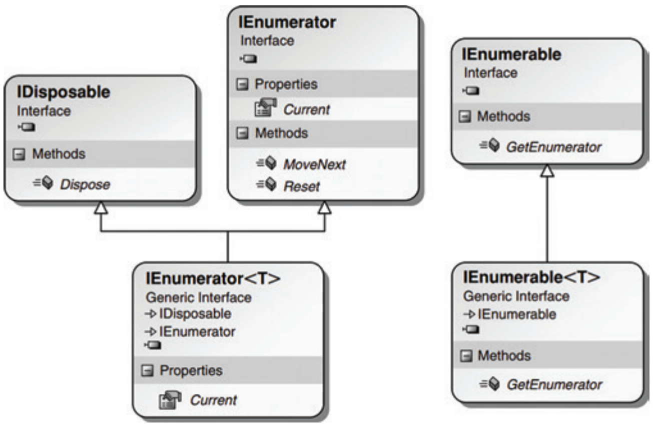


Figure 1 **A Class Diagram of the IEnumerator<T> and IEnumerator Interfaces**

From this code, the C# compiler creates a CIL equivalent of the for loop:

```
int[] tempArray;
int[] array = new int[]{1, 2, 3, 4, 5, 6};

tempArray = array;
for (int counter = 0; (counter < tempArray.Length); counter++)
{
  int item = tempArray[counter];

  Console.WriteLine(item);
}
```

In this example, note that foreach relies on the support for the Length property and the index operator ([]). With the Length property, the C# compiler can use the for statement to iterate through each element in the array.

Figure 2 **A Separate Enumerator Maintaining State During an Iteration**

```
System.Collections.Generic.Stack<int> stack =
  new System.Collections.Generic.Stack<int>();
int number;
System.Collections.Generic.Stack<int>.Enumerator
  enumerator;

// ...

// If IEnumerable<T> is implemented explicitly,
// then a cast is required.
// ((IEnumerable<int>)stack).GetEnumerator();
enumerator = stack.GetEnumerator();
while (enumerator.MoveNext())
{
  number = enumerator.Current;
  Console.WriteLine(number);
}
```

foreach with IEnumerable<T>: Although the preceding code works well on arrays where the length is fixed and the index operator is always supported, not all types of collections have a known number of elements. Furthermore, many of the collection classes, including Stack<T>, Queue<T> and Dictionary<TKey and TValue>, don't support retrieving elements by index. Therefore, a more general approach of iterating over collections of elements is needed. The iterator pattern provides this capability. Assuming you can determine the first, next, and last elements, knowing the count and supporting retrieval of elements by index is unnecessary.

The System.Collections.Generic.IEnumerator<T> and nongeneric System.Collections.IEnumerator interfaces are designed to enable the iterator pattern for iterating over collections of elements, rather than the length-index pattern shown previously. A class diagram of their relationships appears in **Figure 1**.

IEnumerator, which IEnumerator<T> derives from, includes three members. The first is bool MoveNext. Using this method, you can move from one element within the collection to the next, while at the same time detecting when you've enumerated through every item. The second member, a read-only property called Current, returns the element currently in process. Current is overloaded in IEnumerator<T>, providing a type-specific implementation of it. With these two members on the collection class, it's possible to iterate over the collection simply using a while loop:

```
System.Collections.Generic.Stack<int> stack =
  new System.Collections.Generic.Stack<int>();
int number;
// ...

// This code is conceptual, not the actual code.
while (stack.MoveNext())
{
  number = stack.Current;
  Console.WriteLine(number);
}
```

In this code, the MoveNext method returns false when it moves past the end of the collection. This replaces the need to count elements while looping.

You can have multiple bookmarks, and moving any one of them enumerates over the collection independently of the others.

(The Reset method usually throws a NotImplementedException, so it should never be called. If you need to restart an enumeration, just create a fresh enumerator.)

The preceding example showed the gist of the C# compiler output, but it doesn't actually compile that way because it omits two important details concerning the implementation: interleaving and error handling.

State Is Shared: The problem with an implementation such as the one in the previous example is that if two such loops interleave each other—one foreach inside another, both using the same collection—the collection must maintain a state indicator of the current element so that when MoveNext is called, the next element can be determined. In such a case, one interleaving loop can affect the other. (The same is true of loops executed by multiple threads.)

To overcome this problem, the collection classes don't support IEnumerator<T> and IEnumerator interfaces directly. Rather, there's a second interface, called IEnumerable<T>, whose only method is GetEnumerator. The purpose of this method is to return an object that supports IEnumerator<T>. Instead of the collection class maintaining the state, a different class—usually a nested class so that it has access to the internals of the collection—will support the IEnumerator<T> interface and will keep the state of the iteration loop. The enumerator is like a "cursor" or a "bookmark" in the sequence. You can have multiple bookmarks, and moving any one of them enumerates over the collection independently of the others. Using this pattern, the C# equivalent of a foreach loop will look like the code shown in **Figure 2**.

Cleaning up Following Iteration: Given that the classes that implement the IEnumerator<T> interface maintain the state, sometimes

Figure 3 **Compiled Result of foreach on Collections**

```
System.Collections.Generic.Stack<int> stack =
  new System.Collections.Generic.Stack<int>();
System.Collections.Generic.Stack<int>.Enumerator
  enumerator;
IDisposable disposable;

enumerator = stack.GetEnumerator();
try
{
  int number;
  while (enumerator.MoveNext())
  {
    number = enumerator.Current;
    Console.WriteLine(number);
  }
}
finally
{
  // Explicit cast used for IEnumerator<T>.
  disposable = (IDisposable) enumerator;
  disposable.Dispose();

  // IEnumerator will use the as operator unless IDisposable
  // support is known at compile time.
  // disposable = (enumerator as IDisposable);
  // if (disposable != null)
  // {
  //   disposable.Dispose();
  // }
}
```

Figure 4 **Error Handling and Resource Cleanup with using**

```
System.Collections.Generic.Stack<int> stack =
  new System.Collections.Generic.Stack<int>();
int number;

using(
  System.Collections.Generic.Stack<int>.Enumerator
    enumerator = stack.GetEnumerator())
{
  while (enumerator.MoveNext())
  {
    number = enumerator.Current;
    Console.WriteLine(number);
  }
}
```

you need to clean up the state after it exits the loop (because either all iterations have completed or an exception is thrown). To achieve this, the IEnumerator<T> interface derives from IDisposable. Enumerators that implement IEnumerator don't necessarily implement IDisposable, but if they do, Dispose will be called, as well. This enables the calling of Dispose after the foreach loop exits. The C# equivalent of the final CIL code, therefore, looks like **Figure 3**.

Notice that because the IDisposable interface is supported by IEnumerator<T>, the using statement can simplify the code in **Figure 3** to what is shown in **Figure 4**.

However, recall that the CIL doesn't directly support the using keyword. Thus, the code in Figure 3 is actually a more accurate C# representation of the foreach CIL code.

foreach *without* IEnumerable: C# doesn't require that IEnumerable/IEnumerable<T> be implemented to iterate over a data type using foreach. Rather, the compiler uses a concept known as duck typing; it looks for a GetEnumerator method that returns a type with a Current property and a MoveNext method. Duck typing involves searching by name rather than relying on an interface or explicit method call to the method. (The name "duck typing" comes from the whimsical idea that to be treated as a duck, the object must merely implement a Quack method; it need not implement an IDuck interface.) If duck typing fails to find a suitable implementation of the enumerable pattern, the compiler checks whether the collection implements the interfaces.

Figure 5 **Iterator Interfaces Pattern**

```
using System;
using System.Collections.Generic;

public class BinaryTree<T>:
  IEnumerable<T>
{
  public BinaryTree ( T value)
  {
    Value = value;
  }

  #region IEnumerable<T>
  public IEnumerator<T> GetEnumerator()
  {
    // ...
  }
  #endregion IEnumerable<T>

  public T Value { get; }  // C# 6.0 Getter-only Autoproperty
  public Pair<BinaryTree<T>> SubItems { get; set; }
}

public struct Pair<T>: IEnumerable<T>
{
  public Pair(T first, T second) : this()
  {
    First = first;
    Second = second;
  }
  public T First { get; }
  public T Second { get; }

  #region IEnumerable<T>
  public IEnumerator<T> GetEnumerator()
  {
    yield return First;
    yield return Second;
  }
  #endregion IEnumerable<T>

  #region IEnumerable Members
  System.Collections.IEnumerator
    System.Collections.IEnumerable.GetEnumerator()
  {
    return GetEnumerator();
  }
  #endregion

  // ...

}
```

Introducing Iterators

Now that you understand the internals of the foreach implementation, it's time to discuss how iterators are used to create custom implementations of the IEnumerator<T>, IEnumerable<T> and corresponding nongeneric interfaces for custom collections. Iterators provide clean syntax for specifying how to iterate over data in collection classes, especially using the foreach loop, allowing the end users of a collection to navigate its internal structure without knowledge of that structure.

The problem with the enumeration pattern is that it can be cumbersome to implement manually because it must maintain all the state necessary to describe the current position in the collection. This internal state might be simple for a list collection type class; the index of the current position suffices. In contrast, for data structures that require recursive traversal, such as binary trees, the

Figure 6 **Yielding Some C# Keywords Sequentially**

```
using System;
using System.Collections.Generic;

public class CSharpBuiltInTypes: IEnumerable<string>
{
  public IEnumerator<string> GetEnumerator()
  {
    yield return "object";
    yield return "byte";
    yield return "uint";
    yield return "ulong";
    yield return "float";
    yield return "char";
    yield return "bool";
    yield return "ushort";
    yield return "decimal";
    yield return "int";
    yield return "sbyte";
    yield return "short";
    yield return "long";
    yield return "void";
    yield return "double";
    yield return "string";
  }

    // The IEnumerable.GetEnumerator method is also required
    // because IEnumerable<T> derives from IEnumerable.
  System.Collections.IEnumerator
    System.Collections.IEnumerable.GetEnumerator()
  {
    // Invoke IEnumerator<string> GetEnumerator() above.
    return GetEnumerator();
  }
}

public class Program
{
  static void Main()
  {
    var keywords = new CSharpBuiltInTypes();
    foreach (string keyword in keywords)
    {
      Console.WriteLine(keyword);
    }
  }
}
```

Spreadsheets Made Easy.

WIN Windows Forms | Silverlight | WPF WPF

Fastest Calculations

Evaluate complex Excel-based models and business rules with the fastest and most complete Excel-compatible calculation engine available.

Powerful Controls

Add powerful Excel-compatible viewing, editing, formatting, calculating, filtering, sorting, charting, printing and more to your WinForms, WPF and Silverlight applications.

Comprehensive Charting

Enable users to visualize data with comprehensive Excel-compatible charting which makes creating, modifying, rendering and interacting with complex charts easier than ever before.

Scalable Reporting

Easily create richly formatted Excel reports without Excel from any ASP.NET, Windows Forms, WPF or Silverlight application.

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com state can be quite complicated. To mitigate the challenges associated with implementing this pattern, C# 2.0 added the yield contextual keyword to make it easier for a class to dictate how the foreach loop iterates over its contents.

To correctly implement the iterator pattern, you need to maintain some internal state to keep track of where you are while enumerating the collection.

Defining an Iterator: Iterators are a means to implement methods of a class, and they're syntactic shortcuts for the more complex enumerator pattern. When the C# compiler encounters an iterator, it expands its contents into CIL code that implements the enumerator pattern. As such, there are no runtime dependencies for implementing iterators. Because the C# compiler handles implementation through CIL code generation, there's no real runtime performance benefit to using iterators. However, there is a substantial programmer productivity gain in choosing iterators over manual implementation of the enumerator pattern. To understand this improvement, I'll first consider how an iterator is defined in code.

Figure 7 **A List of Some C# Keywords Output from the Code in Figure 6**

```
object
byte
uint
ulong
float
char
bool
ushort
decimal
int
sbyte
short
long
void
double
string
```

Figure 8 **Using yield return to Implement an IEnumerable<T> Property**

```
IEnumerable<(string City, string Country)> CountryCapitals
{
  get
  {
    yield return ("Abu Dhabi","United Arab Emirates");
    yield return ("Abuja", "Nigeria");
    yield return ("Accra", "Ghana");
    yield return ("Adamstown", "Pitcairn");
    yield return ("Addis Ababa", "Ethiopia");
    yield return ("Algiers", "Algeria");
    yield return ("Amman", "Jordan");
    yield return ("Amsterdam", "Netherlands");
    // ...
  }
}
```

Iterator Syntax: An iterator provides a shorthand implementation of iterator interfaces, the combination of the IEnumerable<T> and IEnumerator<T> interfaces. **Figure 5** declares an iterator for the generic BinaryTree<T> type by creating a GetEnumerator method (albeit, with no implementation yet).

Yielding Values from an Iterator: The iterator interfaces are like functions, but instead of returning a single value, they *yield* a sequence of values, one at a time. In the case of BinaryTree<T>, the iterator yields a sequence of values of the type argument provided for T. If the nongeneric version of IEnumerator is used, the yielded values will instead be of type object.

To correctly implement the iterator pattern, you need to maintain some internal state to keep track of where you are while enumerating the collection. In the BinaryTree<T> case, you track which elements within the tree have already been enumerated and which are still to come. Iterators are transformed by the compiler into a "state machine" that keeps track of the current position and knows how to "move itself" to the next position.

The yield return statement yields a value each time an iterator encounters it; control immediately returns to the caller that requested the item. When the caller requests the next item, the code begins to execute immediately following the previously executed yield return statement. In **Figure 6**, the C# built-in data type keywords are returned sequentially.

The results of **Figure 6** appear in **Figure 7**, which is a listing of the C# built-in types.

Clearly, more explanation is required but I'm out of space for this month so I'll leave you in suspense for another column. Suffice it to say, with iterators you can magically create collections as properties, as shown in **Figure 8**—in this case, relying on C# 7.0 tuples just for the fun of it. For those of you wanting to look ahead, you can check out the source code or take a look at Chapter 16 of my "Essential C#" book.

Wrapping Up

In this column, I stepped back to functionality that's been part of C# since version 1.0 and hasn't changed much since the introduction of generics in C# 2.0. Despite the frequent use of this functionality, however, many don't understand the details of what's taking place internally. I then scratched the surface of the iterator pattern—leveraging the yield return construct—and provided an example.

Much of this column was pulled from my "Essential C#" book (IntelliTect.com/EssentialCSharp), which I'm currently in the midst of updating to "Essential C# 7.0." For more information, check out Chapters 14 and 16. ■

Mark Michaelis *is founder of IntelliTect, where he serves as its chief technical architect and trainer. For nearly two decades he has been a Microsoft MVP, and a Microsoft Regional Director since 2007. Michaelis serves on several Microsoft software design review teams, including C#, Microsoft Azure, SharePoint and Visual Studio ALM. He speaks at developer conferences and has written numerous books including his most recent, "Essential C# 6.0 (5th Edition)" (itl.tc/EssentialCSharp). Contact him on Facebook at facebook.com/Mark.Michaelis, on his blog at IntelliTect.com/Mark, on Twitter: @markmichaelis or via e-mail at mark@IntelliTect.com.*

Thanks to the following IntelliTect technical experts for reviewing this article: *Kevin Bost, Grant Erickson, Chris Finlayson, Phil Spokas and Michael Stokesbary*

msdn magazine

MOBILE.
TABLET.
DESKTOP.
PRINT.

WHERE YOU NEED US MOST.

MSDN.MICROSOFT.COM

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

CHICAGO

SEPTEMBER 18 - 21, 2017

DOWNTOWN MARRIOTT

GOLD SPONSOR

Syncfusion

SUPPORTED BY

Visual Studio

msdn magazine

Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA INC

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

Development Topics Include:

- Visual Studio / .NET Framework
- JavaScript / HTML5 Client
- Native Client
- Software Practices
- Database and Analytics
- Angular JS
- ASP.NET / Web Server
- Agile
- ALM / DevOps
- Cloud Computing
- Windows Client
- Xamarin

REGISTER NOW

Register Today and Save $300!

Use promo code VSLCHI2

ROCK YOUR CODE TOUR 2017

"I loved that this conference isn't a sales pitch. It's actual developers giving the sessions - I felt the classes were really geared towards me!"
- Jon Plater, Kantar Retail

"The speakers were clearly experts. Nice breadth of topics. Great hotel; great location."
- Fred Kauber, Tranzact

vslive.com/chicagomsdn

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

Exploring the Map Control

Maps are truly one of mobile devices' great features. With a smartphone in hand, you can expertly navigate your way through a new city, find all the great places to eat, get real-time updates to traffic conditions, even find a new route—if a faster one is available. Mapping can also help your users visualize data points or provide a quick search of local services.

Maps have truly changed the way users interact with their devices and their world. Fortunately, the Universal Windows Platform (UWP) comes equipped with a fully functional Map control that leverages the power and imagery of Bing's mapping services. In this month's column, I'll explore Map control and show just how easy it is to add to your apps.

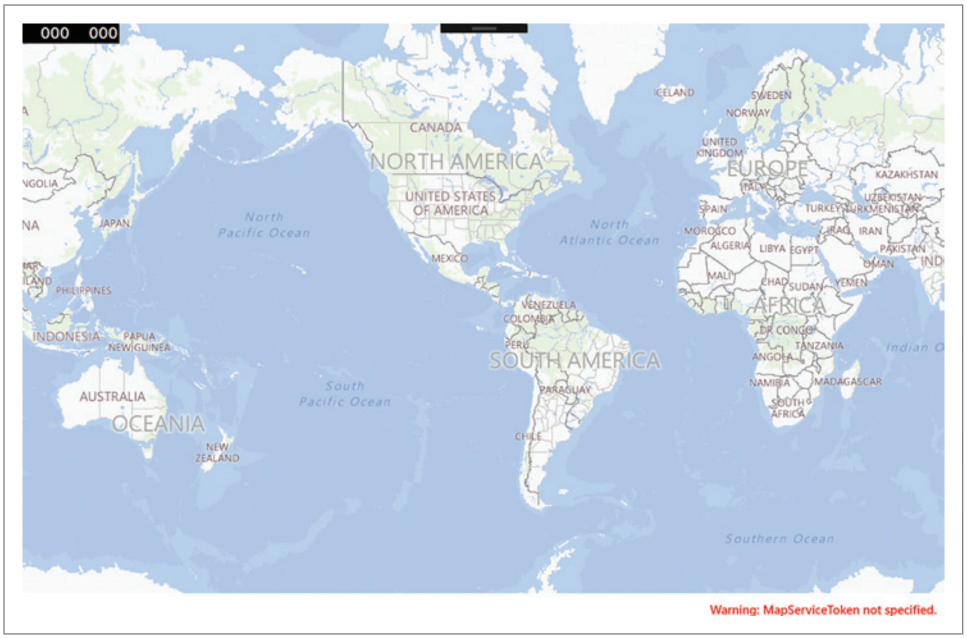


Figure 1 **Basic Map Control**

Setting Up the Project

Create a new blank UWP project in Visual Studio by choosing New Project from the File menu. Expand the Installed Templates | Windows | Blank App (Universal Windows). Name the project MapControl and then click OK. Immediately afterward, a dialog box will appear asking you which version of Windows the app should target. For this project, the default options will be fine, so you can just click OK.

Open the MainPage.xaml file and add the following namespace declaration to the Page tag:

```
xmlns:maps="using:Windows.UI.Xaml.Controls.Maps"
```

This makes the elements located in the Windows.UI.Xaml.Controls.Maps namespace addressable to the XAML elements contained in the Page control by using the maps: prefix. For a detailed explanation of XAML Namespaces and Namespace Mapping, Windows Dev Center has an in-depth article on the matter at bit.ly/2jwcO2D.

Code download available at bit.ly/MapControl.

Adding the Map Control

To add a map to this app, simply add the following XAML:

```
<maps:MapControl x:Name="mapControl" >
</maps:MapControl>
```

Maps have truly changed the way users interact with their devices and their world.

Run the solution by pressing F5 or choosing Start Debugging from the Debug menu. Your app should look something like **Figure 1**. Note that you can click and drag around the map to navigate it. Double-clicking will zoom in on a point. Additionally, touch gestures also manipulate the Map control. You'll also notice red text in the lower-left corner of the control stating, "Warning: MapServiceToken not specified."

The Map control uses imagery and other services, such as traffic data, from Bing Maps. Accordingly, the Bing team would like to

FASTEST WAY TO BUILD HIGH-PERFORMING, FEATURE COMPLETE APPLICATIONS

Easy-to-use UI controls trusted by developers and enterprises worldwide

Windows, Web, and Mobile UI Controls, Reporting, and Productivity Tools

Global Xamarin Dev Days Partner

Visual Studio 2017 Extension Sim-Ship Partners

Extensible controls with easy-to-use universal API

Industry's best high-performance grids, charts, and reports

Small footprint reduces app bloat

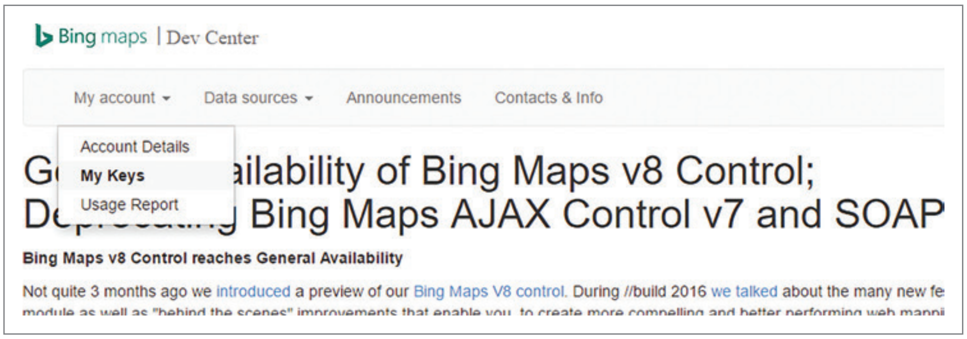
Global support and community

ComponentOne Studio

Download your free 30-day trial at
www.ComponentOne.com

GrapeCity

have the ability to audit which accounts drive usage. This helps limit abuse and keep costs low for everyone. To remove the warning message, you'll need to obtain a MapServiceToken from the Bing Maps Dev Center. In most instances, usage of Bing Maps services will be free. For pricing details, refer to the pricing chart for Basic and Enterprise keys at bit.ly/2j6o96x.

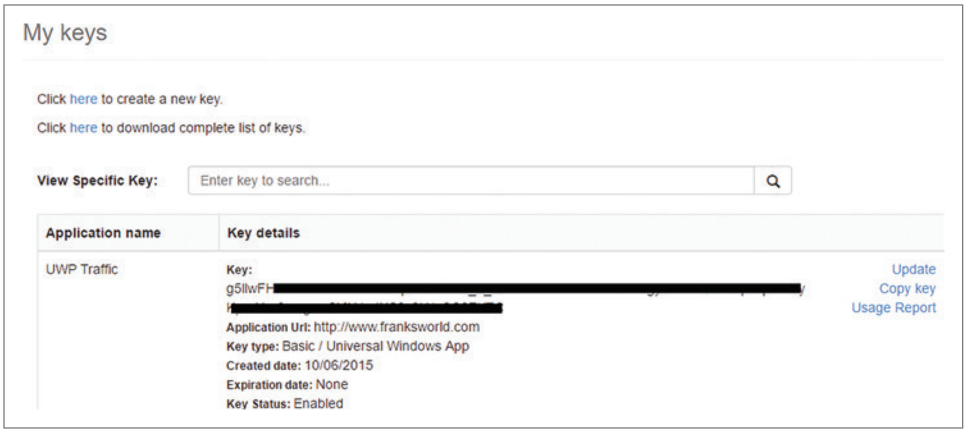


G ailability of Bing Maps v8 Control;
D Bing Maps AJAX Control v7 and SOAP

Bing Maps v8 Control reaches General Availability

Not quite 3 months ago we introduced a preview of our Bing Maps V8 control. During //build 2016 we talked about the many new fe

Figure 2 **Accessing My Keys on the Bing Maps Dev Center**


My keys

Click here to create a new key.

Click here to download complete list of keys.

View Specific Key: Enter key to search...

Application name	Key details	
UWP Traffic	**Key:** g5llwFH[redacted] **Application Url:** http://www.franksworld.com **Key type:** Basic / Universal Windows App **Created date:** 10/06/2015 **Expiration date:** None **Key Status:** Enabled	Update Copy key Usage Report

Figure 3 **My Keys Page with Listing of Previously Registered Keys**

Registering for a Bing Maps Service Token

In a browser, go to bingmapsportal.com, click sign in, and use your Microsoft account credentials to log in. On the landing page, click the My account menu and choose My Keys, as shown in **Figure 2**.

If you've registered keys with Bing Maps before, then they'll appear in a list, as shown in **Figure 3**. To create a new key, click on the option that states "Click here to create a new key."

A form will appear asking for information about the application for which the key will be created (see **Figure 4**). Name the app, choose Basic for Key type, and Universal Windows App for Application type. The Application URL is optional. Click create and you'll be presented with a new key. Copy it and return to the MainPage.xaml file. Modify the XAML to add the MapServiceToken attribute and paste in the key as its value, like so:

```
<maps:MapControl x:Name="mapControl" MapServiceToken="{Insert Key Here}" >
  </maps:MapControl>
```

Run the solution again and the warning message is gone.

Controlling the Map with Code

As you can see, adding a Map control to a UWP app is fairly easy. The control even comes with the ability for users to manipulate the map's center point and zoom level with mouse or touch. The map would be more useful if it could be manipulated in code. For instance, it would be useful to change the style of the map, so that it can display aerial imagery or terrain data and not just the default road view shown in **Figure 1**.

Changing Map Styles

The Map control has a Style property of type MapStyle, which is an enumeration. For example, to change the map to show aerial imagery with roads superimposed on them, all it takes is this line of code:

```
mapControl.Style = MapStyle.AerialWithRoads;
```

To implement the option of choosing any available MapStyle, the app will need some more controls. Add the following Grid definitions to the Grid on the Page control in the MainPage.xaml file to create some space for the controls you're about to add:

```
<Grid.RowDefinitions>
  <RowDefinition Height="50*"/>
  <RowDefinition Height="50*"/>
  <RowDefinition Height="577*"/>
</Grid.RowDefinitions>
```

Next, add the Grid.Row attribute to the Map control and set its value to 2.

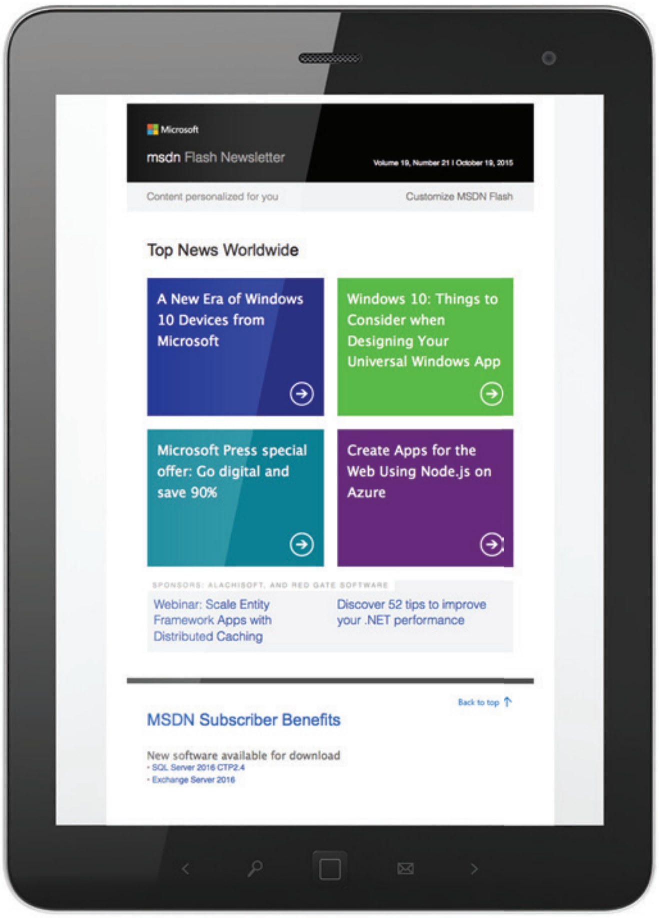
The Map control uses imagery and other services from Bing Maps.

Now, add the following Button controls in a StackPanel:

```
<StackPanel Orientation="Horizontal" Grid.Row="0">
  <Button Content="Roads" Click="Button_Click" Margin="3" />
  <Button Content="Aerial" Click="Button_Click" Margin="3" />
  <Button Content="Aerial With Roads" Click="Button_Click" Margin="3"/>
  <Button Content="Terrain" Click="Button_Click" Margin="3" />
  <Button Content="3D Aerial" Click="Button_Click" Margin="3" />
  <Button Content="3D Aerial With Roads" Click="Button_Click" Margin="3"/>
</StackPanel>
```

In the MainPage.xaml.cs file, add the event handler code in **Figure 5** to determine which button was clicked and then set the map to the appropriate style.

Run the app now and explore the map using the different styles available. Also note that when using one of the 3D views that the control, once again, handles all mouse and touch functions automatically.



Microsoft

msdn Flash Newsletter

Volume 19, Number 21 | October 19, 2015

Content personalized for you

Customize MSDN Flash

Top News Worldwide

A New Era of Windows 10 Devices from Microsoft

Windows 10: Things to Consider when Designing Your Universal Windows App

Microsoft Press special offer: Go digital and save 90%

Create Apps for the Web Using Node.js on Azure

SPONSORS: ALACHISOFT, AND RED GATE SOFTWARE

Webinar: Scale Entity Framework Apps with Distributed Caching

Discover 52 tips to improve your .NET performance

Back to top

MSDN Subscriber Benefits

New software available for download
- SQL Server 2016 CTP2.4
- Exchange Server 2016

Get news from MSDN in your inbox!

Sign up to receive **MSDN FLASH**, which delivers the latest resources, SDKs, downloads, partner offers, security news, and updates on national and local developer events.

msdn magazine

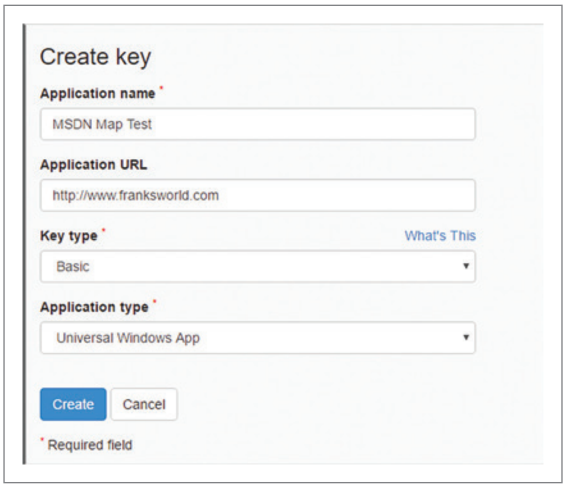
msdn.microsoft.com/flashnewsletter

Zooming In and Out

By default, the map will be fully zoomed out, providing a view of the entire Earth. That might not be useful for a lot of purposes. The Map control has a property of type Double called ZoomLevel to set how far zoomed in or out the map will appear. The value ranges from 1 to 20, with 1 being zoomed in all the way and 20 showing the entire world. Values outside this range are ignored.

Adding zoom in and zoom out to the app is quite easy. First, add the following XAML elements to the MainPage.xaml file inside the Grid control:

```
<StackPanel Orientation="Horizontal" Grid.Row="1">
  <Button x:Name="btnZoomIn" Content="Zoom In" Click="btnZoomIn_Click"
    Margin="3" />
  <Button x:Name="btnZoomOut" Content="Zoom Out" Click="btnZoomOut_Click"
    Margin="3" />
</StackPanel>
```


Create key

Application name *

MSDN Map Test

Application URL

http://www.franksworld.com

Key type * What's This

Basic

Application type *

Universal Windows App

* Required field

Figure 4 **Create Key Form**

Another useful feature of mapping software is the ability to place markers on various points of interest.

Next, add the following two event handlers for the two buttons:

```
private void btnZoomIn_Click(object sender, RoutedEventArgs e)
{
  mapControl.ZoomLevel = mapControl.ZoomLevel + 1;
}
private void btnZoomOut_Click(object sender, RoutedEventArgs e)
{
  mapControl.ZoomLevel = mapControl.ZoomLevel - 1;
}
```

Placing Markers on Maps

Another useful feature of mapping software is the ability to place markers on various points of interest. This could be a great way to point out local attractions. The Map control makes this easy, as well. For this example, place a marker on the Washington Monument in Washington, D.C. The Washington Monument's geospatial coordinates are 38.8895 north and 77.0353 west. To do this, add the following code to the MainPage.xaml.cs file:

```
private void LoadMapMarkers()
{
  MapIcon landmarkMapIcon = new MapIcon();
  landmarkMapIcon.Location = new Geopoint(
    new BasicGeoposition() { Latitude = 38.8895, Longitude = -77.0353 } );
  landmarkMapIcon.Title = "Washington Monument";
  mapControl.MapElements.Add(landmarkMapIcon);
}
```

Next, call the function from the MainPage constructor method.

Run the app now and you'll see a marker has been placed on the map in Washington, D.C., exactly where the Washington Monument sits.

Latitude and Longitude

All points on Earth can be represented by a coordinate system that measures the angular distance from the equator and the Prime Meridian in Greenwich, England. Degrees latitude measures north and south in degrees from -90 at the South Pole to 90 at the North Pole. Degrees longitude measures east or west in degrees from -180 to 180.

Viewing Traffic Data

Bing tracks traffic flow data from cities around the world and the Map control can overlay this data onto a map very easily. To add this feature to the app, add the following markup to the MainPage.xaml file right after the Zoom Out button:

```
<CheckBox Checked="CheckBox_Checked" Unchecked="CheckBox_Unchecked"
  Margin="3">Show Traffic</CheckBox>
```

Then add the following event handlers to the MainPage.xaml.cs file:

```
private void CheckBox_Checked(object sender, RoutedEventArgs e)
{
  mapControl.TrafficFlowVisible = true;
}

private void CheckBox_Unchecked(object sender, RoutedEventArgs e)
{
  mapControl.TrafficFlowVisible = false;
}
```

Run the app, check the Show Traffic CheckBox, and zoom into any major city. Traffic data will only be displayed when the zoom level is at 8 or greater.

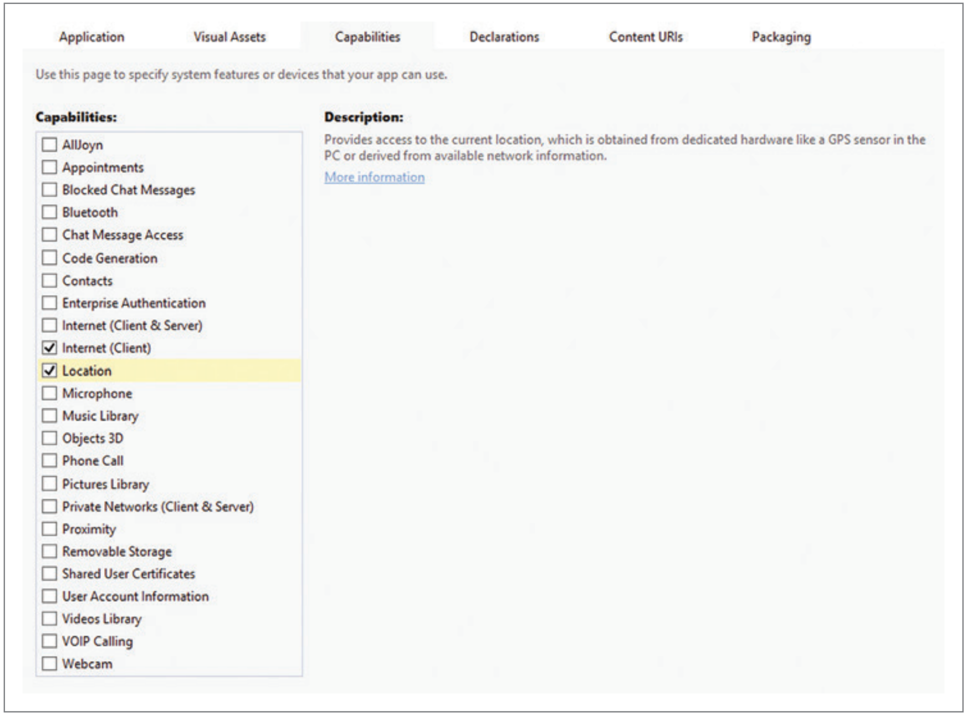
Geolocation

Now the app has several controls to change the style of the map, zoom in or out, and even display traffic flow data. It would be

Figure 5 **Event Handler Code**

```
private void Button_Click(object sender, RoutedEventArgs e)
  {
    var senderButton = sender as Button;
    var buttonTest = senderButton.Content.ToString();

    switch (buttonTest)
    {
      case "Aerial":
      mapControl.Style = MapStyle.Aerial;
      break;
    case "Aerial With Roads":
    mapControl.Style = MapStyle.AerialWithRoads;
    break;
    case "Terrain":
      mapControl.Style = MapStyle.Terrain;
      break;
    case "3D Aerial":
      mapControl.Style = MapStyle.Aerial3D;
      break;
    case "3D Aerial With Roads":
      mapControl.Style = MapStyle.Aerial3DWithRoads;
      break;
    default:
    mapControl.Style = MapStyle.Road;
    break;
  }
}
```



Use this page to specify system features or devices that your app can use.

Capabilities:

- [ ] AllJoyn
- [ ] Appointments
- [ ] Blocked Chat Messages
- [ ] Bluetooth
- [ ] Chat Message Access
- [ ] Code Generation
- [ ] Contacts
- [ ] Enterprise Authentication
- [ ] Internet (Client & Server)
- [x] Internet (Client)
- [x] Location
- [ ] Microphone
- [ ] Music Library
- [ ] Objects 3D
- [ ] Phone Call
- [ ] Pictures Library
- [ ] Private Networks (Client & Server)
- [ ] Proximity
- [ ] Removable Storage
- [ ] Shared User Certificates
- [ ] User Account Information
- [ ] Videos Library
- [ ] VOIP Calling
- [ ] Webcam

Description:

Provides access to the current location, which is obtained from dedicated hardware like a GPS sensor in the PC or derived from available network information.

More information

Figure 6 **Enabling the Location Capability for the App**

helpful for the map control to initialize near the user's current location. That way, users can quickly get a view of their local surroundings, as well as traffic conditions nearby. To do this, you'll need to enable the Location capability in the Package.appxmanifest file, as shown in **Figure 6**. In Visual Studio, double-click on the Package.appxmanifest file in the Solution Explorer pane. In the editor, make sure the checkbox next to Location is checked. Save the file. Now, the app has the Location Capability.

In the MainPage.xaml file add the following Button control:

```
<Button x:Name="btnCurrentLocation" Content="Current Location"
  Click="btnCurrentLocation_Click" Margin="3"/>
```

And add the following event handler in the MainPage.xaml.cs file:

```
private async void btnCurrentLocation_Click(object sender, RoutedEventArgs e)
{
  var locationAccessStatus = await Geolocator.RequestAccessAsync();
  if (locationAccessStatus == GeolocationAccessStatus.Allowed)
  {
    Geolocator geolocator = new Geolocator();
    Geoposition currentPosition = await geolocator.GetGeopositionAsync();
    mapControl.Center = new Geopoint(new BasicGeoposition()
    {
      Latitude = currentPosition.Coordinate.Latitude,
      Longitude = currentPosition.Coordinate.Longitude
    });
    mapControl.ZoomLevel = 12;
  }
}
```

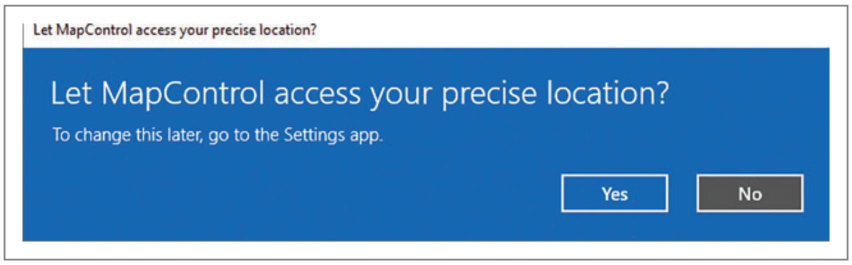
The btnCurrentLocation_Click method first queries the GeoLocator object for what the access status the app has to it. Upon the first time the app runs on the user's system, the system will display a dialog box similar to the one shown in **Figure 7**. If the user clicks on Yes, the code gets the current position from the Geolocator. Afterward, the code centers the map on the latitude and longitude provided. If the Location capability hasn't been added, the user wouldn't have even been prompted with the dialog.

Run the app now, click on the Current Location button, click on Yes if prompted. You should see the map focus in on, or very near, your current location. Accuracy will depend on a number of factors. If your device is connected to a GPS receiver, then your accuracy will be very high—likely down to around 10 meters. If your device lacks a GPS receiver, the GeoLocator relies on other means to determine location, such as using Wi-Fi. For more information on how Wi-Fi positioning works, refer to the Wikipedia entry at bit.ly/2jWvmWM. If Wi-Fi is unavailable, the GeoLocator will use the client device's IP address to determine location, where VPN connections and other factors can significantly lower accuracy.

Maps are truly one of the more essential features of mobile devices.

Wrapping Up

Maps are truly one of the more essential features of mobile devices. Many apps could benefit from the addition of a Map control to visualize data or assist users in finding nearby services. In this column, you saw just how easy it is to add a map to your application and that, in most cases, access to the mapping services comes at no cost to you, the developer. ■



Let MapControl access your precise location?

Let MapControl access your precise location?

To change this later, go to the Settings app.

Figure 7 **Asking the User for Permission to Access Precise Location Information**

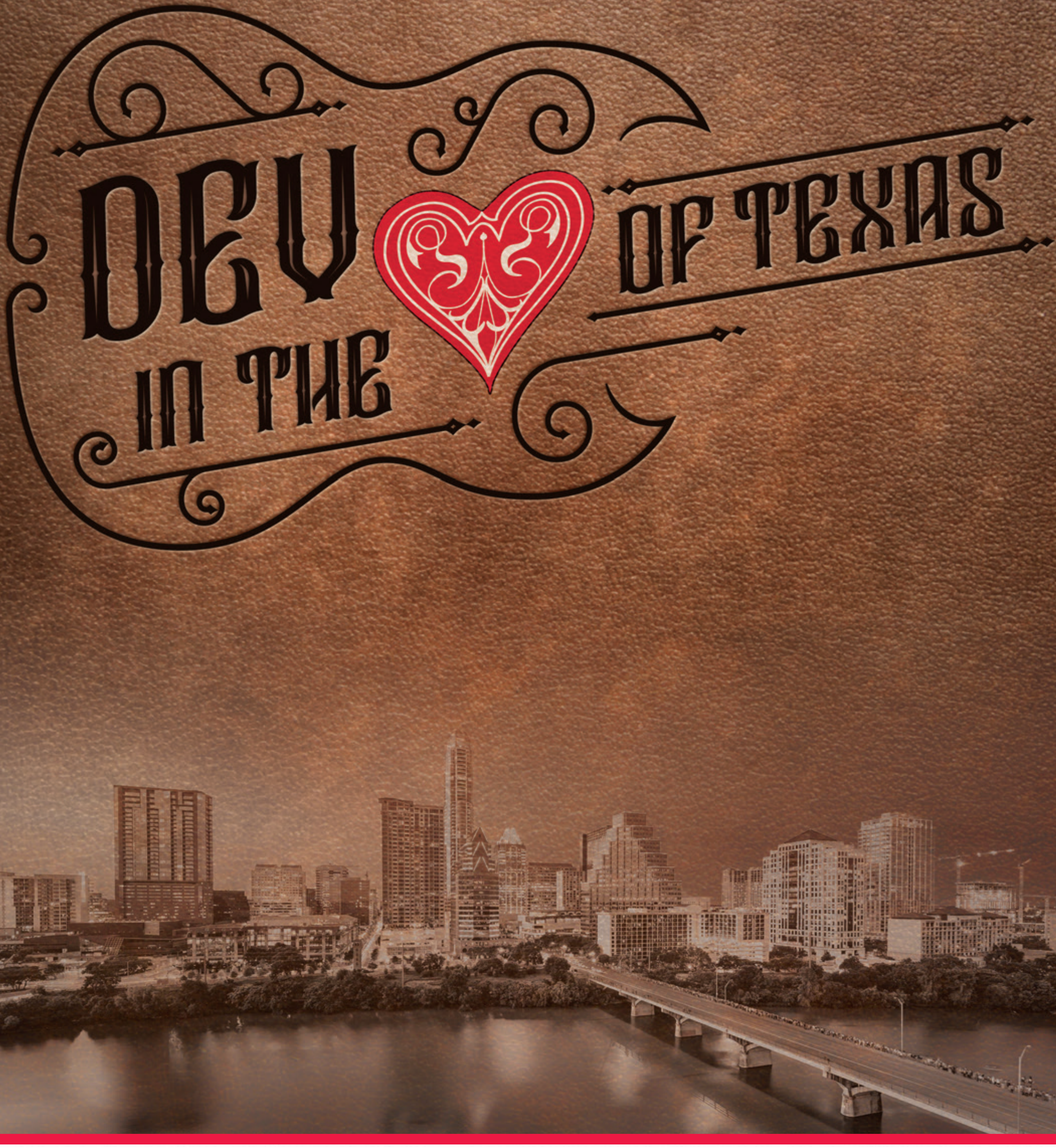
Frank La Vigne *is an independent consultant, where he helps customers leverage technology in order to create a better community. He blogs regularly at FranksWorld.com and has a YouTube channel called Frank's World TV (FranksWorld.TV).*

Thanks to the following technical experts for reviewing this article: *Rachel Appel*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

AUSTIN, TX
MAY 15-18, 2017
HYATT REGENCY



EVENT PARTNER Microsoft

GOLD SPONSORS Progress Syncfusion

SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA YOUR GROWTH. OUR BUSINESS.

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

TURN THE PAGE FOR MORE EVENT DETAILS →

Development Topics include:

- ➤ Visual Studio / .NET
- ➤ JavaScript / HTML5
- ➤ Angular
- ➤ Native Mobile & Xamarin
- ➤ Software Practices
- ➤ Database and Analytics
- ➤ ASP.NET Core
- ➤ Web API
- ➤ ALM / DevOps
- ➤ Cloud Computing
- ➤ UWP
- ➤ Unit Testing

FRIDAY, MAY 19: POST-CON HANDS-ON LABS

Choose From:

- ➤ Angular
- ➤ ASP.NET Core MVC/ Entity Framework

NEW! Only $645!

SPACE IS LIMITED

REGISTER NOW

Register by April 12 and Save $200!

Use promo code VSLAPR4

*Available on 4 day packages; some restrictions apply.

ROCK YOUR CODE TOUR 2017

AUSTIN	WASH. DC	REDMOND	CHICAGO	ANAHEIM	ORLANDO
MAY 15-18	JUNE 12-15	AUG 14-18	SEPT 18-21	OCT 16-19	NOV 12-17

vslive.com/austinmsdn

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

AUSTIN, TX
MAY 15-18, 2017
HYATT REGENCY

REGISTER NOW

Register by April 12 and Save $200!

Use promo code VSLAPR4

*Available on 4 day packages; some restrictions apply.

"The sessions were packed full of extremely valuable info. Austin is a killer location for a conference!!"

– Aaron Eversole, Software Solutions Integrated, LLC

AUSTIN AGENDA AT-A-GLANCE

ALM / DevOps	Cloud Computing	Database and Analytics	Native Client	Software Practices	Visual Studio / .NET Framework	Web Client	Web Server

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, May 15, 2017 *(Separate entry fee required)*		
9:00 AM	6:00 PM	**M01** Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka*	**M02** Workshop: Practical ASP.NET DevOps with VSTS or TFS - *Brian Randell*	**M03** Workshop: Building for the Internet of Things: Hardware, Sensors & the Cloud - *Nick Landry*
6:45 PM	9:00 PM	Dine-A-Round		

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, May 16, 2017			
8:00 AM	9:00 AM	**KEYNOTE:** To Be Announced - *Yochay Kiriaty, Principal Program Manager, Microsoft*			
9:15 AM	10:30 AM	**T01** Go Mobile With C#, Visual Studio, and Xamarin - *James Montemagno*	**T02** Angular 2 – The 75-Minute Crash Course - *Chris Klug*	**T03** What's New in Azure IaaS v2 - *Eric D. Boyd*	**T04** Tour of Visual Studio 2017 - *Jason Bock*
10:45 AM	12:00 PM	**T05** Building Connected and Disconnected Mobile Apps - *James Montemagno*	**T06** Enriching MVC Sites with Knockout JS - *Miguel Castro*	**T07** Bots are the New Apps: Building Bots with ASP.NET WebAPI & Language Understanding - *Nick Landry*	**T08** Getting to the Core of .NET Core - *Adam Tuliper*
12:00 PM	1:30 PM	Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** Lessons Learned from Real World Xamarin.Forms Projects - *Nick Landry*	**T10** Explore Web Development with Microsoft ASP.NET Core 1.0 - *Mark Rosenberg*	**T11** To Be Announced	**T12** Tactical DevOps with VSTS - *Brian Randell*
3:00 PM	4:15 PM	**T13** Xamarin vs. Cordova - *Sahil Malik*	**T14** Assembling the Web - A Tour of WebAssembly - *Jason Bock*	**T15** Cloud Oriented Programming - *Vishwas Lele*	**T16** Exploring C# 7 New Features - *Adam Tuliper*
4:15 PM	5:30 PM	Welcome Reception			

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, May 17, 2017			
8:00 AM	9:15 AM	**W01** To Be Announced	**W02** Introduction to Writing Custom Angular (Not 2.0) Directives - *Miguel Castro*	**W03** Microservices with Azure Container Service & Service Fabric - *Vishwas Lele*	**W04** Database Continuous Integration - *Steve Jones*
9:30 AM	10:45 AM	**W05** Write Once, Run Everywhere - Cordova, Electron, and Angular2 - *Sahil Malik*	**W06** Integrating AngularJS & ASP.NET MVC - *Miguel Castro*	**W07** Go Serverless with Azure Functions - *Eric D. Boyd*	**W08** Continuous Integration and Deployement for Mobile Using Azure Services - *Kevin Ford*
11:00 AM	12:00 PM	**GENERAL SESSION:** .NET in 2017 - *Jeffrey T. Fritz, Senior Program Manager, Microsoft*			
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**W09** Building Cross-Platform Business Apps with CSLA. NET - *Rockford Lhotka*	**W10** Angular 2, ASP.NET Core and Gulp – Happily Forever After, or the Beginning of an Apocalypse? - *Chris Klug*	**W11** Using Cognitive Services in Business Applications - *Michael Washington*	**W12** .NET Deployment Strategies: The Good, The Bad, The Ugly - *Damian Brady*
3:00 PM	4:15 PM	**W13** Creating Great Looking Android Applications Using Material Design - *Kevin Ford*	**W14** Use Docker to Develop, Build, and Deploy Applications, A Primer - *Mark Rosenberg*	**W15** Busy Developer's Guide to the Clouds - *Ted Neward*	**W16** PowerShell for Developers - *Brian Randell*
4:30 PM	5:45 PM	**W17** SOLID – The Five Commandments of Good Software - *Chris Klug*	**W18** JavaScript for the C# (and Java) Developer - *Philip Japikse*	**W19** Introduction to Azure Machine Learning Studio (for the non-Data Scientist) - *Michael Washington*	**W20** Brownfields DevOps in Practice - *Damian Brady*
7:00 PM	9:00 PM	Rollin' On the River Bat Cruise			

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, May 18, 2017			
8:00 AM	9:15 AM	**TH01** A Developers Introduction to HoloLens - *Billy Hollis*	**TH02** Extend and Customize the Visual Studio Environment - *Walt Ritscher*	**TH03** Entity Framework Core for Enterprise Applications - *Benjamin Day*	**TH04** Agile Failures: Stories From The Trenches - *Philip Japikse*
9:30 AM	10:45 AM	**TH05** Unity for .NET Developers - The Time is Now! - *John Alexander*	**TH06** SASS and CSS for Developers - *Robert Boedigheimer*	**TH07** A Tour of SQL Server 2016 Security Features - *Steve Jones*	**TH08** Distributed Architecture: Microservices and Messaging - *Rockford Lhotka*
11:00 AM	12:15 PM	**TH09** Take the Tests: Can You Evaluate Good and Bad Designs? - *Billy Hollis*	**TH10** Debugging Your Website with Fiddler and Chrome Developer Tools - *Robert Boedigheimer*	**TH11** Busy Developer's Guide to NoSQL - *Ted Neward*	**TH12** Top 10 Ways to Go from Good to Great Scrum Master - *Benjamin Day*
12:15 PM	1:30 PM	Lunch			
1:30 PM	2:45 PM	**TH13** Using Visual Studio to Scale your Enterprise - *Richard Hundhausen*	**TH14** Building Single Page Web Applications Using Aurelia.js and the MVVM Pattern - *Ben Hoelting*	**TH15** Azure Data Lake - *Michael Rys*	**TH16** Care and Feeding of Your Product Owner - *John Alexander*
3:00 PM	4:15 PM	**TH17** Windows Package Management with NuGet and Chocolatey - *Walt Ritscher*	**TH18** Tools for Modern Web Development - *Ben Hoelting*	**TH19** U-SQL Killer Scenarios: Performing Advanced Analytics and Big Cognition at Scale with U-SQL - *Michael Rys*	**TH20** Stop the Waste and Get Out of (Technical) Debt - *Richard Hundhausen*

START TIME	END TIME	Full Day Hands-On Labs: Friday, May 19, 2017 *(Separate entry fee required)*	
8:00 AM	5:00 PM	**HOL01** Full Day Hands-On Lab: Busy Developer's HOL on Angular - *Ted Neward*	**HOL02** Full Day Hands-On Lab: Develop an ASP.NET Core MVC/Entity Framework Core App in a Day - *Philip Japikse*

Speakers and sessions subject to change

vslive.com/austinmsdn

Snaglets

I've always admired the genius of Rich Hall, who wrote the book "Sniglets" (Collier Books, 1984) and its successors. In his creation, a sniglet is: "Any word that doesn't appear in the dictionary, but should." For example: "Chwads: discarded gum found beneath tables and countertops."

We geeks are a great source of sniglets, such as "Animousity: vigorously clicking your pointer device because a page is loading too slowly, even though it doesn't do anything to help." I've created my share, with coinages of hassle budget, marketingbozo, and MINFU. (See my April 2013 column, "Coining Currency," at msdn.com/magazine/dn166939).

For this April, I will apply the sniglet principle to our software industry. I do hereby declare and name a new category of thing, which I'll call Snaglets. A Snaglet is an app that doesn't exist, but should. Of course, geeks being geeks, as soon as I publish a Snaglet in this column, someone somewhere is going to write it, just to bust my chops. But herewithin the canonical list of Snaglets at this instant in time:

I do hereby declare and name a new category of thing, which I'll call Snaglets. A Snaglet is an app that doesn't exist, but should.

Roastissuerie: Buzzes every 60 seconds during a phone call to remind you to switch your cell phone to your other hand, so that its microwaves roast both sides of your brain equally. Free version just buzzes every 60 seconds. Paid version keeps track of actual phone position, adjusting intervals to even out any bias.

FakeBit: Shows a phony Fitbit screen claiming that you've been exercising. Free version shows steps and calories. Paid version also allows you to award yourself medals.

Ventrilosnark: Throws a voice making nasty remarks to a specific audience location while a speaker's back is turned. Requires two phones to create phased waveforms to steer sound. Free version uses a random voice. Paid subscription version samples voice of person sitting at the target location and makes snarky remarks in that voice. If your credit card expires, phone makes nasty comments in your voice from your location.

Baby, Maybe: Fertility cycle tracker with random variations built in.

Shut Up, Stella: The perfect gift for anyone who doesn't miss their ex. User specifies name of ex. Program starts to nag when it hears ex's name between alert syllables ("Oh, Stella dear") and continues until user firmly states, "Shut up, [name]." User can select subjects for nagging (for example, drinking too much, farting in bed and so on). Free version uses generic male/female voice. Paid version scours the Web for the particular ex's voice, possibly calling their phone to sample outgoing messages and then nags in that. Inspired by Star Trek episode "I, Mudd."

GPSHutUp: Partner app of "Shut Up, Stella." Uses same free/paid voices. Enter your destination, app gives you directions for the longest, least efficient journey. "No, you idiot, turn right here, not left," which you ignore. Argues back when you insist, "I know where I'm going." When you arrive, grudgingly admits, "Hey, your way really was faster. You're so smart." Even if it wasn't, and you're not.

Fight Night: Starts and continues an argument between automated wizards—Siri, Alexa, Cortana (and Google, if it ever gives her a human name). Choose any two partners, type in a starting insult—"Siri, Cortana says you're rotten to the core"—and off it goes. "Oh yeah? Tell her she's a talking donut, getting stale, and I'm coming over to take a bite." Paid version automatically posts to social media.

DumpCover: App that covers up excretory and related noises when you go to the bathroom during a phone conference and forget to press mute. Paid version automatically mutes the microphone on first detection of any impolite sounds. Free version does that for the first five occurrences, then kicks in after five-second delay.

Scat!: A phone lying on a couch cushion detects when a cat jumps on the good couch, scares cat down with loud barking noises. Free version works at first, but effectiveness decays logarithmically, as cat wises up. Paid version interfaces with Internet of Things smart squirt bottle to provide negative reinforcement.

Plattski: Free version spouts random gibberish sentences, periodically attempts to convince you that they contain profound wisdom. Takes over your phone, won't shut up or uninstall no matter what you do. Paid version (only bitcoins accepted) finally shuts up. ■

DAVID S. *Platt teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

Looking for a powerful distributed cache?

Get the easiest, most powerful in-memory data grid designed to scale your .NET applications.

- Fast and linearly scalable
- Enterprise-grade availability
- Industry-leading ease of use
- Integrated in-memory computing

Go with the industry leader in distributed caching

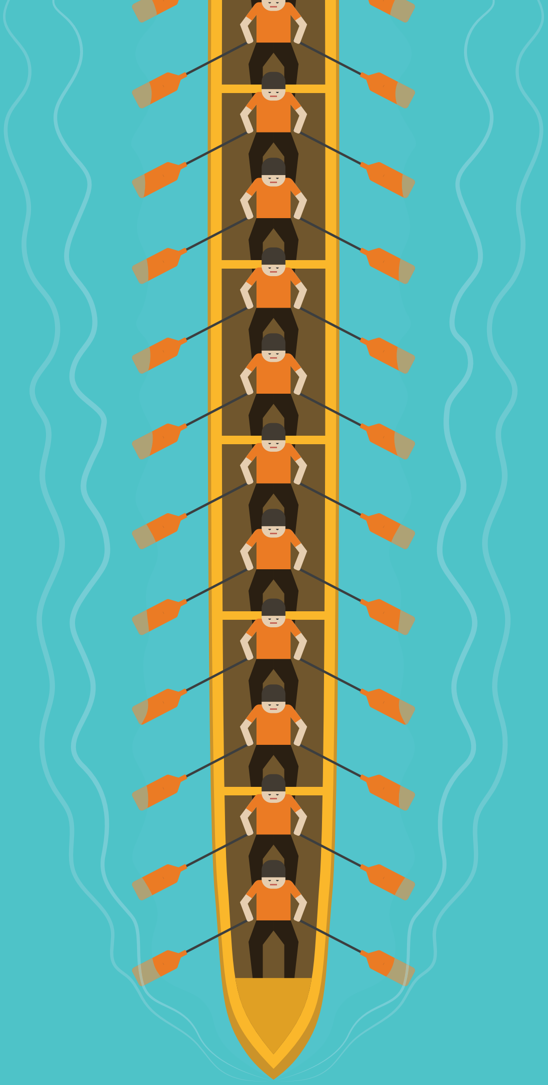
Trusted by hundreds of enterprise customers for more than 12 years, ScaleOut's distributed caching technology delivers rock-solid performance, legendary ease of use, and world-class support. Unlike Redis, it offers a better migration path from AppFabric Caching. Here's why:

	ScaleOut	Redis
Source-code compatible AppFabric Caching APIs for seamless migration	✔	✖
Architected from the ground up for automatic scaling and high availability	✔	✖
Fully coherent data storage and access for mission-critical applications	✔	✖
Advanced .NET features: distributed LINQ query, C# MapReduce, and more	✔	✖

Replacing AppFabric Caching? Trouble switching to Redis?
Check out our AppFabric-compatible drop-in: **www.scaleoutsoftware.com/appfabric**

ScaleOut Software

Please visit **scaleoutsoftware.com** or call us at **+1.503.643.3422** to learn how ScaleOut's technology can help accelerate your business.



Shrink your cost,
not your team

SYNCFUSION'S UNLIMITED FLAT-FEE LICENSE

- Access to 800+ pre-built controls and frameworks.
- Enterprise products including Big Data, Dashboard, and Data Integration Platforms.
- No user count required.
- Eliminate concerns around runtime, per-server fees.
- One low, annual fee!

STARTING
@
$3,995

FIND OUT HOW YOU CAN SAVE

www.syncfusion.com/SDunlimited

Syncfusion®