

msdn magazine



Build Apps for Windows 10.....24

Introducing the FREE DevExpress Grid Built and Optimized for Xamarin.Forms



Get your free copy of the
DevExpress Grid for Xamarin.Forms
devexpress.com/xamarin

The FREE

DevExpress Grid for Xamarin.Forms

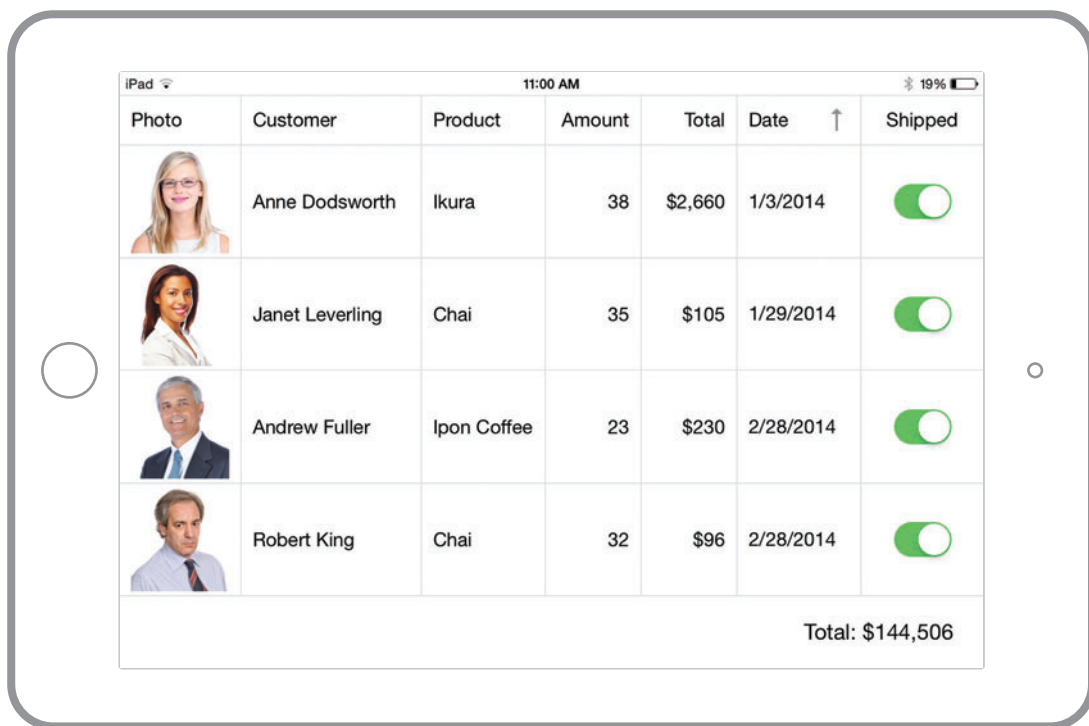






Photo	Customer	Product	Amount	Total	Date	Shipped
	Anne Dodsworth	Ikura	38	\$2,660	1/3/2014	<input checked="" type="checkbox"/>
	Janet Leverling	Chai	35	\$105	1/29/2014	<input checked="" type="checkbox"/>
	Andrew Fuller	Ipon Coffee	23	\$230	2/28/2014	<input checked="" type="checkbox"/>
	Robert King	Chai	32	\$96	2/28/2014	<input checked="" type="checkbox"/>
				Total: \$144,506		

devexpress.com/xamarin

msdn

magazine



Build Apps
for Windows 10.....24

An Introduction to Building Windows Apps for Windows 10 Devices Jerry Nixon and Andy Wigley	24
Analyze Performance While Debugging in Visual Studio 2015 Charles Willis and Dan Taylor	32
Use IntelliTrace to Diagnose Issues Faster Angelos Petropoulos	40
What Every Programmer Should Know About Compiler Optimizations, Part 2 Hadi Brais	52
2D Game Engines for the Web Michael Oneppo	64

COLUMNS

FIRST WORD

Any App, Any Developer
S. Somasegar, page 6

UPSTART

First Day
Ryder Donahue, page 8

WINDOWS WITH C++

Adding Compile-Time Type Checking to Printf
Kenny Kerr, page 12

DATA POINTS

The EF6, EF7 and ASP.NET 5 Soup
Julie Lerman, page 18

MODERN APPS

End-to-End Testing in Modern Web Sites and Apps
Rachel Appel, page 72

DON'T GET ME STARTED

Gone Viral
David Platt, page 80

The preferred choice for Microsoft developers and consultants.

Axosoft helps you and your team make the most of your project management time. Plan your releases effectively, improve your process, and keep everyone in the loop.

Your developers can update their work items and review backlogs from **within Visual Studio**, and **associate TFS commits** with your features and defects.

Spend less time managing your projects and more time writing and shipping code. Get started with Axosoft now!

try us today for free

at axosoft.com/MSDN

popular integrations



slack



Visual Studio



zendesk

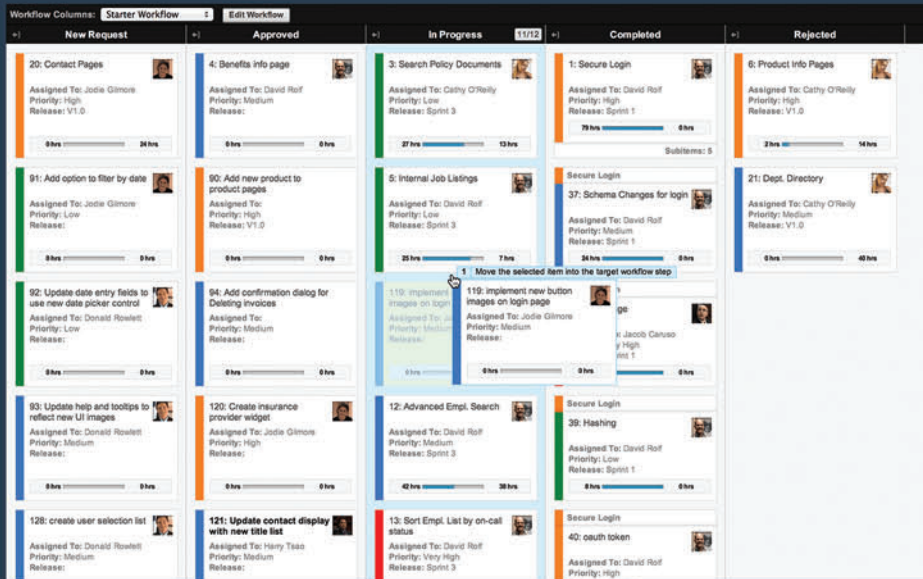
and more



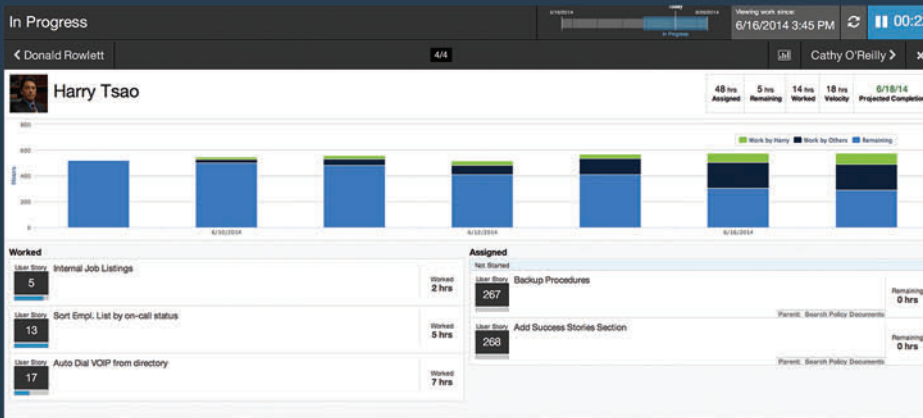
axosoft

800.653.0024

take back
your development time



Check out item status and progress at a glance with our Kanban board view



View team members' work capacity and assignments in our Daily Scrum mode



Automated burndown charts, velocity, projected ship dates and more

msdn magazine

MAY 2015 VOLUME 30 NUMBER 5

Director Keith Boyd
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 Lafe Low
Features Editor Sharon Terdeman
Group Managing Editor Wendy Hernandez
Senior Contributing Editor Dr. James McCaffrey
Contributing Editors Rachel Appel, Dino Esposito, Kenny Kerr, Julie Lerman, Ted Neward, David S. Platt, Bruno Terkaly, Ricardo Villalobos
Vice President, Art and Brand Design Scott Shultz
Art Director Joshua Gould



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
Graphic Designer Erin Horlacher
Senior Graphic Designer Alan Tao
Senior Web Designer Martin Peace

PRODUCTION STAFF

Director, Print Production David Seymour
Print Production Coordinator Anna Lyn Bayaua

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
Senior Site Administrator Shane Lee
Site Administrator Biswarup Bhattacharjee
Senior Front-End Developer Rodrigo Munoz
Junior Front-End Developer Anya Smolinski
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 Bundy
Editorial Director, Custom Content Lee Pender
Senior Program Manager, Client Services & Webinar Production Chris Flack
Project Manager, Lead Generation Marketing Mahal Ramos
Coordinator, Lead Generation Marketing Obum Ukabam

MARKETING

Chief Marketing Officer Carmel McDonagh
Vice President, Marketing Emily Jacobs
Senior Manager, Marketing Christopher Morales

ENTERPRISE COMPUTING GROUP EVENTS

Senior Director, Events Brent Sutton
Senior Director, Operations Sara Ross
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



Chief Executive Officer
Rajeev Kapur
Chief Operating Officer
Henry Allain
Senior Vice President & Chief Financial Officer
Richard Vitale
Executive Vice President
Michael J. Valenti
Vice President, Information Technology & Application Development
Erik A. Lindgren
Chairman of the Board
Jeffrey S. Klein

ID STATEMENT MSDN Magazine (ISSN 1528-4859) is published monthly 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: jloug@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: FirstInitial.LastName@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 contents herein do not necessarily express those of the publisher.





LEADTOOLS® V19

THE WORLD LEADER IN IMAGING SDKs



Version 19 has been crafted with developers in mind. It has been optimized to give you an all-in-one development solution for all your document, medical and multimedia needs.

Document
Imaging SDKs

Medical
Imaging SDKs

Multimedia
Imaging SDKs

- Document Viewer *New*
- Document Converter *New*
- OCR, MICR, OMR & ICR
- Barcode & Forms Recognition
- Create, Load, Save, Edit & View PDF
- Annotations, TWAIN & SVG *New*

- DICOM, CCOW & HL7 *New*
- PACS Client & Server Framework
- DICOM Storage Server Framework
- Web & Desktop Viewers
- Image Processing & Annotations
- Medical 3D (MPR, MIP, VRT)

- Playback, Capture & Convert
- Streaming - MPEG2-TS & RTSP
- MPEG-4, H.264 & H.265 *New*
- DVD & DVR
- Media Foundation & DirectShow
- Distributed Transcoding

.NET Windows API WinRT Linux iOS OS X Android HTML5

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM



SALES@LEADTOOLS.COM
800.637.1840





Introducing Upstart

The times, they are a-changin'.

At the 2000 Microsoft Professional Developers Conference (PDC) 15 years ago, the new Microsoft .NET Framework and the managed languages C# and Visual Basic .NET that went with it were revealed. The announcements reflected an urgent need for Microsoft to appeal to business developers intrigued by the managed Java platform and programming language, and quickly grew into a wildly successful platform serving a broad range of development scenarios.

Fast forward to 2015, and Microsoft faces a new sea change. Web, mobile and cloud have redefined large swaths of the development space. Shops that relied solely on Visual Studio and targeted rich Windows client PCs and servers today build smartphone apps, hybrid Web applications and cloud-savvy services. For Microsoft, the transition to this complex world is reflected in its support for platforms like Apache Cordova, Xamarin and universal Windows apps.

As Keith Boyd, Microsoft principal director and editorial lead at *MSDN Magazine*, told me in an interview: "Microsoft is changing—rapidly. The cultural forces that are shaping our business model are leading to significant changes in the way we run our businesses, and in the kinds of people we hire to help propel us forward."

In fact, says Boyd, young programmers entering the development space might have little knowledge of Visual Studio, the Microsoft .NET Framework or Windows. At the same time, many veteran .NET developers are seeking new skills and tools as they transition to mobile, cloud and Web development. He says Microsoft, and by extension *MSDN Magazine*, must reach out to these coders, which is why this month we are launching a new column, called Upstart, to reflect the challenges and opportunities that face developers entering the field or transitioning into new arenas within it.

In our inaugural Upstart column, Ryder Donahue offers a glimpse of what it's like to step aboard the Microsoft mothership. An accomplished young developer before he arrived at Microsoft, Donahue was a member of the team that won the 2013 U.S. Imagine Cup programming competition and earned a bid to the Worldwide Imagine Cup Finals in St. Petersburg, Russia.

Today, Donahue is a member of the Xbox on Windows team, where he's working on the Xbox App for Windows 10. He's loving the challenge, but those first few months in Redmond were tough. As he writes in this month's column: "For me, it was humbling to go from writing thousands of lines of code a day to just a handful, and sometimes spending half a day trying to get an application to compile."

Those moving into modern app development will benefit from recent tooling advances.

His advice for newly minted developers is equally relevant for veterans migrating to new skill sets. Rein in your expectations, he says, and be prepared to go slow the first few months while you get your bearings. The good news? Those moving into modern app development will benefit from recent tooling advances.

"I feel like one major change that has come to modern app development, compared to how it was even 10 years ago, is how you can create something really great and high quality in a relatively short amount of time. The tools that are given to developers, and the resources available, empower anyone to really make their dream app a reality," he says. "Given my passion is mostly centered around modern app dev, I hope my writing will inspire readers to go out and create their own awesome apps."

Boyd shares that hope, and believes *MSDN Magazine* can play an important role for developers moving into new areas of development.

"Our goal is to continue to use *MSDN Magazine* as our conduit to professional developers—both young and old, experienced and unproven," he says. "I want recently minted grads to be just as excited about receiving the magazine in their mailbox as their more seasoned colleagues."

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

© 2015 Microsoft Corporation. All rights reserved.

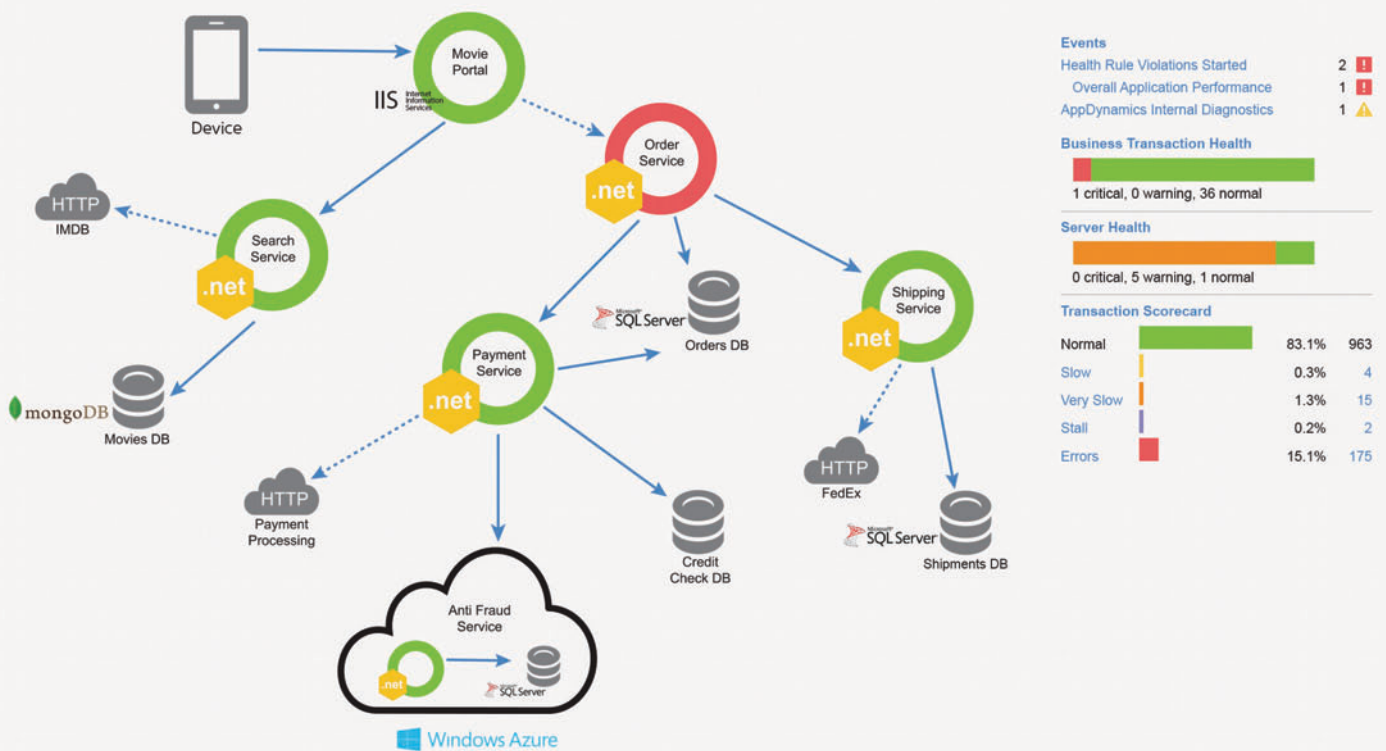
Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN and Microsoft logos are used by 1105 Media, Inc. under license from owner.

There's nothing about .NET that AppDynamics doesn't see.

AppDynamics gives you the visibility to take command of your .NET application's performance, no matter how complicated or distributed your environment is.



When your business runs on .NET and SQL Server, count on AppDynamics to give you the complete visibility you need to be sure they are delivering the performance and business results you need — no matter how complex, distributed or asynchronous your environment, live 'in production' or during development.

See every line of code. Get a complete view of your environment with deep code diagnostics and auto-discovery. Understand performance trends with dynamic baselining. And drastically reduce time to root cause and remediation.

See why the world's largest .NET deployments rely on the AppDynamics Application Intelligence Platform. Sign up for a FREE trial today at www.appdynamics.com/Microsoft.

Any App, Any Developer

It is truly an exciting time to be a developer. As the mobile, cloud and DevOps transitions sweep across the industry, we see developers benefiting from new opportunities to reach customers, access software infrastructure, and take ideas into production, where programmers can learn and adjust faster than ever before.

Last November, at our Connect(); conference in New York, I had the opportunity to talk about how we are opening up our Microsoft development platforms and tools to an increasingly wide base of developers across the industry, as we help organizations make the three important transitions to mobile, cloud and DevOps.

One slide in particular really articulated the significance of the change. It said simply: “Our Vision: Any App, Any Developer.”

Core to this vision is a focus on openness, flexibility and easy interoperation with all of the great progress happening throughout the industry. Today, more than ever, the tools and services we build target a broad array of developer needs—from cross-platform mobile development tools in Visual Studio, to support for a wide range of programming languages, developer stacks and OSes in Microsoft Azure.

Core to this message is a focus on openness, flexibility and easy interoperation with all of the great progress happening throughout the industry.

Last November, we announced our plan to open source .NET Core, and to take it cross-platform to Linux and OS X. Since then, we’ve made some great progress, open sourcing the .NET Core Framework and, more recently, the .NET Core CLR. The open source community has rallied around these projects, contributing hundreds of commits and helping us to bring cross-platform .NET to life. One of many great examples is when we initially released .NET Core CLR with Linux support, the open source community jumped on the opportunity to bring this support to Mac OS X, too. Within a week, this additional support was integrated into the project.

We also released the new Visual Studio Community 2013 in November, a full-featured edition of Visual Studio, available free for non-enterprise usage. Developers have picked up Visual Studio Community 2013 rapidly over the last six months, with more than

2 million downloads and a fast-growing, active developer base. The Community version opens up access to Visual Studio extensions to all users, giving more developers access to the great ecosystem of plug-ins, and giving extension authors access to an even larger base of users. I’ve been excited to see this access to extensions lead new developer communities to embrace Visual Studio Community, with extensions for Unity development and Python development topping the list of most-used extensions.

Development organizations, teams and applications are becoming increasingly heterogeneous, so it’s vital to have great tools that support the breadth of application development needs within a project. Our goal is to deliver tools and services that add value for any team, regardless of the platforms they use in either development or production. From the mixed Java and .NET teams that we work with every day, to companies building for a variety of mobile platforms, to organizations embracing microservices architectures made up of many different technology components, the need is clear: Developer services must span the diversity of work the organization is doing, even as teams combine best-of-breed solutions for each part of the development cycle together into their DevOps practices.

Last month, a handful of folks from my team had the opportunity to go to EclipseCon to talk with Java developers about areas where we’re bringing key Microsoft developer and platform services to the Java space. During the week, we announced our Application Insights SDK for Java, presented sessions on the Azure Toolkit for Eclipse, and highlighted how developers are using Team Foundation Server and Visual Studio Online for Java development today. In each of these cases we’ve been building on open foundations in our products—REST APIs, OAuth, and service hooks that make it easy to connect our developer services into existing development workflows. It might seem strange to some of these developers to see Microsoft taking these steps, but it’s a natural byproduct of our commitment to “any app, any developer.”

This month, at Build, we’ll have the opportunity to talk about the next wave of Microsoft developer platform and tools openness across Windows, Azure, Visual Studio, the .NET Framework and more. It’s an exciting time to be a developer, and I couldn’t be more excited about our work to open up our platforms and tools to any developer working on any application.

Namaste! ■

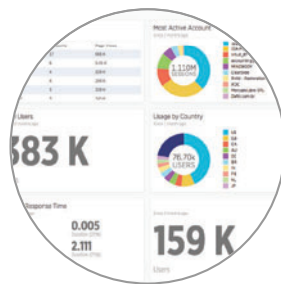
S. SOMASEGAR is the corporate vice president of the Developer Division at Microsoft. He is responsible for developer tools and services, including the programming languages and runtimes designed for a broad base of software developers and development teams, as well as for the Visual Studio, Team Foundation Server, and Visual Studio Online lines of products and services.

One source of truth

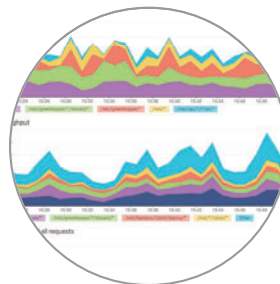
See all your data. Boost performance. Drive accountability for everyone.



Mobile Developers
End-to-end visibility,
24/7 alerting, and
crash analysis.



Front-end Developers
Deep insights into
your browser-side
app's engine.



IT Operations
Faster delivery.
Fewer bottlenecks.
More stability.



App Owners
Track engagement.
Pinpoint issues.
Optimize usability.

Move from finger-pointing blame to data-driven accountability.
Find the truth with a single source of data from multiple views.

newrelic.com/truth

First Day

About six months ago, I graduated from college with a bachelor's in Computer Science. I had some success under my belt, including, but not limited to, being on the winning team of the Microsoft U.S. Imagine Cup, and publishing an app in the Windows Store that has earned more than 400,000 downloads.

When I got a job offer to work at Microsoft, I was excited to join the largest software company in the world. Given my portfolio and experience, I felt prepared for whatever challenge was thrown my way.

How naïve I was.

Every programmer understands the learning curve. Our daily work often involves encountering something unfamiliar or difficult. It becomes a constant war of knowledge acquisition and application. The learning curve I experienced at Microsoft challenged me on a whole new level. I had never seen, let alone worked on, such a large and mature code base that had so much information locked within tribal knowledge.

The development environment alone took a couple of weeks to set up. This included setting up all the necessary software, source enlistsments and, of course, the permissions ... so many permissions.

To give you some perspective, a week before I started working, I was writing about 2,000 lines of quality, production-level lines of code a day. After my first week of work, I had written about 10 lines. Which didn't compile. The development environment alone took a couple of weeks to set up. This included setting up all the necessary software, source enlistsments and, of course, the permissions ... so many permissions.

Given the type of software we write and maintain, I'm not sure if there's a better way to ease the transition into enterprise software development.

This may paint a grim picture for new software engineers entering the professional field, but I can't say enough about the supportive nature of everyone I've worked with at Microsoft. Like any workplace, the people you work with define the environment, and mine has been a blast. I can't stress enough the importance of maintaining a positive attitude, and being able to have some fun both with the project and the team. When you spend more than eight hours a day, five days a week with them—and sometimes, *a lot* more—your team becomes your second family. It's important to get along with them.

This column seeks to offer advice and guidance to developers just getting started, so I'll leave you with some. My biggest regret as a young programmer was not participating in an internship, which is not only often *paid*, but provides invaluable, real-world experience. Had I taken a summer to intern with a company such as Microsoft, I might have learned a lot of lessons before my official career had even begun—and done so under far less pressure.

There are other options. For instance, contributing to a large open source project. You'll enjoy the same shock I encountered, wrangling source code that's likely older than you are and has more contributors than people you've met in your entire life. But you'll get comfortable reading large code bases and that confidence will serve you well when you walk through the door on the first day of your new career.

Also, be prepared to struggle with your own expectations. For me, it was humbling to go from writing thousands of lines of code a day to just a handful, and sometimes spending half a day trying to get an application to compile. This was especially difficult when my teammates were confidently finishing work items left and right. But as a new employee, this is completely normal. Don't be afraid to ask for help. Remember, everyone around you wants to see you succeed. Your teammates have all been through the same experience you're going through and understand that you'll have questions.

When I started at Microsoft I was told it would take three to six months to get grounded and gain a proper sense of direction. That sounded preposterous on my first day, as I had never needed that much time to adjust to anything! Having just passed my seven-month anniversary here, I've come to the conclusion that this bit of advice was spot on. The quicker you acknowledge and embrace the challenge of your new career, the less you'll stress the small things and the more you'll enjoy the journey. ■

RYDER DONAHUE is a software developer engineer at Microsoft. Originally from the Hawaiian Islands, he now resides in Redmond, Wash., with his fiancée and their cat, Marbles.

DOMAINS | MAIL | HOSTING | eCOMMERCE | SERVERS



STARTING AT
\$3.99
/month*

~~\$6.99~~
/month

1&1 UNLIMITED HOSTING

NEW!

NO LIMITS!

UNLIMITED POSSIBILITIES FOR YOUR WEB PROJECTS*

Complete

- Unlimited Webspace
- Unlimited Bandwidth
- Unlimited Traffic
- Unlimited E-mail Accounts
- Unlimited E-mail Storage
- Unlimited Databases
- Unlimited Domains
(One Domain Included)

Reliable

- Geo-redundancy
- Daily Backups
- 1&1 CDN
- 1&1 SiteLock Basic
- 24/7 Support

Easy

- 1&1 Click & Build Apps including WordPress and Joomla!®
- 1&1 Mobile Website Builder



☎ 1 (877) 461-2631

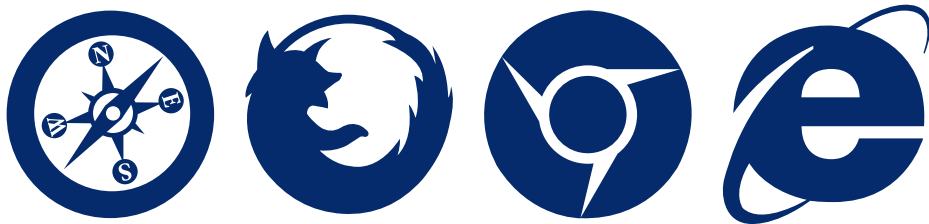


1and1.com

*1&1 Unlimited Hosting is \$3.99/month for twelve months, after which the regular price of \$6.99/month applies. Some features listed are only available with package upgrade. Visit www.1and1.com for full promotional details. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are property of their respective owners. Rubik's Cube® used by permission of Rubik's Brand Ltd. © 2015 1&1 Internet Inc. All rights reserved.

Finally!

Cross-browser,
true WYSIWYG
document editing
and reporting
for ASP.NET

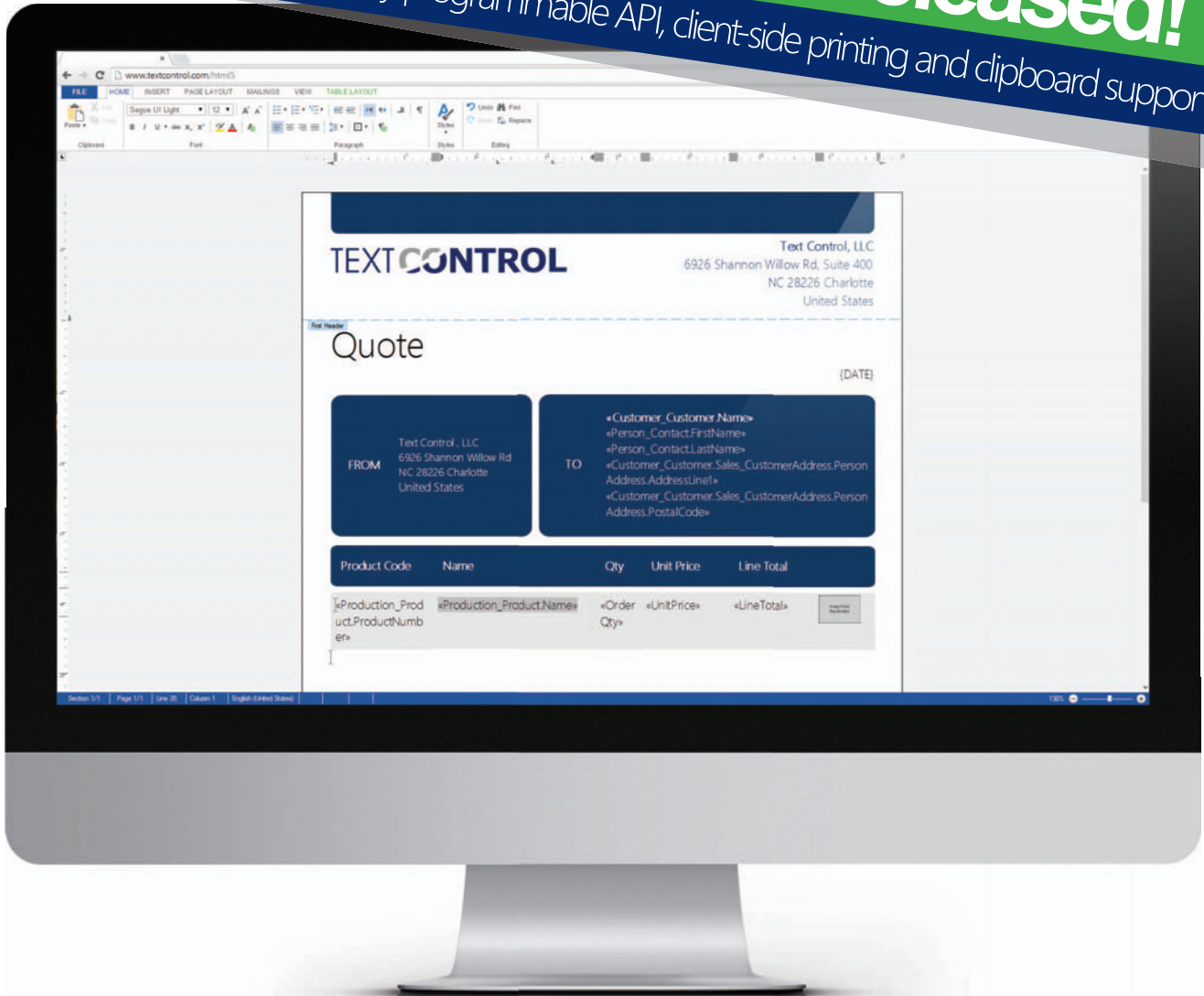


The first true WYSIWYG, HTML5-based web editor and reporting template designer for ASP.NET **WebForms** and **MVC**. Give your users an MS Word compatible editor to create powerful reporting templates anywhere - in any browser.

New version released!

Announcing
X12

Fully programmable API, client-side printing and clipboard support



Edit and create
MS Word
documents



Create and modify
Adobe® PDF
documents



Create reports
and mail merge
templates



Integrate with
Microsoft®
Visual Studio

Live demo and 30-day trial version download at:
www.textcontrol.com/html5

X12

TEXT CONTROL



Adding Compile-Time Type Checking to Printf

I explored some techniques for making `printf` more convenient to use with modern C++ in my March 2015 column (msdn.magazine.com/magazine/dn91318). I showed how to transform arguments using a variadic template in order to bridge the gap between the official C++ string class and the antiquated `printf` function. Why bother? Well, `printf` is very fast, and a solution for formatted output that can take advantage of that while allowing developers to write safer, higher-level code is certainly desirable. The result was a `Print` variadic function template that could take a simple program like this:

```
int main()
{
    std::string const hello = "Hello";
    std::wstring const world = L"World";
    Print("%d %s %ls\n", 123, hello, world);
}
```

And effectively expand the inner `printf` function as follows:

```
printf("%d %s %ls\n", Argument(123), Argument(hello), Argument(world));
```

The `Argument` function template would then also compile away and leave the necessary accessor functions:

```
printf("%d %s %ls\n", 123, hello.c_str(), world.c_str());
```

While this is a convenient bridge between modern C++ and the traditional `printf` function, it does nothing to resolve the difficulties of writing correct code using `printf`. `printf` is still `printf` and I'm relying on the not entirely omniscient compiler and library to sniff out any inconsistencies between conversion specifiers and the actual arguments provided by the caller.

Surely modern C++ can do better. Many developers have attempted to do better. The trouble is that different developers have different requirements or priorities. Some are happy to take a small performance hit and simply rely on `<iostream>` for type checking and extensibility. Others have devised elaborate libraries that provide interesting features that require additional structures and allocations to keep track of formatting state. Personally, I'm not satisfied with solutions that introduce performance and runtime overhead for what should be a fundamental and fast operation. If it's small and fast in C, it should be small, fast and easy to use correctly in C++. "Slower" should not enter into that equation.

So what can be done? Well, a little abstraction is in order, just so long as the abstractions can be compiled away to leave something very close to a handwritten `printf` statement. The key is to realize that conversion specifiers such as `%d` and `%s` are really just placeholders for the arguments or values that follow. The trouble is that these placeholders make assumptions about the types of the corresponding arguments without having any way of knowing whether those types are correct. Rather than trying to add runtime type checking that will confirm these assumptions, let's throw away this pseudo-type

information and instead let the compiler deduce the types of arguments as they're naturally resolved. So rather than writing:

```
printf("%s %d\n", "Hello", 2015);
```

I should instead restrict the format string to the actual characters to output and any placeholders to expand. I might even use the same metacharacter as a placeholder:

```
Write("% %\n", "Hello", 2015);
```

There's really no less type information in this version than in the preceding `printf` example. The only reason `printf` needed to embed that extra type information was because the C programming language lacked variadic templates. The latter is also much cleaner. I would certainly be happy if I didn't have to keep looking up various `printf` format specifiers to make sure I've got it just right.

I also don't want to limit output to just the console. What about formatting into a string? What about some other target? One of the challenges of working with `printf` is that while it supports output to various targets, it does so through distinct functions that aren't overloads and are difficult to use generically. To be fair, the C programming language doesn't support overloads or generic programming. Still, let's not repeat history. I'd like to be able to print to the console just as easily and generically as I can print to a string or a file. What I have in mind is illustrated in **Figure 1**.

From a design or implementation perspective, there should be one `Write` function template that acts as a driver and any number of targets or target adapters that are bound generically based on the target identified by the caller. A developer should then be able to easily add additional targets as needed. So how would this work? One option is to make the target a template parameter. Something like this:

```
template <typename Target, typename ... Args>
void Write(Target & target,
           char const * const format, Args const & ... args)
{
    target(format, args ...);
}

int main()
{
    Write(printf, "Hello %d\n", 2015);
}
```

Figure 1 Generic Output with Type Deduction

```
Write(printf, "Hello %\n", 2015);

FILE * file = // Open file
Write(file, "Hello %\n", 2015);

std::string text;
Write(text, "Hello %\n", 2015);
```


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, EDI/X12, NAESB, OFTP ...
- **Credit Card Processing**
Authorize.Net, TSYS, FDMS ...
- **Shipping & Tracking**
FedEx, UPS, USPS ...
- **Accounting & Banking**
QuickBooks, OFX ...
- **Internet Business**
Amazon, eBay, PayPal ...
- **Internet Protocols**
FTP, SMTP, IMAP, POP, WebDav ...
- **Secure Connectivity**
SSH, SFTP, SSL, Certificates ...
- **Secure Email**
S/MIME, OpenPGP ...
- **Network Management**
SNMP, MIB, LDAP, Monitoring ...
- **Compression & Encryption**
Zip, Gzip, Jar, AES ...



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. Since 1994, 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

This works to a point. I can write other targets that conform to the signature expected of `printf` and it ought to work well enough. I could write a function object that conforms and appends output to a string:

```
struct StringAdapter
{
    std::string & Target;

    template <typename ... Args>
    void operator()(char const * const format, Args const & ... args)
    {
        // Append text
    }
};
```

I can then use it with the same `Write` function template:

```
std::string target;
Write(StringAdapter{ target }, "Hello %d", 2015);
assert(target == "Hello 2015");
```

At first this might seem quite elegant and even flexible. I can write all kinds of adapter functions or function objects. But in practice it quickly becomes rather tedious. It would be much more desirable to simply pass the string as the target directly and have the `Write` function template take care of adapting it based on its type. So, let's have the `Write` function template match the requested target with a pair of overloaded functions for appending or formatting output. A little compile-time indirection goes a long way. Realizing that much of the output that may need to be written will simply be text without any placeholders, I'll add not one, but a pair of overloads. The first would simply append text. Here's an `Append` function for strings:

```
void Append(std::string & target,
            char const * const value, size_t const size)
{
    target.append(value, size);
}
```

I can then provide an overload of `Append` for `printf`:

```
template <typename P>
void Append(P target, char const * const value, size_t const size)
{
    target("%.*s", size, value);
}
```

I could have avoided the template by using a function pointer matching the `printf` signature, but this is a bit more flexible because other functions could conceivably be bound to this same implementation and the compiler isn't hindered in any way by the pointer indirection. I can also provide an overload for file or stream output:

```
void Append(FILE * target,
            char const * const value, size_t const size)
{
    fprintf(target, "%.*s", size, value);
}
```

Of course, formatted output is still essential. Here's an `AppendFormat` function for strings:

```
template <typename ... Args>
void AppendFormat(std::string & target,
                  char const * const format, Args ... args)
{
    int const back = target.size();
    int const size = snprintf(nullptr, 0, format, args ...);
    target.resize(back + size);
    snprintf(&target[back], size + 1, format, args ...);
}
```

It first determines how much additional space is required before resizing the target and formatting the text directly into the string. It's tempting to try and avoid calling `snprintf` twice by checking whether there's enough room in the buffer. The reason I tend to

always call `snprintf` twice is because informal testing has indicated that calling it twice is usually cheaper than resizing to capacity. Even though an allocation isn't required, those extra characters are zeroed, which tends to be more expensive. This is, however, very subjective, dependent on data patterns and how frequently the target string is reused. And here's one for `printf`:

```
template <typename P, typename ... Args>
void AppendFormat(P target, char const * const format, Args ... args)
{
    target(format, args ...);
}
```

An overload for file output is just as straightforward:

```
template <typename ... Args>
void AppendFormat(FILE * target,
                  char const * const format, Args ... args)
{
    fprintf(target, format, args ...);
}
```

I now have one of the building blocks in place for the `Write` driver function. The other necessary building block is a generic way of handling argument formatting. While the technique I illustrated in my March 2015 column was simple and elegant, it lacked the ability to deal with any values that didn't map directly to types supported by `printf`. It also couldn't deal with argument expansion or complex argument values such as user-defined types. Once again, a set of overloaded functions can solve the problem quite elegantly. Let's assume the `Write` driver function will pass each argument to a `WriteArgument` function. Here's one for strings:

```
template <typename Target>
void WriteArgument(Target & target, std::string const & value)
{
    Append(target, value.c_str(), value.size());
}
```

Realizing much of the output that may need to be written will simply be text without any placeholders, I'll add not one, but a pair of overloads.

The various `WriteArgument` functions will always accept two arguments. The first represents the generic target while the second is the specific argument to write. Here, I'm relying on the existence of an `Append` function to match the target and take care of appending the value to the end of the target. The `WriteArgument` function doesn't need to know what that target actually is. I could conceivably avoid the target adapter functions, but that would result in a quadratic growth in `WriteArgument` overloads. Here's another `WriteArgument` function for integer arguments:

```
template <typename Target>
void WriteArgument(Target & target, int const value)
{
    AppendFormat(target, "%d", value);
}
```

In this case, the `WriteArgument` function expects an `AppendFormat` function to match the target. As with the `Append` and

The screenshot shows a web browser window with the URL `DevExpress.com/Demos-14.1/RWA.DevAV/Customers.aspx`. The application header includes 'DEVAV CUSTOMERS' and navigation buttons like 'New', 'Delete', and 'Revenue Analysis'. A table lists customer records with columns: Name, Address, City, State, Zipcode, and an information icon. The 'K&S Music' row is highlighted. Below the table, a 'CUSTOM FILTERS' sidebar is visible, and a 'K&S Music Contacts' section displays five contact cards for Rose Bert, Tammy Pickard, Sally Port, Kay Tribble, and Joss Ahrens, each with a photo, name, title, and address.

STATE	Name	Address	City	State	Zipcode
All	Super Mart of the West	P O Box 5342427	Bentonville	AR	72716
Illinois	Electronics Depot	P O Box 6593265	Atlanta	GA	30339
California	K&S Music	P O Box 8596565	Minneapolis	MN	55403
Arizona	Tom's Club	P O Box 3433125	Issaquah	WA	98124
Georgia	E-Mart	P O Box 2424569	Hoffman Estates	IL	60179

Page 1 of 2 (20 items) < 1 2 >

K&S Music Contacts

- Rose Bert**, Manager, 1401 2nd Ave, Seattle, WA 98101
- Tammy Pickard**, Buyer, 1401 2nd Ave, Seattle, WA 98101
- Sally Port**, Clerk, 633 N Milwaukee St, Boise, ID 83704
- Kay Tribble**, Manager, 5255 E Broadway Blvd, Tucson, AZ 85711
- Joss Ahrens**, Assistant Manager, 633 N Milwaukee St, Boise, ID 83704

Your Next Great Web App Starts Here

With DevExpress web controls, you can build a bridge to the future on the platform you know and love. The 95+ AJAX Web Forms Controls & 50+ MVC Extensions that ship inside the DevExpress ASP.NET Subscription allow you to create functional, elegant and interactive experiences for the web, regardless of the target browser or computing device. All major browsers including Internet Explorer, FireFox, Chrome, Safari and Opera, are fully supported and continuously tested to ensure the highest compatibility.

Get started today.

Download your free 30-day trial and experience the DevExpress difference yourself.

devexpress.com/asp



AppendFormat overloads, writing additional WriteArgument functions is straightforward. The beauty of this approach is that the argument adapters don't need to return some value up the stack to the printf function, as they did in the March 2015 version. Instead, WriteArgument overloads actually scope the output such that the target is written immediately. This means complex types can be used as arguments and temporary storage can even be relied upon to format their text representation. Here's a WriteArgument overload for GUIDs:

```
template <typename Target>
void WriteArgument(Target & target, GUID const & value)
{
    wchar_t buffer[39];
    StringFromGUID2(value, buffer, _countof(buffer));
    AppendFormat(target, "%.*ls", 36, buffer + 1);
}
```

I could even replace the Windows StringFromGUID2 function and format it directly, perhaps to improve performance or add portability, but this clearly shows the power and flexibility of this approach. User-defined types can easily be supported with the addition of a WriteArgument overload. I've called them overloads here, but strictly speaking they need not be. The output library can certainly provide a set of overloads for common targets and arguments, but the Write driver function shouldn't assume the adapter functions are overloads and, instead, should treat them like the nonmember begin and end functions defined by the Standard C++ library. The nonmember begin and end functions are extensible and adaptable to all kinds of standard and non-standard containers precisely because they don't need to reside in the std namespace, but should instead be local to the namespace of the type being matched. In the same way, these target and argument adapter functions should be able to reside in other namespaces to support the developer's targets and user-defined arguments. So what does the Write driver function look like? For starters, there's only one Write function:

```
template <typename Target, unsigned Count, typename ... Args>
void Write(Target & target,
           char const (&format)[Count], Args const & ... args)
{
    assert(Internal::CountPlaceholders(format) == sizeof ... (args));
    Internal::Write(target, format, Count - 1, args ...);
}
```

The first thing it needs to do is determine whether the number of placeholders in the format string equals the number of arguments in the variadic parameter pack. Here, I'm using a runtime assert, but this really ought to be a static_assert that checks the format string at compile time. Unfortunately, Visual C++ is not quite there yet. Still, I can write the code so that when the compiler catches up, the code can be easily updated to check the format

Figure 2 Visualizing a Vector

```
template <typename Target, typename Value>
void WriteArgument(Target & target, std::vector<Value> const & values)
{
    for (size_t i = 0; i != values.size(); ++i)
    {
        if (i != 0)
        {
            WriteArgument(target, ", ");
        }

        WriteArgument(target, values[i]);
    }
}
```

string at compile time. As such, the internal CountPlaceholders function should be a constexpr:

```
constexpr unsigned CountPlaceholders(char const * const format)
{
    return (*format == '%') +
        (*format == '\\0' ? 0 : CountPlaceholders(format + 1));
}
```

When Visual C++ achieves full conformance with C++14, at least with regard to constexpr, you should be able to simply replace the assert inside the Write function with static_assert. Then it's on to the internal overloaded Write function to roll out the argument-specific output at compile time. Here, I can rely on the compiler to generate and call the necessary overloads of the internal Write function to satisfy the expanded variadic parameter pack:

```
template <typename Target, typename First, typename ... Rest>
void Write(Target & target, char const * const value,
           size_t const size, First const & first, Rest const & ... rest)
{
    // Magic goes here
}
```

I'll focus on that magic in a moment. Ultimately, the compiler will run out of arguments and a non-variadic overload will be required to complete the operation:

```
template <typename Target>
void Write(Target & target, char const * const value, size_t const size)
{
    Append(target, value, size);
}
```

Both internal Write functions accept a value, along with the value's size. The variadic Write function template must further assume there is at least one placeholder in the value. The non-variadic Write function need make no such assumption and can simply use the generic Append function to write any trailing portion of the format string. Before the variadic Write function can write its arguments, it must first write any leading characters and, of course, find the first placeholder or metacharacter:

```
size_t placeholder = 0;

while (value[placeholder] != '%')
{
    ++placeholder;
}
```

Only then can it write the leading characters:

```
assert(value[placeholder] == '%');
Append(target, value, placeholder);
```

The first argument can then be written and the process repeats until no further arguments and placeholders are left:

```
WriteArgument(target, first);
Write(target, value + placeholder + 1, size - placeholder - 1, rest ...);
```

I can now support the generic output in **Figure 1**. I can even convert a GUID to a string quite simply:

```
std::string text;
Write(text, "{%}", __uuidof(IUnknown));
assert(text == "{00000000-0000-0000-C000-000000000046}");
```

What about something a little more interesting? How about visualizing a vector:

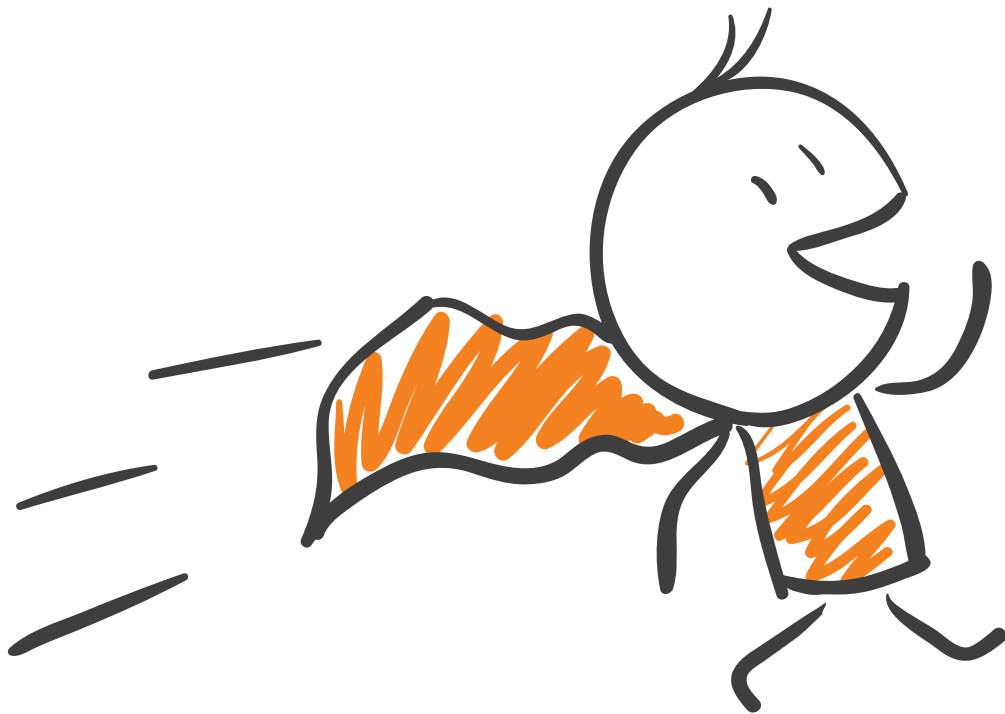
```
std::vector<int> const numbers{ 1, 2, 3, 4, 5, 6 };
std::string text;
Write(text, "{ % }", numbers);
assert(text == "{ 1, 2, 3, 4, 5, 6 }");
```

For that I simply need to write a WriteArgument function template that accepts a vector as an argument, as shown in **Figure 2**.

Notice how I didn't force the type of the elements in the vector. That means I can now use the same implementation to visualize a vector of strings:

Unleash the **UI Superhero** in You

With DevExpress tools, you'll deliver amazing user-experiences for Windows®, the Web and your Mobile world.



UI CONTROLS FOR

DESKTOP / WEB / MOBILE

Unleash the UI Superhero in you and deliver elegant solutions that fully address customer needs today & leverage your existing knowledge to build next generation touch-enabled solutions for tomorrow. Whether it's 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.



Office-Inspired Apps



Touch-Enabled Windows
& Web Apps



HTML5 Mobile Apps



Reporting-Dashboard
& Analytics Apps

When Only the Best Will Do

“ Given the completeness and maturity of the many components in this year's packages, the Libraries category was tough going for the judges. However, one vendor's component suite made a stunning impact with its leaps in forward-thinking and polished presentation.

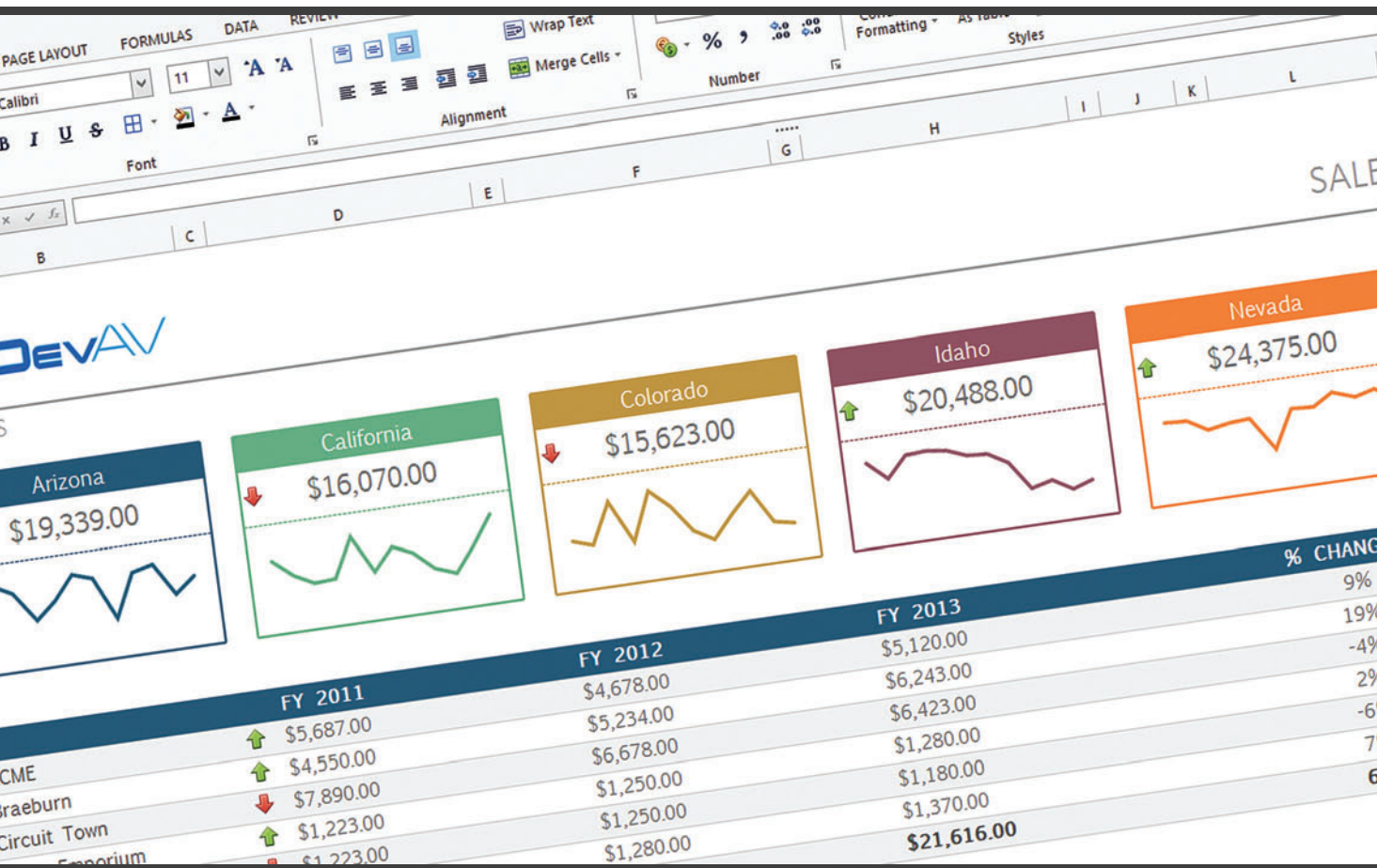
DevExpress has been in the running for the top slot in the Jolt Libraries award for some time, but this year they have captured lightning in a bottle and were deemed the winner by a large margin.”

Mike Riley
for Dr. Dobb's Magazine



Office[®] Inspired Apps

Get started today and create high-performance, high-impact .NET solutions that fully replicate the look, feel and user-experience of **Microsoft Office[®]**.



Your Next Great Business App Starts Here

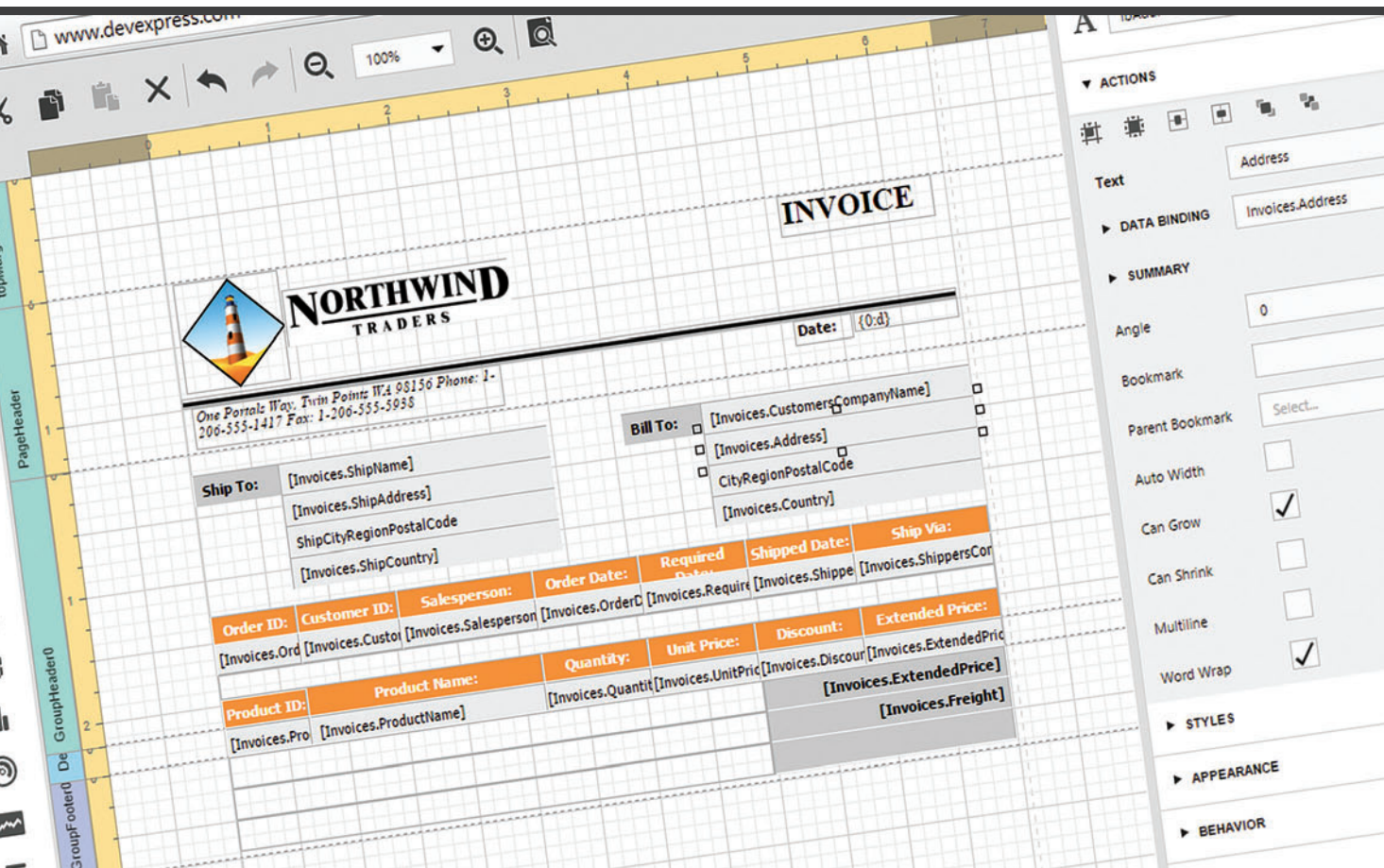
Explore our complete range of Office-inspired controls for all major .NET platforms.

devexpress.com/office



Reporting Made Easy

Inform and analyze with absolute ease. DevExpress Universal includes an enterprise-ready reporting platform with integrated Windows & Web report designers.



Your Next Great Business Report Starts Here

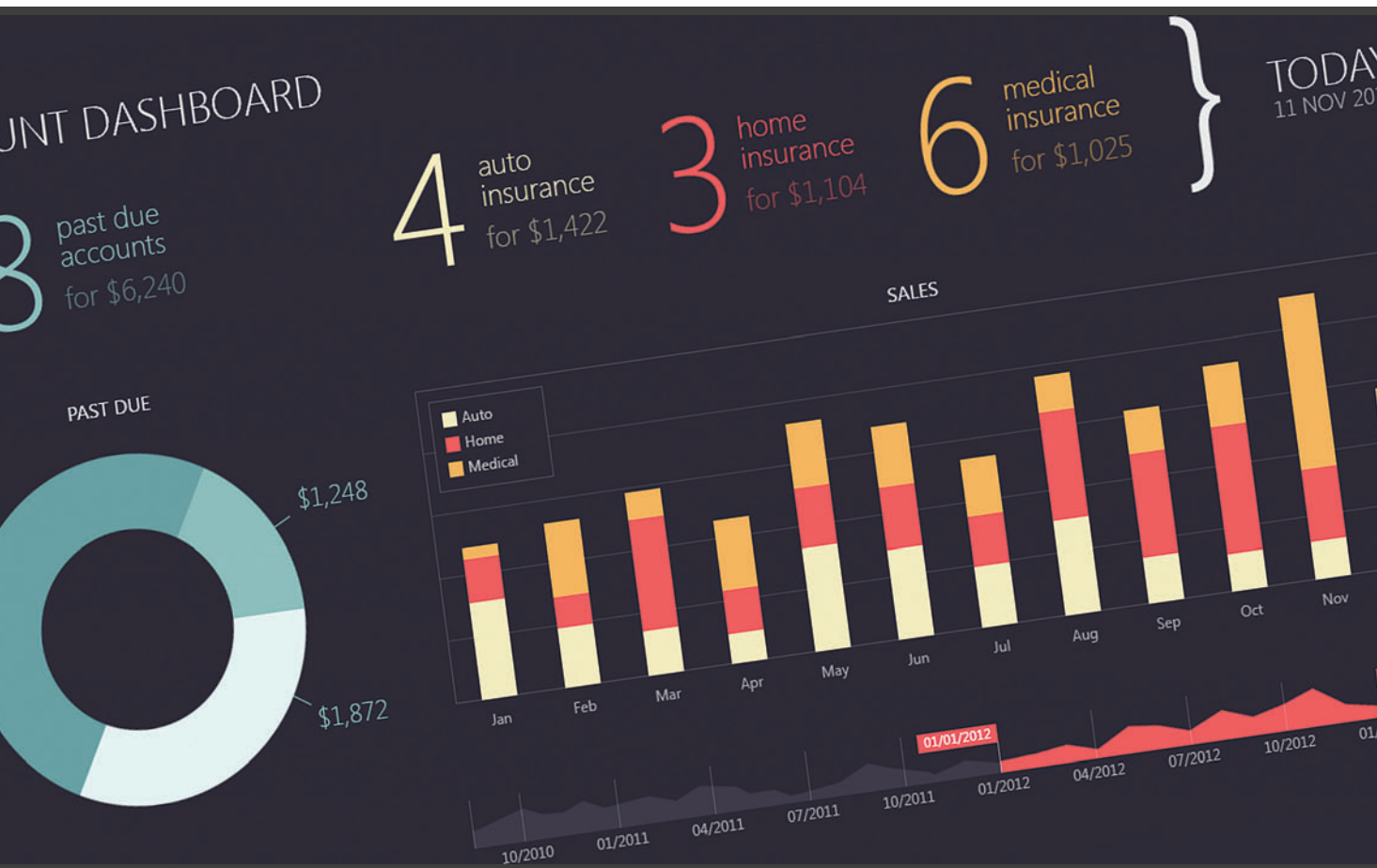
See the power and flexibility of our reporting platform in action.

devexpress.com/reports



Dashboards & Analytics

DevExpress Universal ships with everything you'll need to create information-rich decision support systems and to distribute your dashboard solutions royalty-free.



Your Next Great Dashboard Starts Here

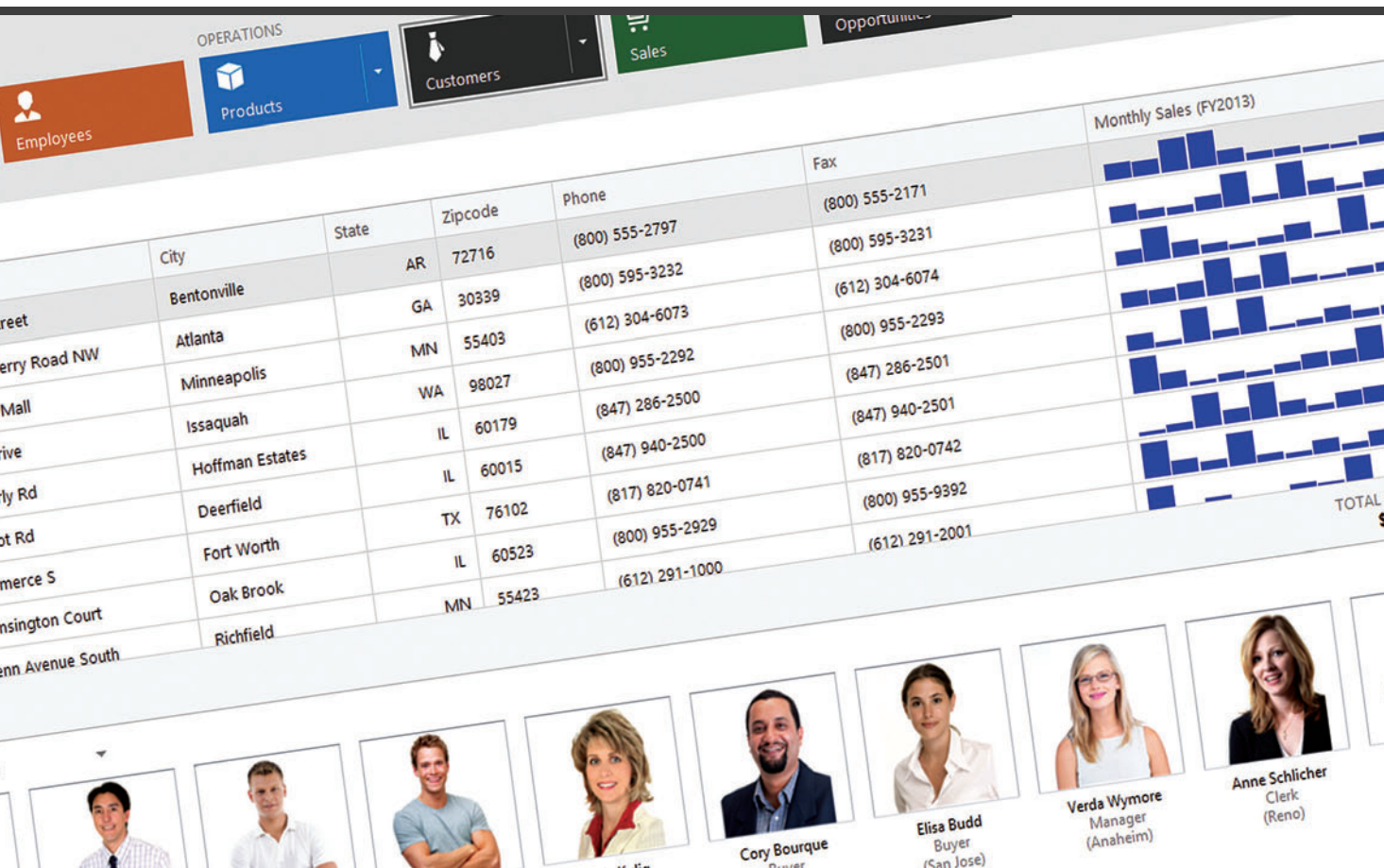
Learn how you can create fully customizable Dashboards with our flexible data visualization tools.

devexpress.com/dashboard



Touch-Optimized Hybrid Apps

DevExpress Universal provides a comprehensive collection of UX tools so you can create touch-first apps that are built for today and ready for tomorrow.



Your Next Great Hybrid App Starts Here

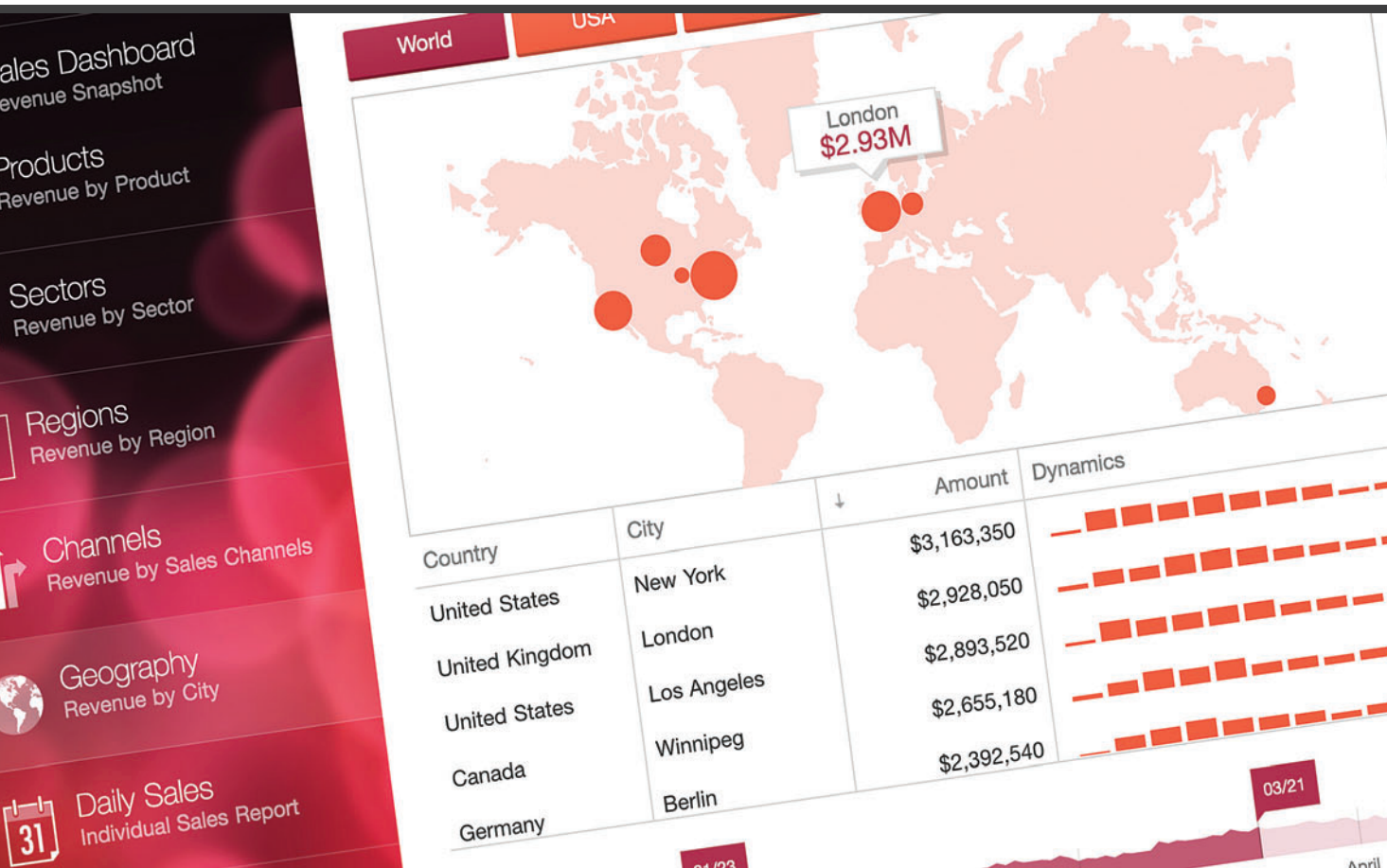
Build touch-first apps for any Windows device and leverage existing code investments.

devexpress.com/hybrid



Mobile Apps for Your BYOD World

Consistent user experiences, everywhere. DevExpress Universal allows you to create apps with a native UX for iOS, Android and Windows Phone, without extra effort, code or UI customization.



Your Next Great Mobile App Starts Here

Put the power of HTML5 and JS into action today.

devexpress.com/mobile



When Only the Best Will Do

Experience the DevExpress difference for yourself and see why your peers have consistently voted DevExpress Universal best-in-class.



Download your free 30-day trial
from devexpress.com/try



All trademarks or registered trademarks are property of their respective owners.

```
std::vector<std::string> const names{ "Jim", "Jane", "June" };
std::string text;
Write(text, "{ % }", names);
assert(text == "{ Jim, Jane, June }");
```

Of course, that begs the question: What if I want to further expand a placeholder? Sure, I can write a WriteArgument for a container, but it provides no flexibility in tweaking the output. Imagine I need to define a palette for an app's color scheme and I have primary colors and secondary colors:

```
std::vector<std::string> const primary = { "Red", "Green", "Blue" };
std::vector<std::string> const secondary = { "Cyan", "Yellow" };
```

The Write function will gladly format this for me:

```
Write(printf,
      "<Colors>%%/Colors>",
      primary,
      secondary);
```

The output, however, is not quite what I want:

```
<Colors>Red, Green, BlueCyan, Yellow</Colors>
```

That's obviously wrong. Instead, I'd like to mark up the colors such that I know which are primary and which are secondary. Perhaps something like this:

```
<Colors>
  <Primary>Red</Primary>
  <Primary>Green</Primary>
  <Primary>Blue</Primary>
  <Secondary>Cyan</Secondary>
  <Secondary>Yellow</Secondary>
</Colors>
```

Let's add one more WriteArgument function that can provide this level of extensibility:

```
template <typename Target, typename Arg>
void WriteArgument(Target & target, Arg const & value)
{
    value(target);
}
```

Notice the operation seems to be flipped on its head. Rather than passing the value to the target, the target is being passed to the value. In this way, I can provide a bound function as an argument instead of just a value. I can attach some user-defined behavior and not merely a user-defined value. Here's a WriteColors function that does what I want:

```
void WriteColors(int (*target)(char const *, ...),
                std::vector<std::string> const & colors, std::string const & tag)
{
    for (std::string const & color : colors)
    {
        Write(target, "<%>%%/>", tag, color, tag);
    }
}
```

Notice this isn't a function template and I've had to essentially hardcode it for a single target. This is a target-specific customization, but shows what's possible even when you need to step out of the generic type deduction directly provided by the Write driver function. But how can it be incorporated into a larger write operation? Well, you might be tempted for a moment to write this:

```
Write(printf,
      "<Colors>\n%%/Colors>\n",
      WriteColors(printf, primary, "Primary"),
      WriteColors(printf, secondary, "Secondary"));
```

Putting aside for a moment the fact that this won't compile, it really wouldn't give you the right sequence of events, anyway. If this were to work, the colors would be printed before the opening <Colors> tag. Clearly, they should be called as if they were arguments in the order that they appear. And that's what the new WriteArgument function template allows. I just need to bind

the WriteColors invocations such that they can be called at a later stage. To make that even simpler for someone using the Write driver function, I can offer up a handy bind wrapper:

```
template <typename F, typename ... Args>
auto Bind(F call, Args && ... args)
{
    return std::bind(call, std::placeholders::_1,
                    std::forward<Args>(args) ...);
}
```

This Bind function template simply ensures that a placeholder is reserved for the eventual target to which it will be written. I can then rightly format my color palette as follows:

```
Write(printf,
      "<Colors>%%/Colors>\n",
      Bind(WriteColors, std::ref(primary), "Primary"),
      Bind(WriteColors, std::ref(secondary), "Secondary"));
```

And I get the expected tagged output. The ref helper functions aren't strictly necessary, but avoid making a copy of the containers for the call wrappers.

Not convinced? The possibilities are endless. You can handle character string arguments efficiently for both wide and normal characters:

```
template <typename Target, unsigned Count>
void WriteArgument(Target & target, char const (&value)[Count])
{
    Append(target, value, Count - 1);
}

template <typename Target, unsigned Count>
void WriteArgument(Target & target, wchar_t const (&value)[Count])
{
    AppendFormat(target, "%.*s", Count - 1, value);
}
```

In this way I can easily and safely write output using different character sets:

```
Write(printf, "% %", "Hello", L"World");
```

What if you don't specifically or initially need to write output, but instead just need to calculate how much space would be required? No problem, I can simply create a new target that sums it up:

```
void Append(size_t & target, char const *, size_t const size)
{
    target += size;
}

template <typename ... Args>
void AppendFormat(size_t & target,
                 char const * const format, Args ... args)
{
    target += snprintf(nullptr, 0, format, args ...);
}
```

I can now calculate the required size quite simply:

```
size_t count = 0;
Write(count, "Hello %", 2015);
assert(count == strlen("Hello 2015"));
```

I think it's safe to say that this solution finally addresses the type frailty inherent in using printf directly while preserving most of the performance benefits. Modern C++ is more than able to meet the needs of developers looking for a productive environment with reliable type checking while maintaining the performance for which C and C++ is traditionally known. ■

KENNY KERR is a computer programmer based in Canada, as well as an author for *Pluralsight* and a Microsoft MVP. He blogs at kennykerr.ca and you can follow him on Twitter at twitter.com/kennykerr.

THANKS to the following Microsoft technical expert for reviewing this article:
James McNellis



The EF6, EF7 and ASP.NET 5 Soup

My January 2015 Data Points column, “Looking Ahead to Entity Framework 7,” highlighted what was coming in EF7 by looking at the state of the EF7 alpha at the time I wrote the article. In December 2014, just as that column’s issue was going to press, the EF team made an important decision about staging the EF7 releases. I managed to squeeze two sentences into the article at the last minute: “According to the post at bit.ly/1ykagF0, ‘EF7—Priorities, Focus and Initial Release,’ the first release of EF7 will focus on compatibility with ASP.NET 5. Subsequent releases will add more features.”

Since then, I found that sorting out what, exactly, this means to be a little confusing. I want to be sure you understand well enough what this initial release of EF7 targets, whether you should be using it and what your options are.

It will help to discuss the difference between the next version of the Microsoft .NET Framework and what ASP.NET 5 apps will run on. I’ll provide a high-level look, but check out Daniel Roth’s article, “Deep Dive into the ASP.NET 5 Runtime,” from the March 2015 issue to gain a more in-depth understanding. Then I’ll explain how EF6 and EF7 fit into the mix.

ASP.NET 5 (aka ASP.NET vNext) has been designed to have less reliance on the full .NET Framework and even Windows itself. Under the ASP.NET 5 umbrella, you can create Web applications and Web APIs using the new combined feature set in ASP.NET MVC 6, class libraries and even a console app. If you think of ASP.NET as a way to create Web sites, the console app is a mystery. But ASP.NET 5 is really providing a new “version” of the .NET Framework on which apps can sit. In fact, this new version currently has two flavors—one that runs on the full Common Language Runtime (CLR) and another that runs on the new CoreCLR. A third, which will run on a cross-platform CLR, will be added in the near future. The most streamlined version is called the .NET Core, which runs in a streamlined runtime, CoreCLR. If you think back to your early lessons on the .NET Framework, you might recall that Microsoft defined the Common Language Infrastructure (CLI), which is a specification. Then Microsoft built the CLR based on the CLI. Mono is another implementation of the CLI that runs on Linux and Mac OS X. .NET apps have always depended on the availability of a CLI implementation. Most of us have used Visual Studio and Windows to create .NET apps

that depend on the CLR. Xamarin has become increasingly popular for developing apps that can run on Mono.

The CoreCLR and Entity Framework

Now, Microsoft has created a streamlined implementation of the CLI called the CoreCLR that’s not only open source, but also can be distributed by NuGet and other package managers. This gives developers the ability to create lightweight apps that are great for mobile devices and cloud-based server applications. And it also removes the constraint to deploy apps only to Windows machines.

Entity Framework 6 (and earlier) relies on the full .NET Framework and the full CLR, and won’t run with apps targeting the CoreCLR. But EF7 has been designed as a set of smaller, composable APIs that can be mixed and matched based on the feature set you need. For example, if you’re targeting a non-relational data store, you don’t need the relational or migrations features that are designed for relational databases. With EF7, that logic is in a separate DLL that you won’t need to retrieve and load into memory.

When you choose to target the CoreCLR, you do so by selecting the correct runtime environment. To go with ASP.NET 5, Microsoft created the K Runtime Environment (KRE) that, as described by ASP.NET MVP Gunnar Peipman, “... is the code required to bootstrap and run an ASP.NET vNext application. The K Runtime will be renamed at some point before it’s released. Keep an eye out for this change. For example, the KVM (version manager) will become the DNVM (.NET Version Manager). The runtime includes things like the compilation system, SDK tools and the native CLR hosts” (bit.ly/1x9rp0n). There are 32-bit and 64-bit versions of the KRE that support the CoreCLR. You can see these as ASP.NET 5 application target options in **Figure 1**.

The easiest way to see EF7 in an application, outside of watching the demo video in my Pluralsight course, “Looking Ahead to

This article discusses unreleased technology. All information is subject to change.

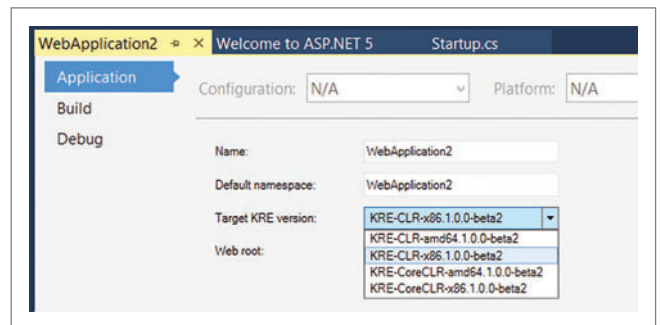


Figure 1 Selecting the Target KRE in an ASP.NET 5 Application

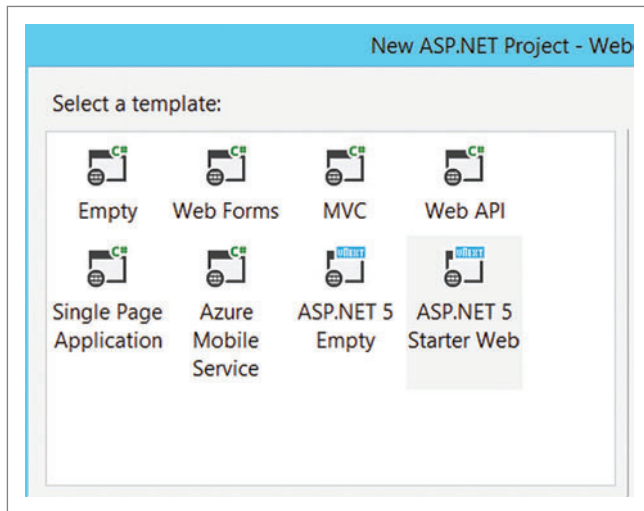


Figure 2 Creating a New ASP.NET 5 Application with Entity Framework 7 Preconfigured

Entity Framework 7” (bit.ly/18ct13F), is to use the ASP.NET 5 project template, which will give you a solution with EF7 already wired up to provide identity-based authentication to your application.

Start by choosing an ASP.NET Web Application and then the ASP.NET 5 Starter Web template, as shown in **Figure 2**.

You’ll find that the EF7 EntityFramework.SqlServer and EntityFramework.Commands packages are already selected. The SqlServer package will cause its dependencies (EntityFramework.Core, EntityFramework.Relational) to be pulled in.

You can see this in the project.json file, which lists the packages for NuGet to retrieve in its dependencies section. Here are a few lines from that section:

```
"dependencies": {
  "EntityFramework.SqlServer": "7.0.0-beta2",
  "EntityFramework.Commands": "7.0.0-beta2",
  "Microsoft.AspNet.Mvc": "6.0.0-beta2",
```

You can also see what ends up in the references section, which by default lists the references for the ASP.NET 5.0 and ASP.NET Core 5.0 versions of the application. **Figure 3** shows the listing with the ASP.NET

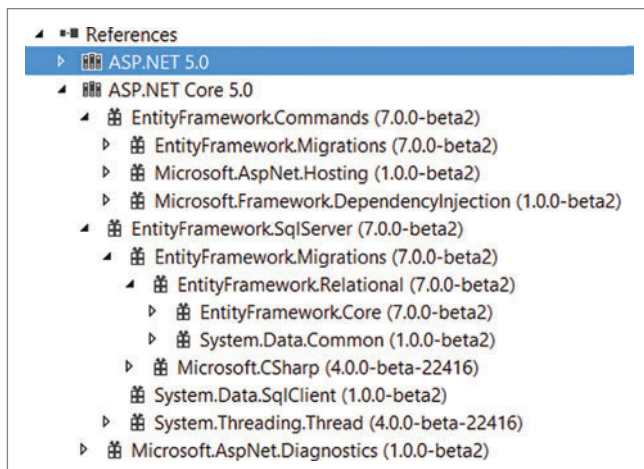


Figure 3 Both ASP.NET 5.0 and ASP.NET Core 5.0 References Contain Entity Framework 7 Packages and APIs

Core 5.0 references expanded. Note that because the template currently targets the beta2 version, you’ll see the Migrations package referenced, although in a later version, Migrations became part of the Relational API.

ASP.NET 5 and EF7 both embrace and rely heavily on dependency injection (DI). You can see in the project’s Startup.cs in **Figure 4** that there is logic to let the app know it should use Entity Framework, its SQL Server API and the single predefined DbContext used for security, ApplicationDbContext. You can see this same context being wired up as the Identity service for the application. The connection string is stored in the config.json file and the Startup constructor wires up that file so the application will be able to find it when needed.

The project template now sets you up by default to use EF7 and to use its logic whether you’re targeting the streamlined CoreCLR or the full .NET Framework. In fact, by default, the project isn’t defined to run under the CoreCLR, but to use the full .NET Framework. So let’s look at where this option fits.

The Full CLR for ASP.NET 5 (and Entity Framework)

There’s an update to the .NET Framework coming. .NET Framework 4.6, and it will allow you to continue to create all of the styles of software you’ve been able to already—from Windows Forms and Web Forms to Windows Presentation Foundation (WPF) and ASP.NET MVC 5 apps. Thanks to the other KRE versions shown in **Figure 1**, KRE-CLR-amd64 and KRE-CLR-x86, it’s also possible to run an ASP.NET 5 application on top of the full .NET Framework. This lets you have your cake and eat it, too, if you want to benefit from other new features of ASP.NET 5, such as MVC 6 or the new Web API, which is a merger of MVC Controllers and Web API. It also gives you backward compatibility because you’re targeting the full .NET Framework and CLR and can access its full set of features. By default, the project.json file indicates you want to be able to run the application under either version of the CLR:

```
"frameworks": {
  "aspnet50": { },
  "aspnetcore50": { }
},
```

You could remove the “aspnetcore50” line completely (along with the comma at the end of the previous line) and the effect would be twofold. First, the ASP.NET Core 5.0 section would disappear from References in Solution Explorer. Second, the KRE targets displayed in **Figure 1** would be reduced to only the KRE-CLR options.

You’d still be targeting EF7 and getting all of the benefits of this new version. But, what hasn’t been obvious is that because KRE-CLR targets the full .NET Framework, it’s compatible with Entity Framework 6. This means if you have existing EF6 logic, it’s possible to use that in combination with ASP.NET 5 (though not the CoreCLR version), and gain the benefits of the new features of ASP.NET 5 without having to rework your EF6 code to align with EF7.

ASP.NET 5 (Full CLR) with EF6

Knowing this was possible, naturally I had to try to get EF6 into an ASP.NET 5 solution. Keep in mind that the approach I took to achieve this may very well need to change between the very early version I’m using right now (in February 2015) and when ASP.NET 5 and Visual Studio 2015 are released.



Open, Create, Convert, Print & Save Files

from within your *own* applications.

➤ ASPOSE.TOTAL

allows you to process these file formats:

- Word documents
- Excel spreadsheets
- PowerPoint presentations
- PDF documents
- Project documents
- Visio documents
- Outlook emails
- OneNote documents



DOC XLS PPT PDF EML
PNG XML RTF HTML VSD
BMP & barcode images.

SCAN FOR
20% SAVINGS!



 **ASPOSE**
Your File Format APIs

Contact Us:

US: +1 888 277 6734
EU: +44 141 416 1112
AU: +61 2 8003 5926
sales@aspose.com

Helped over 11,000 companies and over 250,000 users work with documents in their applications.



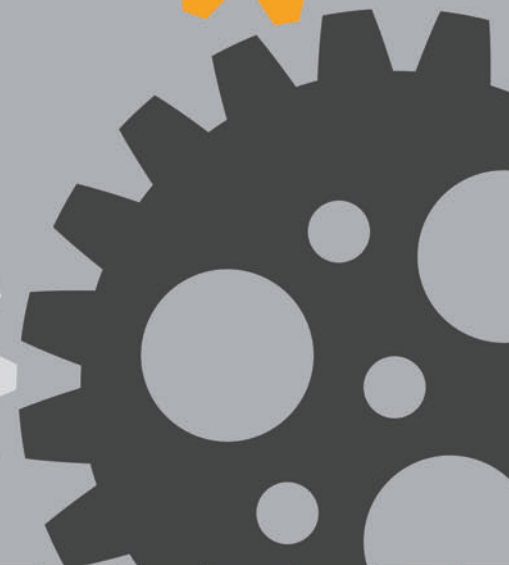
.NET, Java & Cloud

Your File Format APIs



GET STARTED NOW

- Free Trial
- 30 Day Temp License
- Free Support
- Community Forums
- Live Chat
- Blogs
- Examples
- Video Demos



I found that the most important factor in achieving this was keeping the Entity Framework logic and dependencies in a separate project from the ASP.NET 5 project. I typically design my applications this way anyway, rather than having all layers of my app in a single project. With a separate project devoted to EF, you don't have to worry about the ASP.NET 5 app pulling in EF7 APIs on the fly.

Rather than starting with the ASP.NET 5 Starter Web Project, you're better off using the ASP.NET Empty Project template. The project.json file has only one dependency specified, Microsoft.AspNet.Server.IIS.

Next, you create a second project. I chose a .NET 4.6 Class Library project. I could then use the Package Manager Console to install EF6 by ensuring I was pointing to nuget.org as the package source and to the class library as the default project. At this point I was able to call install-package entityframework as I've always done. This brought entityframework.dll and entityframework.sqlserver.dll into the project.

To test, I created a simple class, Ninja.cs (because I think of EF6 as the Ninja edition, while EF7 is the Samurai edition):

```
public class Ninja {
    public int Id { get; set; }
    public string Name { get; set; }
}
```

and a DbContext that exposed a DbSet of Ninjas:

```
public class NinjaContext : DbContext {
    public NinjaContext()
        :base(@"Data Source=(localdb)\mssqllocaldb;
            Initial Catalog=NinjaContext;
            Integrated Security=True;"){ }

    public DbSet<Ninja> Ninjas { get; set; }
}
```

For simplicity in testing, I've hardcoded the SQL Server connection string directly into my DbContext class.

Figure 4 Partial Listing of Default Startup.cs

```
public class Startup {
    public Startup(IHostingEnvironment env) {
        Configuration = new Configuration()
            .AddJsonFile("config.json")
            .AddEnvironmentVariables();
    }

    public IConfiguration Configuration { get; set; }

    public void ConfigureServices(IServiceCollection services) {
        services.AddEntityFramework(Configuration)
            .AddSqlServer()
            .AddDbContext<ApplicationDbContext>();

        services.AddIdentity<ApplicationUser, IdentityRole>(Configuration)
            .AddEntityFrameworkStores<ApplicationDbContext>();
        services.AddMvc();
    }
}
```

Figure 5 Startup Class to Configure the ASP.NET 5 App That Uses Entity Framework 6

```
public class Startup {
    public void Configure(IApplicationBuilder app) {
        app.UseMvc(routes =>
        {
            routes.MapRoute(
                name: "default",
                template: "{controller}/{action}/{id?}",
                defaults: new { controller = "Ninja", action = "Index" });
        });
    }

    public void ConfigureServices(IServiceCollection services) {
        services.AddMvc();
    }
}
```

With my model defined, I can enable migrations, add a new migration and create the database as I've always done. Note that when I installed EF6 into the data project, I left the app.config file that the package created so I could set that project as the startup project and then the migration commands would work correctly. I also used the DbMigrationsConfiguration Seed method to pre-seed the database with a few Ninjas.

The CoreCLR is currently too bleeding edge for my tastes.

I don't want to make EF calls directly from the ASP.NET 5 application and create version confusion, so I have a class in my EF6 project that encapsulates the queries I'll need. I've called it Repository because it suits my purpose, but this isn't a class that follows the repository pattern. This isn't challenging with a demo app and a single query, but you can use your favorite pattern for separation of concerns for more complex apps:

```
public class Repository {
    public List<Ninja> GetAllNinjas() {
        using (var context=new NinjaContext())
        {
            return context.Ninjas.ToList();
        }
    }
}
```

With the data access project all set up, I can return to the ASP.NET 5 project. I've created a simple controller to retrieve and return a View:

```
public class NinjaController : Controller
{
    public IActionResult Index()
    {
        var repo = new Repository();

        return View(repo.GetAllNinjas());
    }
}
```

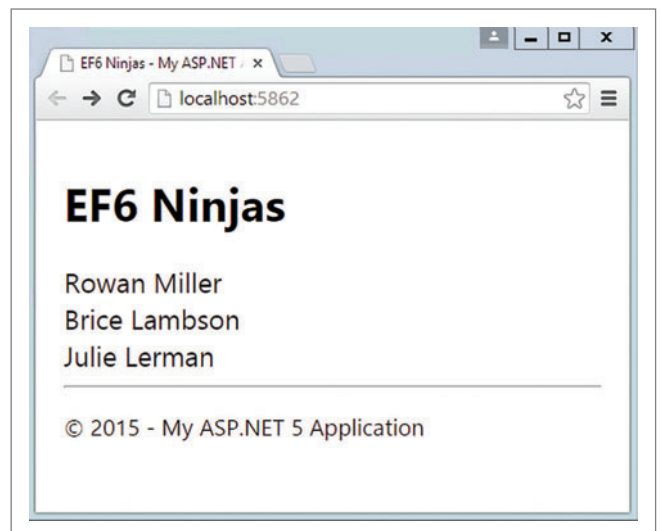


Figure 6 ASP.NET 5 Application Displaying Data via Entity Framework 6

I also created a simple Index.cshtml that will list the Ninjas passed in by the controller:

```
@model List<EF6Model.Ninja>

@{
    ViewBag.Title = "EF6 Ninjas";
}
@foreach (var item in Model)
{
    <div>
        @Html.Label(item.Name);
    </div>
}
```

Finally, **Figure 5** shows code added to the startup.cs file, to inject the MVC services and specify routing to the Ninja controller.

I also removed the aspnetcore50 framework from the project.json file, ensuring that I only target the full CLR.

With the database and some seed data created by migrations, I'm able to see the ASP.NET 5 app working with the EF6-based project, displaying data that was retrieved using EF6 (see **Figure 6**).

Will I Be Using EF7 and ASP.NET 5 Right Away?

I'm very interested in the new capabilities of EF7 and in learning how to best leverage them. I'm also looking forward to understanding this new avenue of programming that ASP.NET 5 opens. The CoreCLR will become richer as it evolves, as will EF7. I'm impressed with the state of EF7 as it will be released with ASP.NET 5. But because I'm not a Web developer, I don't have a reason to work in production code at the bleeding edge of ASP.NET, and the EF team has been clear that it recommends using the initial release of EF7, which will be marked as a prerelease, only with ASP.NET 5 applications. Otherwise, the team's guidance is to wait until EF7 is released as a true version. Therefore, I may well wait for the subsequent releases of EF7, but continue to follow and experiment with the project as it evolves to gain more parity with the features of EF6 that I might otherwise miss.

The CoreCLR is currently too bleeding edge for my tastes. If I find a scenario that fits what will be available in the full CLR release of ASP.NET 5, as well as the feature set of the EF7 prerelease, then I'll be eager to take that path. It's more likely, however, that I'll end up waiting for the subsequent releases. And by then ASP.NET 5 and the CoreCLR will be richer, too. For me, personally, the time until then will provide a chance to explore and learn about these tools in preparation for the later release.

In the meantime, I highly recommend reading Scott Guthrie's blog post, "Introducing ASP.NET 5," at bit.ly/1W4zzm to have a better understanding of all of the new features that ASP.NET 5 brings. The composable APIs are only one of many exciting improvements that developers will gain in this shift. ■

JULIE LERMAN is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/ blog and is the author of "Programming Entity Framework" (2010), as well as a Code First edition (2011) and a DbContext edition (2012), all from O'Reilly Media. Follow her on Twitter at twitter.com/julielerman and see her Pluralsight courses at juliel.me/PS-Videos.

THANKS to the following technical expert for reviewing this article:
Rick Strahl

msdnmagazine.com

NEW Entity Framework Profiler 3.0



Don't waste nights optimizing your database queries.

Don't let database dictate how your application works.



Find out how it can help you!

- OPTIMIZE QUERIES
- FIND BOTTLENECKS
- IMPROVE ENTITY FRAMEWORK PERFORMANCE
- SAVE COUNTLESS TIME

www.hibernatingrhinos.com

Proudly
Developed by



An Introduction to Building Windows Apps for Windows 10 Devices

Jerry Nixon and Andy Wigley

You have lived to see it: a single Windows OS that can run across every type of Windows device. It has a single device platform to enable true universal hardware drivers and a single application platform to enable true universal Windows apps. Years in the making, this is a significant engineering accomplishment.

At the OS level, this means a single, maintainable and agile code base. For developers, it provides a unified, reliable API surface across every Windows device, from Internet of Things (IoT) devices such as Raspberry Pi to phone, Xbox, tablet, Surface Hub, laptop, PC and more (like Microsoft HoloLens). As shown in **Figure 1**, it's a write-once-run-everywhere promise delivered in Windows 10 with the universal application platform (UAP).

This article discusses:

- The road to one Windows OS
- What you need to run on every device
- How to adapt your interface for different screens
- How to adapt your code for different devices
- What every Windows app needs to be successful

Technologies discussed:

Windows, Windows Phone, Xbox, XAML

Code download available at:

msdn.microsoft.com/magazine/msdnmag0515

The Journey to Windows 10

The convergence of Windows has been a long time coming. Back in 2011, Microsoft had three platforms with three OSes. The PC and server OS was Windows, built on the Windows NT code base. The phone OS was Windows Phone, a derivative of Windows CE with surface-level similarities to Windows NT, but a different code base. The Xbox 360 OS was Windows NT, but it was a 10-year-old fork so wildly divergent that it, too, was a distinct code base.

Windows 10 has become the small footprint, one core OS that runs on every device family.

At that time, Microsoft worked to bring a common Internet Explorer to each platform. There was no Windows Core, no Windows platform, and no UAP. The implementation of Internet Explorer on these three OSes was successful, but required considerable engineering gymnastics.

With Windows Phone 8, the Windows NT OS kernel replaced Windows CE on phones. This convergence moved things down the road toward a single code base. Windows, Windows Phone and Xbox 360 all leveraged the same kernel, though each still had

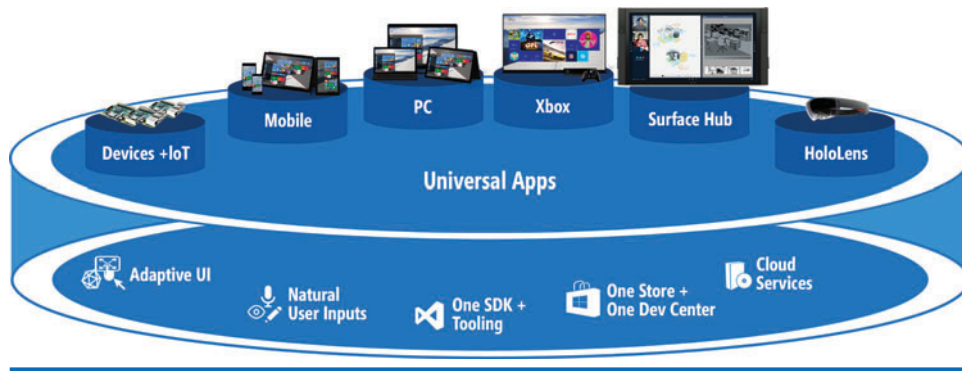


Figure 1 The Universal App Platform Enables Apps Across All Windows Device Families

unique code bases. In 2013, Xbox One launched and with it an OS core shared with Windows 8. Microsoft was so close to one code base you could smell it, but it was still servicing three distinct OSes.

Windows 10 was the chance to bring together this troika and converge engineering efforts. At the same time, however, new applications of technology demanded the addition of more Windows targets: IoT, Microsoft HoloLens, Surface Hub and future members of the Windows device family. Windows 10 needed to be one OS not only for Windows, Phone and Xbox, but also for every future platform.

Microsoft did it. Windows 10 has become the small footprint, one core OS that runs on every device family. This was not as simple as File | Save As. Smart people worked hard to deliver an engineering marvel in an incredible time frame. Windows 10 is the single code base necessary to enable the UAP. Every Microsoft product from this point forward will be written against the single core that makes up Windows 10.

Unity Not Uniformity Uniting the code bases to one core OS doesn't mean one UI across different devices. Windows Phone has a clever, much-loved, one-handed interface significantly distinctive from the 10-foot Xbox experience. The same is true with the Surface Hub, Microsoft HoloLens and Raspberry Pi. These deliver a bulk of their value through their unique experiences. Still, the OS with its libraries, runtime and frameworks, is the same. The device platform and the application platform are the same. The UI and shell features, however, are distinct and tuned to the correct usage model for each device. These are not the same.

In theory, someone could boot to this core OS and even run apps, but no one ever will because it's just a building block. To properly support each form factor, device-specific shell components are added to the core OS—like the Start Menu, specific HID support, and any pieces and parts necessary to enable device-specific features such as desktop applications. These extra components build up from the basic OS to form the different OS SKUs you see as Microsoft products, such as Windows, Server, Xbox and HoloLens.

One Application Platform Here's a fun game to play with your friends. Tell them you're going to embrace the latest Microsoft app innovations, but you won't be targeting Windows 10. How? Next-generation Windows apps won't target the OS. Instead, they'll target the app platform. On Windows, the UAP is a consistent application model and API surface guaranteed across every Windows device.

The UAP isn't a runtime. A Windows app, even one written in a managed language (like Visual Basic or C#) compiles to the metal like any other app. It doesn't run inside a runtime. It doesn't require a runtime. UAP is a common API surface across devices, so targeting the UAP is targeting a specific set and version of APIs.

It's worth pointing out that you build Windows apps and games with the tools and technologies you already know. Windows

apps written in managed languages still enjoy the Microsoft .NET Framework, which itself is only a collection of interfaces, base classes and helper methods. The subset of the full .NET Framework that's used in managed Apps targeting the UAP is called .NET Core. Supplementing this, the majority of the APIs you use in Apps targeting the UAP are in the Windows Runtime, which projects into every language, not just the managed languages.

It's Not Just XAML This article will demonstrate a XAML application, but DirectX and JavaScript apps (Windows Web apps) are also supported by the UAP, just as they were in Windows 8. That being said, it's fascinating to look at the emerging XAML story. XAML is important to many Microsoft platforms—Windows Presentation Foundation (WPF), Silverlight in the browser and on Windows Phone, and now in the Windows UI platform (which started out as code name "Jupiter").

Microsoft Office 2016 is now a family of UAP apps. What UI technology does it use? XAML. Thanks to this relationship, the XAML platform is rich with features and controls that Microsoft and third-party developers, like you, can use in their Windows apps.

When it comes to XAML, Microsoft is all in.

The Windows desktop shell introduces many new features, like the Start Menu and Action Center. What UI technology does it use? XAML. Thanks to this relationship, the XAML platform is hyper-performant, delivering rendering capabilities for sub-second performance if you take advantage of it.

When it comes to XAML, Microsoft is all in. Many important OS apps, like Photos, and MSN apps, such as Health & Fitness, rely on the XAML UI platform to deliver the same rich features every developer can leverage in their Windows apps. What you see in Microsoft apps you can do, too. Not only is the API surface area the same for everyone, so is the XAML UI platform.

Value to Software Developers It's not enough to write an app that can run on every device. To deliver real value to users, your Windows app needs to light up on different devices. Thanks to the extensibility of the UAP, you can include device-specific code in a single binary that will run on every device.

You get more than one single binary with the UAP, you also get one Store for everything—for phone, tablet, desktop and even Xbox apps. The experience is simplified; the monetization is simplified; and the metrics to monitor marketplace success are simplified, as well.

That one Store and the platform let you deploy assets appropriately. This means assets intended for the Xbox experience won't push down to the phone. And the capability in Windows 8 to package assets targeting specific resolutions and scale remain in the UAP.

It's worth pointing out that you build Windows apps and games with the tools and technologies you already know.

As always, you stay in control. Just because the UAP supports every Windows device, doesn't mean you have to. You choose which family of devices your Windows app will support. If your Windows app is phone-only, Xbox-only or HoloLens-only, that's up to you. The Windows Store ensures your app is delivered to the device families you select.

The value to you is not just broader reach, but an easier overall experience. There's one set of tooling, including Visual Studio and Blend for Visual Studio, which you already know and love. There's a familiar set of languages, including JavaScript, the .NET Framework (Visual Basic or C#) and C++/CX. In the end, you and your team build Windows apps using what you already know.

Plenty to Consider

With great power comes great responsibility. The UAP enables Windows apps to run on every type of Windows device. This is awesome, but it comes with a caveat: Not every device provides the same UX. This means that although you get to employ many of the same responsive Web design (RWD) techniques you use in your Web applications, you must think through how your Windows app workflow plays out on different types of devices intended for different types of use. The UAP can only enable support on different devices; it's up to the developer and the designer to build a UX that's great on all of them.

Microsoft provides generous tooling to help build responsive and adaptive Windows apps. Visual Studio can simulate aspect ratio, scale and size during design time. Visual Studio can also simulate (and sometimes emulate) specific device targets even if you don't own the hardware. This lets you test Windows apps and finesse your experiences along the way.

The XAML toolbox has several new controls and enhancements to help you create responsive and adaptive interfaces that look great on every device and every size display. For example, the RelativePanel is new to XAML developers. It inherits from Panel like every other layout control, such as Grid and StackPanel, but allows designers and developers to position child elements relative to other child elements. The resulting XAML visual tree is simpler to render and far simpler to manipulate in response to layout changes. Visual States are another enhancement for XAML developers, making it simpler to respond to layout changes.

This is important: Creating a Windows app that targets multiple devices doesn't mean writing to the lowest common denominator. The UI is rich and so is the feature set. Runtime checking (using the Windows.Foundation.Metadata.ApiInformation namespace) empowers you to include device-specific capabilities that light up your apps for the best possible UX on every device. New features and converged controls are the building blocks you need to dream big.

Anatomy of a Windows App

Now let's take a look at the essential techniques for creating a Windows app that will run across any device family. We're assuming you're familiar with Windows 8.1 Windows Runtime (WinRT) XAML app development. Windows apps are the evolution of those apps. You'll find many resources on Microsoft Virtual Academy for learning, which you can find at aka.ms/w8learn. This article concentrates on new features in the UAP—for running Windows apps across device families.

In Visual Studio 2015, the Templates/Visual C#/Windows Universal node in the New Project dialog has several project templates: the Blank App, the Class Library and the Windows Runtime Component. You use the Blank App template to build a Windows app. The Class Library and Windows Runtime Component templates allow you to encapsulate UI and logic for reuse in other projects. A Class Library supports non-UAP apps, but is limited

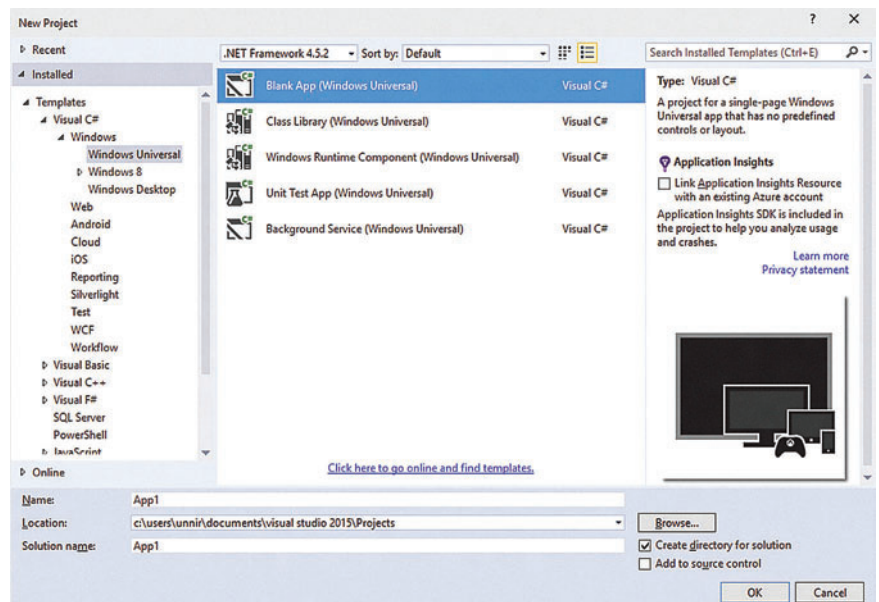
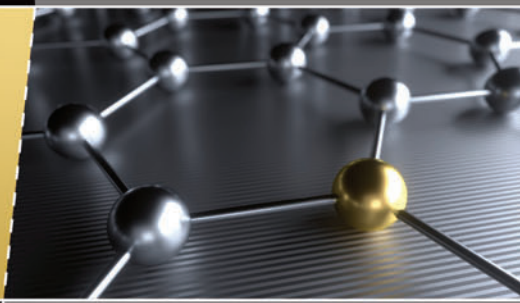


Figure 2 By Default, Windows Apps Now Use the Blank Template

Switch to Amyuni PDF



Create & Edit PDFs in .Net - ActiveX - WinRT

- 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 WinRT Component enables publishing C#, C++CX or Javascript apps to Windows Store
- New Postscript/EPS to PDF conversion module



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



Advanced HTML to PDF & XAML

- Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- Easy Integration and deployment within developer's applications
- WebkitPDF is based on the Webkit Open Source library and Amyuni PDF Creator



High Performance PDF Printer For Desktops and Servers



- Our high-performance printer driver optimized for Web, Application and Print Servers. Print to PDF in a fraction of the time needed with other tools. WHQL tested for Windows 32 and 64-bit including Windows Server 2012 R2 and Windows 8.1
- Standard PDF features included with a number of unique features. Interface with any .Net or ActiveX programming language
- Easy licensing and deployment to fit system administrator's requirements



AMYUNI 

All development tools available at

www.amyuni.com

All trademarks are property of their respective owners. © Amyuni Technologies Inc. All rights reserved.

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

Figure 3 The RelativePanel Allows You to Layout Your Interface in a Simple Way

```
<Grid Background="{StaticResource EggshellBrush}">
  <RelativePanel x:Name="PromoArea">
    <Image x:Name="BannerImage" HorizontalAlignment="Right"
      Height="280" Stretch="UniformToFill"
      Source="Assets/clouds.png"
      RelativePanel.AlignRightWithPanel="True"/>
    <Grid x:Name="BannerText" Margin="24"
      Background="{StaticResource BlueBrush}">
      <StackPanel Margin="12" HorizontalAlignment="Stretch">
        <TextBlock x:Name="Headline" Text="Come fly with us"
          Margin="0,-32,0,0" FontSize="48"
          Foreground="{StaticResource EggshellBrush}"
          FontFamily="{StaticResource LustScriptFont}" />
        <TextBlock x:Name="Subtitle" FontSize="21.333"
          Foreground="{StaticResource EggshellBrush}"
          FontFamily="{StaticResource DomusTitlingFont}" />
        <Run Text="Fly return to London"/>
        <LineBreak/>
        <Run Text="For only $800"/>
      </StackPanel>
    </Grid>
  </RelativePanel>
</Grid>
```

to managed languages; Windows Runtime Components can be shared across languages (including JavaScript and C++/CX), but have rules that constrain their public API surface.

For this sample, choose Blank App, as shown in Figure 2.

Where are all the other templates? Consider the Hub App template that shipped with Windows 8. Many developers used it. Many developers copied it. This rash of “me too” apps created visual consistency within the Windows Store, but didn’t contribute to ecosystem diversity. Now, the Blank App template is at center stage, encouraging developers to create visually consistent yet distinctive interfaces on the platform. Many community-based templates have already started appearing in the Visual Studio Gallery, including one, Template10, that was written by the authors of this article.

Hello World! You’ve created your first Windows app. Though the UI is blank, it can already run on every Windows device. The Visual Studio Solution Explorer reveals how simple a basic Windows app is: a single project with an App.xaml and a single MainPage.xaml file for the initial UI.

Your solution includes other familiar support files. The Package.appxmanifest declares the capabilities the app will request from the user’s machine, such as the user’s location; accessing the camera; and the file system. The XML schema has been expanded, but is about the same as the appxmanifest for Windows 8.1 universal apps.

Where are the two heads? Windows 8 universal apps required both a Phone and a Windows head project. UAP doesn’t require multiple heads. Instead, you adapt your interfaces to accommodate to wherever your Windows app is running. That being said, you certainly can create a multi-headed solution if it suits your development team’s workflow. Both approaches are equally supported.

Including Content When you open MainPage.xaml, you’ll see the improved Visual Studio XAML design-time experience. The designer is richer and faster; the ability to simulate aspect ratio and scale has improved; and the tooling itself is expanded. Now let’s add a little XAML, as shown in Figure 3. (Thanks to our colleague David Crawford for this sample.)

The code in Figure 3 creates the page header for a simple app for a fictional airline. Specifically, it leverages the new XAML RelativePanel, which allows you to rearrange the interface in a simple way. The RelativePanel will position the banner image to the right of the page and contains the grid holding the airline’s recent special offerings.

Adding Some Assets The XAML references three files we’ve added to the Assets folder—an image file, Clouds.png, and two custom fonts, DomusTitlingFont.ttf and LustScriptFont.ttf. The fonts and custom Brush resources are declared in App.xaml:

```
<Application.Resources>
  <SolidColorBrush x:Key="BlueBrush" Color="#FF1C90D1"/>
  <SolidColorBrush x:Key="EggshellBrush" Color="#FFFFFFF7"/>
  <FontFamily x:Key="LustScriptFont">
    Assets/Fonts/LustScriptDisplay.otf#Lust Script Display
  </FontFamily>
  <FontFamily x:Key="DomusTitlingFont">
    Assets/Fonts/DomusTitling.otf#Domus Titling
  </FontFamily>
</Application.Resources>
```

These files are included in the code download that accompanies this article.

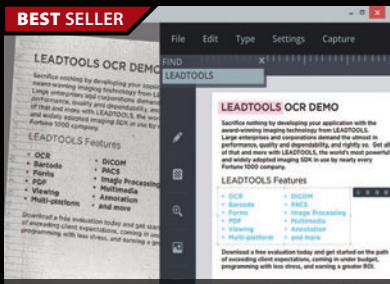
Note that the bitmap image is at one scale. If you want to accommodate devices with higher resolution, you could scale your assets and name them using the appropriate scale factor so every user gets the best visual experience without downloading assets for other scale factors.

Running on Devices Back in MainPage.xaml, the UI is taking shape. To run the app, you can select the target in the Visual Studio device target dropdown. Notice that it includes the Windows Simulator (for touch testing), Local Machine, Remote Machine (for testing ARM) and Device (real phone hardware). The phone emulators are in the same list. Choose and run on Local Machine and afterward on one of the Phone emulators to see your Windows app run on different devices without any special compilations.

The UI adapts to different screens, but device differences extend to more than screen size.

You might have noticed that running a Windows app on the Local Machine, in other words on the desktop of your PC, is a windowed experience and not the full-screen experience of Windows 8. This is because you’re running your app on the desktop SKU of Windows 10. The mobile SKU of Windows 10 still launches Windows apps full screen to make touch navigation easier. But, the desktop SKU of Windows 10 will also launch Windows apps full screen if you choose the touch experience through the Continuum interface on a tablet or a convertible laptop.

Adaptive Interfaces Though the Windows app runs on both devices, upon closer inspection the UI isn’t great on the phone’s smaller screen. The header text is too big for the small screen and is truncated.



LEADTOOLS Document Imaging SDKs V19

from \$2,995.00 SRP



Add powerful document imaging functionality to desktop, tablet, mobile & web applications.

- Universal document viewer & conversion framework for PDF, Office, CAD, TIFF & more
- OCR, MICR, OMR, ICR and Forms Recognition supporting structured & unstructured forms
- PDF & PDF/A Create, Load, Save, View, Edit
- Barcode Detect, Read, Write for UPC, EAN, Code 128, Data Matrix, QR Code, PDF417
- Zero-footprint HTML5/JavaScript UI Controls & Web Services



GrapeCity ComponentOne Studio 2015 V1

from \$1,315.60



.NET Controls for Serious Application Development

- Solutions for Data Visualization, Data Management, Reporting and Business Intelligence
- Hundreds of .NET UI controls for all Visual Studio platforms
- Built-in themes and an array of designers for custom styling
- Adaptive, mobile-friendly and touch-enabled controls



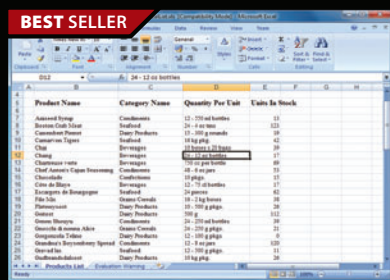
Help & Manual Professional

from \$583.10



Easily create documentation for Windows, the Web and iPad.

- 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 HTML, WebHelp, CHM, PDF, ePUB, RTF, e-book or print
- Styles and Templates give you full design control



Aspose.Total for .NET

from \$2,449.02



Every Aspose .NET component 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

Figure 4 XAML Now Supports Declaring Rules for Adapting an Interface

```

<Grid Background="{StaticResource EggsShellBrush}">
  <VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="WindowStates">
      <VisualState x:Name="NarrowState">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="1"/>
        </VisualState.StateTriggers>
        <VisualState.Setters>
          <Setter Target="BannerImage.Height" Value="120"/>
          <Setter Target="BannerText.(RelativePanel.Below)"
            Value="BannerImage"/>
          <Setter Target="BannerText.Width" Value="660"/>
          <Setter Target="BannerText.Margin" Value="0,0,0,24"/>
          <Setter Target="Headline.FontSize" Value="28"/>
          <Setter Target="Subtitle.FontSize" Value="12"/>
        </VisualState.Setters>
      </VisualState>
      <VisualState x:Name="MediumState">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="660"/>
        </VisualState.StateTriggers>
        <VisualState.Setters>
          <Setter Target="BannerImage.Height" Value="180" />
          <Setter Target="BannerText.(RelativePanel.AlignTopWith)"
            Value="BannerImage"/>
          <Setter Target="Headline.FontSize" Value="28"/>
          <Setter Target="Subtitle.FontSize" Value="14"/>
        </VisualState.Setters>
      </VisualState>
      <VisualState x:Name="WideState">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="1000"/>
        </VisualState.StateTriggers>
        <VisualState.Setters>
          <Setter Target="BannerText.(RelativePanel.AlignTopWith)"
            Value="BannerImage"/>
        </VisualState.Setters>
      </VisualState>
    </VisualStateGroup>
  </VisualStateManager.VisualStateGroups>
  <RelativePanel...

```

This is the beginning of a longer journey to test and improve the UX on the variety of possible devices for this Windows app.

We'll modify the layout of the header when we detect the narrower screen of the phone. It's important, however, to recognize that it's not the phone being detected; it's the width of the screen. This allows for a narrow experience on both the desktop and the phone.

Note that there's no API to detect a phone. However, should your design require one-handed operation specific to the phone and smaller tablets, you can test for the diagonal size of the physical device in a custom Visual State trigger (which isn't discussed in this article).

Visual States are not new to XAML. The Visual State Manager allows developers and designers to define different Visual States (meaning different screen layouts) and to switch among them at runtime. Visual State Adaptive Triggers are new with the UAP. They do away with the programmatic approach to switching Visual States. Instead, you declare when a Visual State should be visible in XAML and the underlying platform does the rest.

Now, modify the XAML in MainPage.XAML, as shown in **Figure 4**.

In **Figure 4**, notice that there are three Visual States declared: NarrowState, WideState and MediumState. Each of these Visual States corresponds to different ranges of screen width. You're free to create as many or as few Visual States as you need to support your targeted device families. The name you use for each Visual State isn't significant.

The XAML also demonstrates Visual State Setters, which are new in the UAP and let you set a discrete property value without the overhead of a storyboard animation. Here, we're using setters to change the relative position of the children in the RelativePanel by setting the

RelativePanel attached property on the child elements, and we're also changing the height of the BannerImage and the FontSize of the text elements. With the Visual States in place, the interface does a great job adapting to a narrower screen. Run it and see!

Figure 5 shows how the UI adapts to changes in screen width. In your Windows app, you can take advantage of Visual State triggers to adjust elements in any way that best serves your users.

The full version of this sample, which is included in the code download that accompanies this article, develops the UI further, and gives additional examples of using the RelativePanel and Visual State Triggers to implement an adaptive UI.

Adaptive Code The UI adapts to different screens, but device differences extend to more than screen size. For example, phones have hardware buttons such as Back and Camera, which might not be present on a different platform such as a PC. The default UAP has most of the API surface Windows apps need, but device-specific functionality is unlocked with Extension SDKs that you add to your projects just like external assemblies, as shown in **Figure 6**. They enable a broader set of device-specific functionality without

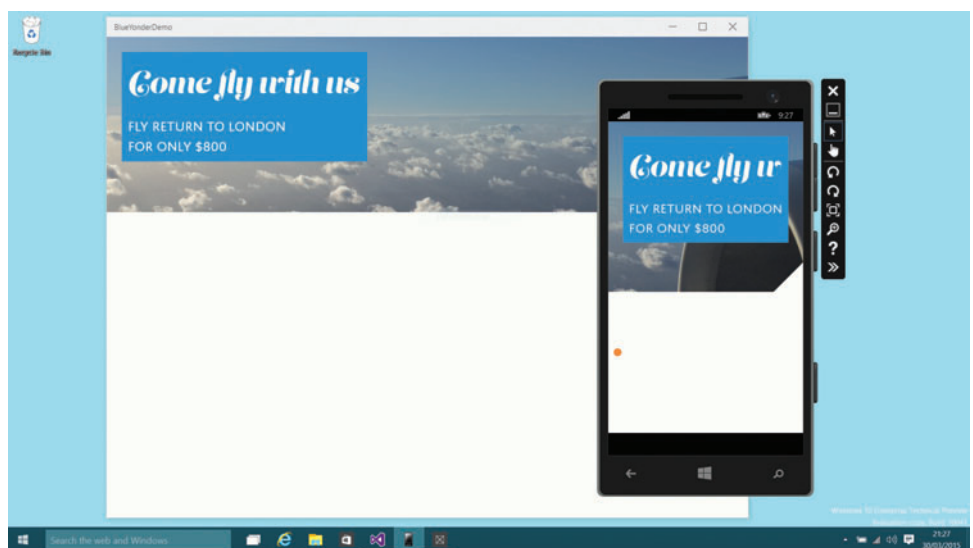


Figure 5 Adapting to Screen Width Changes

invalidating the ability for your app to run on other types of devices.

The two most common platform Extension SDKs are the Desktop and Mobile extensions, which enable functionality unique to their respective Windows SKU. The Mobile extension, for example, enables the APIs necessary to use the hardware camera button.

The Windows Mobile SKU can run on phones and small tablets. However, not all tablets (not even all phones) have a hardware camera button. Extension SDKs enable button support, but don't put buttons on the device. As a result, at run time, you must test for device capabilities before you invoke the capabilities in the Extension SDK.

Just as platform Extension SDKs like Mobile and Desktop unlock the capabilities of devices for Windows apps, custom Extension SDKs add support for additional components, like the Kinect for Windows or third-party hardware. These, too, don't prevent your app from running on other types of devices.

The two most common platform Extension SDKs are the Desktop and Mobile extensions, which enable functionality unique to their respective Windows SKU.

How do you check for device capabilities? You leverage the methods in the `Windows.Foundation.Metadata.ApiInformation` class that return a simple Boolean if a type or method is supported on the current device. You can enable your Windows app to use the Camera button with code like this:

```
if (Windows.Foundation.Metadata.ApiInformation.IsTypePresent(
    "Windows.Phone.UI.Input.HardwareButtons"))
{
    Windows.Phone.UI.Input.HardwareButtons.CameraPressed +=
        HardwareButtons_CameraPressed;
}
```

Notice here how the `Windows.Phone.UI.Input.HardwareButtons` code is allowed to execute only if the Extension SDK is enabled on the device. Unlike with compilation conditionals, testing for capabilities doesn't result in multiple binaries. This means you can light up or gracefully downgrade the UX according to the capabilities of the current device. This is a powerful approach to enable a single binary; it creates limitless variability, letting you make the most of your Windows apps on different device families.

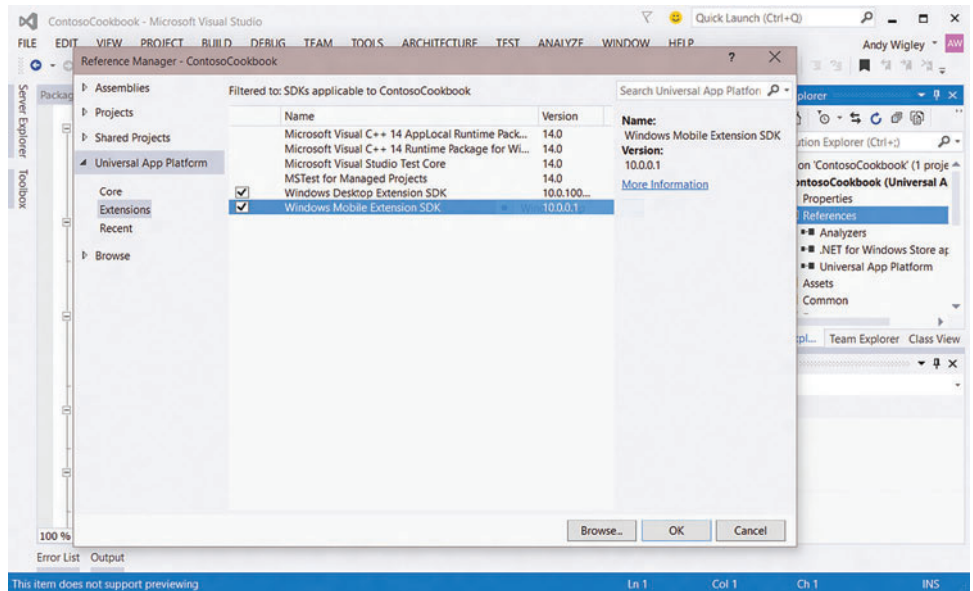


FIGURE 6 Adding an Extension Is as Simple as Adding a Project Reference

Wrapping Up

If you're comfortable with Windows 8 universal app development, then building Windows apps targeting the UAP should feel like home cooking. Windows apps don't target Windows 10; the UAP is the target and it's decoupled from the Windows SKU. UAP increments versions at a cadence apart from Windows. This means Windows apps won't need to retarget every time the Windows OS revs. Windows apps target one or more UAP versions and test for those capabilities just as they test for device capabilities. This flexible approach gives you a nice, clean way to take advantage of future capabilities.

Building a Windows app means your app can run on any Windows device. This gift comes with a real-world caveat: The UAP can run your app, but only developers and designers can make the UI and the code adapt to deliver the best possible UX. If you want to build separate device-specific binaries, you can do that. But, should you choose to build a Windows app that supports multiple device types, all the tooling and infrastructure is in place and ready to make you successful. ■

JERRY NIXON is a developer evangelist for Microsoft from Colorado. Nixon teaches and speaks on Windows, phone and desktop development. His career launched with Microsoft SQL Server 6.5, delivering data-centric solutions while "database developer" was a novel term. He received a civilian Naval Commendation for security work, preceding his work for the startup that would become Microsoft CRM. For 15 years, Nixon built Microsoft-centric, mobile solutions. Today, he speaks on XAML and mobility at events, communities, and universities and the bulk of his free time is spent teaching his three daughters Star Trek character backstories and episode plots.

ANDY WIGLEY is a developer evangelist for Microsoft from the United Kingdom. He joined Microsoft in 2012, and before that worked as a consultant and was a prominent member of the mobile app developers' community. He was proud to be named a Microsoft Most Valuable Professional (MVP) for 10 consecutive years. Wigley is well-known for the popular Windows Phone JumpStart videos that are available on channel9.msdn.com and is happy to be working with Jerry Nixon on a follow-up video series on Windows app development.

Analyze Performance While Debugging in Visual Studio 2015

Dan Taylor and Charles Willis

Many developers spend the majority of their time getting an app to function correctly. Less time is spent focused on app performance. While there have been profiling tools in Visual Studio for quite some time, they were a separate set of tools to learn. Many developers didn't take the time to learn and use them when performance issues would arise.

This article will introduce the new Diagnostic Tools debugger window in Visual Studio 2015. It will also describe how to use it to analyze performance as a regular part of your debugging workflow. I'll first provide an overview of the debugger's features and capabilities and then go on a deep dive walk-through. I'll show you how to use PerfTips to time sections of code between breakpoints and steps, how to use the Diagnostic Tools window to monitor CPU and memory, and how to take snapshots to drill deep into memory growth and leaks.

The features in this article are available for debugging most managed and native projects. Microsoft is continually adding support for more project types and debugging configurations. For up-to-date

This article discusses:

- The new debugging window in Visual Studio 2015
- How to improve app performance while debugging
- Tracking memory growth and leaks

Technologies discussed:

Visual Studio 2015 RC

Code download available at:

aka.ms/diagtoolswndsample

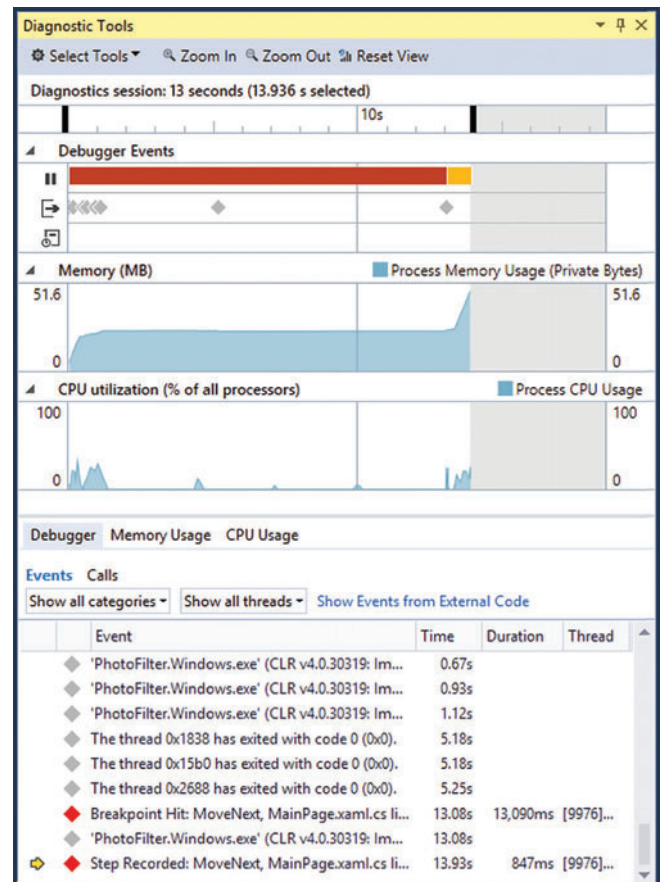


Figure 1 The New Diagnostic Tools Window in Visual Studio 2015

Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/sf

San Francisco

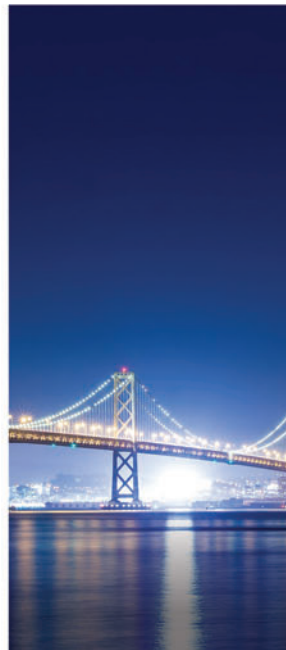
JUNE
15-18

THE FAIRMONT, SAN FRANCISCO, CA



TRACKS INCLUDE:

- Visual Studio / .NET
- JavaScript/HTML5
- Mobile Client
- Database and Analytics
- Cloud Computing
- Windows Client



CODE BY THE BAY

*Register by May 13
and Save \$200!*

USE PROMO CODE VSLMAYT1

Scan the QR code to
register or for more
event details.



FLIP OVER FOR
DETAILS ON
VISUAL STUDIO LIVE!
AUSTIN

SUPPORTED BY



PRODUCED BY



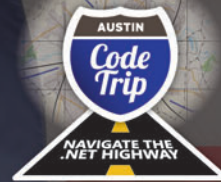
Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/austin

Austin JUNE 1-4

HYATT REGENCY, AUSTIN, TX



**DON'T MESS
WITH CODE**

*Sessions are filling up -
register today!*

USE PROMO CODE VSLMAYTI



Scan the QR code to
register or for more
event details.

GOLD SPONSOR

GitHub

SUPPORTED BY

 **Visual Studio**

msdn
magazine

Visual Studio
MAGAZINE

PRODUCED BY

 **1105 MEDIA**

TRACKS INCLUDE:

- Visual Studio / .NET
- JavaScript/HTML5
- ASP.NET
- Mobile Client
- Database and Analytics
- Cloud Computing
- Windows Client

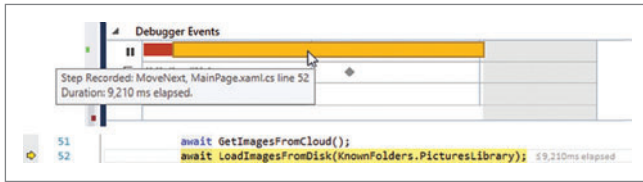


Figure 2 Break Events and PerfTips

information about currently supported features, check out the blog post on the Diagnostic Tools window at aka.ms/diagtoolwindow. A separate article in this issue will explain how you can use IntelliTrace within the Diagnostic Tools window (see “Use IntelliTrace to Diagnose Issues Faster,” p. 38) to quickly identify the root causes of bugs in your code.

Performance While Debugging

Instead of running a full profiling tool, you might take one or more of the following steps:

1. Insert code into an app (such as `System.Diagnostics.Stopwatch`) to measure how long it takes to run between various points, iteratively adding stopwatches as needed to narrow down the hot path.
2. Step through code to see if any particular step “feels slow.”
3. Hit the Break All (“pause”) button at random points to get a feel for how far execution has progressed. In some circles this is referred to as “poor man’s sampling.”
4. Over-optimize code without measuring performance at all, sometimes by applying a set of performance best practices across the entire code base.

These practices are typically not accurate, not a good use of time or both. That’s why there are now performance tools in the debugger. They will help you understand your app’s performance during normal debugging.

Diagnostic Tools Window The primary difference you’ll notice when debugging code in Visual Studio 2015 is the new Diagnostic Tools window that will appear, as shown in **Figure 1**. These Diagnostic Tools present information in two complementary ways. They add graphs to the timeline in the upper-half of the window, and provide detailed information in the tabs on the bottom.

In Visual Studio 2015, you’ll see three tools in the Diagnostics Tools window: Debugger (includes IntelliTrace), Memory Usage

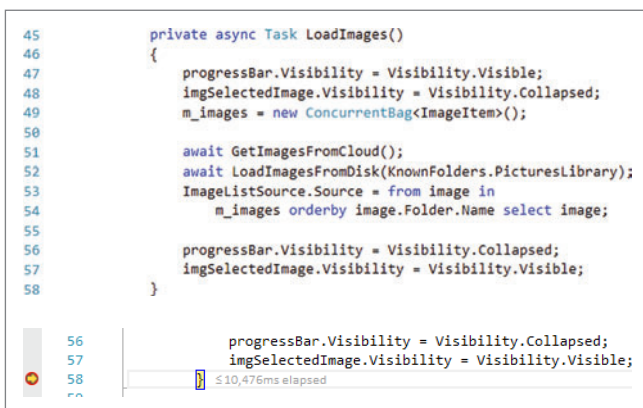


Figure 3 The LoadImages Method

and CPU Usage. You can enable or disable the CPU Usage and Memory Usage tools by clicking on the Select Tools dropdown. The Debugger tool has three tracks that show Break Events, Output Events and IntelliTrace Events.

Break Events History and PerfTips The Break Events let you see how long each section of code took to run. The rectangles represent the duration from when the app started or resumed execution until the Debugger made it pause (see **Figure 2**).

The beginning of the rectangle indicates where you started running the app by using the Continue (F5), Stepping (F10, F11, Shift+F11) or Run-to-cursor (Ctrl+F10) commands. The end of the rectangle indicates where the app stopped because it hit a breakpoint, completed a step or because you used Break All.

The primary difference you’ll notice when debugging code in Visual Studio 2015 is a new Diagnostic Tools window will appear.

The duration of the most recent Break Event is also displayed in the code at the end of the current line in the Debugger. This is called PerfTips. It lets you keep an eye on performance without taking your eyes off your code.

In the details table below the graph, you can also see the history and duration of the Break Events and PerfTips in a tabular format. If you have IntelliTrace, additional events are displayed in the table. You can also use the filter to show only Debugger to see only the history of Break Events.

CPU and Memory Analysis The timeline automatically selects time ranges as you set breakpoints and step. When you hit a breakpoint, the current time range is reset to display only the most recent Break Event. The selection can expand to include the latest Break Event. You can override automatic time range selection by clicking on a Break Event rectangle or by clicking and dragging on the timeline.

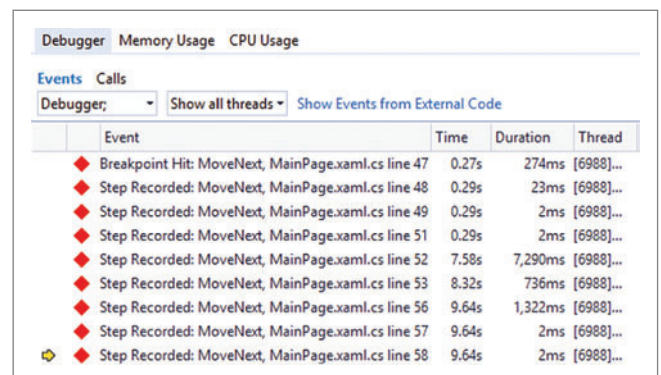


Figure 4 The Debugger Events Table Shows the Elapsed Time for Each Step

```

60 private async Task GetImagesFromCloud()
61 {
62     // Get images list from server
63     HttpClient client = new HttpClient();
64     HttpResponseMessage response = await client.GetAsync(ServerUrl + "/api/Images");
65     response.EnsureSuccessStatusCode();
66     string result = await response.Content.ReadAsStringAsync();
67     dynamic[] pictureList = JsonConvert.DeserializeObject<dynamic[]>(result);
68
69     var folder = KnownFolders.PicturesLibrary;
70     folder = await folder.GetOrCreateFolderAsync("Cloud");
71
72     // Download thumbnails
73     foreach (var image in pictureList)
74     {
75         string fileName = image.Thumbnail;
76         string imageUrl = ServerUrl + "/Images/" + fileName;
77         await DownloadImageAsync(new Uri(imageUrl), folder, fileName);
78     }
79
80     // Download thumbnails
81     var downloadTasks = new List<Task>();
82     foreach (var image in pictureList)
83     {
84         string fileName = image.Thumbnail;
85         string imageUrl = ServerUrl + "/Images/" + fileName;
86         downloadTasks.Add(DownloadImageAsync(new Uri(imageUrl), folder, fileName));
87     }
88     await Task.WhenAll(downloadTasks);
89 }

```

Figure 5 The GetImagesFromCloud Method (Top) and Improved Code (Bottom)

The time range selection lets you correlate ranges on the CPU Usage and Memory Usage graphs so that you can understand the CPU and memory characteristics of specific sections of code. The graphs continue to update while the app is running, which lets you keep an eye on CPU and memory as you interact with your app. You can switch to the Memory Usage tab to take snapshots and view a detailed breakdown of memory usage.

IntelliTrace Performance Insights IntelliTrace (not available in the Visual Studio Community version) lets you gain more insight into performance when debugging managed code. IntelliTrace adds two tracks to the Debugger Events timeline: the Output track and the IntelliTrace track. These events include information shown in the Output window, plus additional events collected by IntelliTrace, such as Exceptions, ADO.NET and so on. The events in these tracks are also shown in the Debugger Events Table.

You can relate IntelliTrace events to CPU Usage spikes and Memory Usage graphs. The time stamps show you how long various actions in your app take. For example, you can add Debug.WriteLine statements in your code and use the timestamps on the Output events to see how long it takes to run from one statement to the next.

Improve Performance and Memory

Now that you've seen the window's features, we'll dive into practical uses of the tools. For this section, we'll walk through solving a set of performance problems in a sample app called PhotoFilter. This app downloads pictures from the cloud and loads pictures from the user's local pictures library so that he can view them and apply image filters.

If you want to follow along, download the source code from aka.ms/diagtoolswndsample. You'll notice different numbers because performance is different on different machines. It will even vary from run to run.

Slow Application Startup When you start debugging the PhotoFilter application, you'll find it takes a long time for the

application to start and pictures to load. This is an obvious problem.

When you're debugging an app's functional issues, you often form a hypothesis and start debugging based on that. In this case, you could hypothesize pictures are slow to load, and look for a good place to set a breakpoint and test that hypothesis. The LoadImages method is a great place to do that.

Set a breakpoint at the start and end of the LoadImages function (as shown in the code in Figure 3) and start debugging (F5). When the code hits the first breakpoint, press Continue (F5) again to run to the second breakpoint. Now there are two break events in the Debugger Events timeline.

The first step shows the application ran for only 274 ms before hitting the first breakpoint. The second shows it took 10,476 ms to run the code in the LoadImages function before hitting the

second breakpoint. You can also see the elapsed time PerfTip in the code showing the same value. So you've narrowed the issue down to the LoadImages function.

To get more information and time how long each line takes, restart debugging so you hit the first breakpoint again. This time, step through each line of code in the method to see which lines are taking the longest. From the PerfTips and duration of the debug break events, you can see GetImagesFromCloud takes 7,290 ms, LoadImagesFromDisk takes 736 ms, the LINQ query takes 1,322 ms and the rest complete in less than 50 ms each.

The timing for all lines is shown in Figure 4. The line numbers shown represent the line of the end of the break event, so Line 52 means how long it took to step over line 51. Now drill further into the GetImagesFromCloud method.

The GetImagesFromCloud method performs two logically separate actions, as shown in Figure 5. It downloads the list of pictures from the server and the thumbnails for each picture synchronously

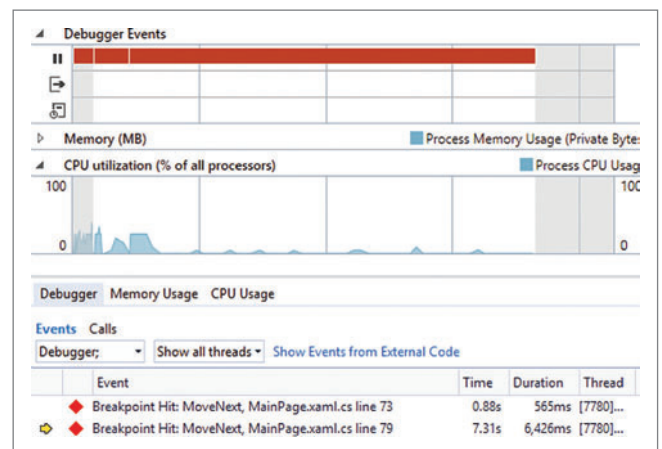


Figure 6 The CPU Usage Graph Indicates Delayed Network I/O

Of drivers who own family cars,

86%

would rather be *driving one of these.*



Move into the Fast Lane with MobileTogether®

In a little over two days, with one developer, we created a full-featured survey application and asked Android™, iOS®, and Windows® mobile users about their dream cars. 86% wanted to go faster.



Ditch the Minivan of Mobile Development

It's no longer practical to wait weeks or months for a cutting-edge mobile business solution. Your team needs to create, test, and deploy to all devices at the speed of business. With MobileTogether, we've removed the roadblocks.

- Drag and drop UI design process
- Powerful visual & functional programming paradigm
- Native apps for all major mobile platforms
- Connectivity to SQL databases, XML, HTML, and more
- Deploy full-featured business app solutions in record time
- Pricing so affordable you might just have money left to start saving for that sports car



www.altova.com/MobileTogether

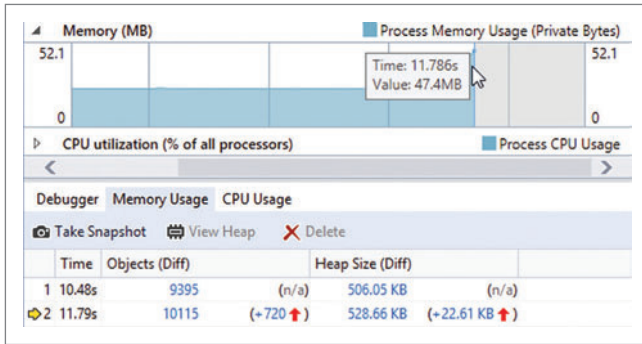


Figure 7 There Is a Noticeable Spike in Memory Usage

(one at a time). You can time these two actions by unsetting your existing breakpoints and placing new ones on the following lines:

```
line 63: HttpClient client = new HttpClient();
line 73: foreach (var image in pictureList)
line 79: }
```

Restart the debugging process and wait until the app hits the first breakpoint. Then let the app run (by pressing F5 to continue) to the second breakpoint. This lets the app retrieve the list of pictures from the cloud. Then let the app run to the second breakpoint to measure downloading the thumbnails from the cloud. The PerfTips and Break Event tells you it took 565 ms to get the list of pictures and 6,426 ms to download the thumbnails. The performance bottleneck is in the thumbnail download.

When you look at the CPU Usage graph (shown in Figure 6) as the method retrieves the list of images, you can see it's relatively high. The graph is fairly flat while the thumbnails download, which indicates this process spends a long time waiting on network I/O.

To minimize the time spent waiting for the round-trips between the client and server, start all the thumbnail download operations at once and wait for them to finish by awaiting completion of the .NET System.Tasks. Replace lines 73 to 79 (from the code in Figure 5) with the following code:

```
// Download thumbnails
var downloadTasks = new List<Task>();
foreach (var image in pictureList)
{
    string fileName = image.Thumbnail;
    string imageUrl = ServerUrl + "/Images/" + fileName;
    downloadTasks.Add(DownloadImageAsync(new Uri(imageUrl), folder, fileName));
}
await Task.WhenAll(downloadTasks);
```

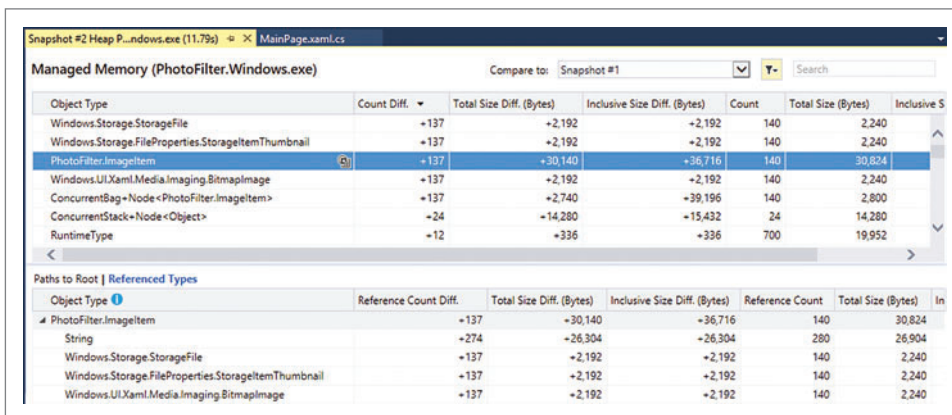


Figure 8 The Heap View Snapshot in Diff Mode

When you time this new version, you can see it takes only 2,424 ms to run. That's about a four-second improvement.

Debug Memory Growth and Leaks If you looked at the Memory Usage graph while diagnosing the slow startup, you might have noticed a sharp increase in memory usage as the app started. The list of thumbnails is a virtualized list, and only one full-size image is displayed at a time. One of the virtues of using a virtualized list is it only loads content displayed on the screen, so you wouldn't expect many thumbnails in memory at once.

To get to the root cause of this issue, you must first find where in the code the memory growth occurs. Then, take snapshots before and after the growth. Compare those snapshots, and you'll find the object types that are contributing most to the growth in memory.

The Memory Usage graph shows a high-level view of how your application is using memory. There's a performance counter named Private Bytes for your app. Private Bytes is a measure of the total amount of memory allocated to a process. That doesn't include memory shared with other processes. It includes the managed heap, native heap, thread stacks and other memory (such as private sections of loaded .dll files).

When developing a new app or diagnosing an issue with an existing one, unexpected growth on the Memory Usage graph will often be the first indication you have code that isn't behaving as intended. Watching the graph, you can use Debugger features such as breakpoints and stepping to narrow down the code path of interest. You can determine from the line numbers and duration shown in the Debugger Events tab back in Figure 4 that the line responsible for the unexpected growth was line 52, the LoadImagesFromDisk method call. Taking a snapshot will often be the next step in pinpointing unexpected memory usage. In the Memory Usage tab, click the Take Snapshot button to generate a snapshot of the heap. You can take snapshots at breakpoints or while the app is running.

If you know which line of code caused the memory usage spike, then you have an idea where to take the first snapshot. Set a breakpoint on the LoadImagesFromDisk method and take a snapshot when your code reaches the breakpoint. This snapshot serves as the baseline.

Next, step over the LoadImagesFromDisk method, and take another snapshot. Now, by diffing the snapshots, you'll be able to see which managed types have been added to the heap as a result of the function call you stepped over. The graph again shows the

memory utilization spike you're investigating (as shown in Figure 7). You can also see by hovering over the graph that the memory was 47.4MB. It's a good idea to make a mental note of the number of megabytes, so you can later verify your fix has had a meaningful impact.

The details view displays a brief overview of each snapshot. The overview includes the snapshot's sequential number, the running time (in seconds) when the snapshot was taken, the size of the heap,

and the number of live objects on the heap. Subsequent snapshots will also display the change in size and object counts from the previous snapshot.

The process of taking a snapshot enumerates only those objects still live on the heap. In other words, if an object is eligible for garbage collection, it's not included in the snapshot. This way, you don't need to worry about when a collection last ran. The data in every snapshot is as if a garbage collection just occurred.

The size of the heap displayed in the snapshot overview will be lower than the Private Bytes displayed in the Memory Usage graph. While the Private Bytes overview shows all types of memory allocated by your process, the snapshot displays the size of all live objects on the managed heap. If you see a large increase in the Memory Usage graph, but growth in the managed heap doesn't account for most of it, the growth is occurring elsewhere in memory.

From the snapshot overview, you can open the Heap View and investigate the contents of the heap by type. Click on the diff link in the Objects (Diff) column for your second snapshot to open the Heap View in a new tab. Clicking that link will sort the types in the Heap View by the number of new objects created since the previous snapshot. That puts the types in which you're interested at the top of the table.

With the Memory graph, you can monitor how your application uses memory as you debug.

The Heap View Snapshot (see **Figure 8**) has two main sections: the Object Type table in the top pane and the References Graph in the lower pane. The Object Type table shows the name, number and size of each object type when the snapshot was taken.

Several of the types in the Heap View are from the framework. If you have Just My Code enabled (the default), these are types that either reference types in your code or are referenced by types in your code. Using this view, you can recognize a type from our code near the top of the table—PhotoFilter.ImageItem.

In **Figure 8**, you can see the Count Diff column shows 137 new ImageItem objects created since the previous snapshot. The top five new object types all have the same number of new objects, so these are likely related.

Let's look at the second pane, the References Graph. If you expect a type to have been cleaned up by the garbage collector, but it still shows up in the Types table, Paths to Root can help you track down what is holding the reference. Paths to Root is one of the two views in the References graph. Paths to Root is a bottom-up tree showing the complete graph of types rooting your selected type. An object is rooted if another live object is holding a reference. Unnecessary rooted objects are often the cause of memory leaks in managed code.

Referenced Types, the other view, is the opposite. For the type selected in the Object Type table, this view shows which other types your selected type is referencing. This information can be useful in determining why objects of the selected type are holding on to more



dtSearch®

Instantly Search Terabytes of Text

25+ fielded and full-text search types

dtSearch's **own document filters** support "Office," PDF, HTML, XML, ZIP, emails (with nested attachments), and many other file types

Supports databases as well as static and dynamic websites

Highlights hits in all of the above

APIs (including 64-bit) for .NET, Java, C++, SQL, etc.

See MS Azure tutorial at www.dtsearch.com/azure

"lightning fast" Redmond Magazine

"covers all data sources" eWeek

"results in less than a second" InfoWorld

hundreds more reviews and developer case studies at www.dtsearch.com

dtSearch products: Web with Spider

Desktop with Spider Engine for Win & .NET-SDK

Network with Spider Engine for Linux-SDK

Publish (portable media) Engine for Android-SDK

Document Filters – included with all products, and also available for separate licensing

beta

Ask about fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

www.dtSearch.com 1-800-IT-FINDS

```

65 public BitmapImage Photo
66 {
67     get
68     {
69         if (m_photo == null)
70         {
71             m_photo = new BitmapImage();
72             m_photo.SetSource(m_thumbnail);
73         }
74         return m_photo;
75     }
76 }
77
78 public async Task LoadImageFromDisk()
79 {
80     m_thumbnail = await m_file.GetThumbnailAsync(ThumbnailMode.PicturesView);
81     m_photo = new BitmapImage();
82     m_photo.SetSource(m_thumbnail);
83 }

```

Figure 9 Code Referencing the m_photo Member Field

memory than expected. This is useful in the current investigation because the types may be using more memory than expected, but they aren't outliving their usefulness.

Select the PhotoFilter.ImagelItem row in the Object Type table. The References graph will update to show the graph for ImagelItem. In the Referenced Types view, you can see the ImagelItem objects are retaining a total of 280 String objects, and 140 each of three framework types: StorageFile, StorageItemThumbnail and BitmapImage.

Total Size makes it look as if String objects are contributing the most to the increase in memory retained by ImagelItem objects. Focusing on the Total Size Diff column makes a lot of sense, but that number alone won't lead you to the root cause. Some framework types, like BitmapImage, only have a small amount of total memory held on the managed heap. The number of BitmapImage instances is a more telling clue. Remember the list of thumbnails in PhotoFilter is virtualized, so it should load these images on demand and make them available for garbage collection when it's done. However, it looks as if all thumbnails are loading in advance. Combined with what you now know about BitmapImage objects being icebergs, continue the investigation focused on those.

Right-click on PhotoFilter.ImagelItem in the References Graph, and select Go To Definition to open the source file for ImagelItem in the editor. ImagelItem defines a member field, m_photo, that's a BitmapImage, as shown in Figure 9.

The first code path that references m_photo is the get method of the Photo property, which is databound to the thumbnail ListView in the UI. It looks like the BitmapImage is being loaded (and thus decoded on the native heap) on demand.

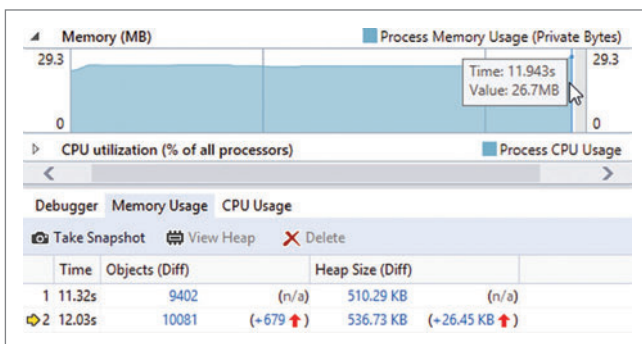


Figure 10 The Memory Graph After Fixing the References Issue

The second code path referencing m_photo is the function LoadImageFromDisk. This item is on the app's startup path. It gets called for each image being displayed, as the app starts up. This effectively preloads all the BitmapImage objects. This behavior works against the virtualized ListView, as all the memory has already been allocated, whether or not the ListView is displaying the image thumbnail. This pre-loading algorithm won't scale well. The more images you have in your Pictures library, the higher the startup memory costs. On-demand loading of the BitmapImage objects is the more scalable solution.

After stopping the debugger, comment out the lines 81 and 82 in LoadImageFromDisk that load the BitmapImage instance. To Verify you've fixed

the memory performance issue without breaking the functionality of the app, rerun the same experiment.

Press F5, and you'll see on the graph total memory usage is now only 26.7MB (see Figure 10). Take another set of snapshots before and immediately after the call to LoadImagesFromDisk and then diff them. You'll see there are still 137 ImagelItem objects, but no BitmapImages (see Figure 11). The BitmapImages will load on-demand once you let the app to continue startup.

As mentioned earlier, this debugger-integrated tool also supports taking snapshots of the native heap, or both the managed and native heaps simultaneously. The heap you profile is based on which debugger you're using:

- The managed-only debugger only takes snapshots of the managed heap.
- The native-only debugger (the default for native projects) only takes snapshots of the native heap.
- The mixed-mode debugger takes snapshots of both the managed and native heap.

You can adjust this setting on the Debug page of your project's properties.

When to Run the Tools Without Debugging

It's important to mention there's extra overhead introduced when you measure performance with the Debugger. The main class of overhead comes from the fact that you're usually running a Debug build of the app. The app you release to users will be a Release build.

In a Debug build, the compiler keeps the executable as close to the original source code as possible in terms of structure and behavior. Everything should work as you'd expect while debugging. On the other hand, a Release build tries to optimize code for performance in ways that degrade the debugging experience. Some examples include in-lining function calls and constant variables, removing unused variables and code paths, and storing variable information in ways that may not be readable by the Debugger.

All of this means CPU-intensive code can sometimes run significantly slower in Debug builds. Non-CPU-intensive operations such as disk I/O and network calls will take the same amount of time. There's usually not a large difference in memory behavior, meaning leaked memory will leak and inefficient memory use will still show as a large increase in both cases.

Object Type	Reference Count Diff.
PhotoFilter.ImageItem	+137
String	+274
Windows.Storage.StorageFile	+137
Windows.Storage.FileProperties.StorageItemThumbnail	+137

Figure 11 The References Graph After Fixing the Memory Issue

The other overhead class to consider is added by the Debugger when it's attached to the target app. The Debugger intercepts events such as module loads and exceptions. It also does other work required to let you set breakpoints and steps. Visual Studio does its best to filter this type of overhead out of the performance tools, but there will still be a small amount of overhead.

If you see an issue in the Release build of an app, it will almost always reproduce in the Debug build, but not necessarily the other way around. For this reason, the Debugger-integrated tools are meant to help you proactively discover performance issues during development. If you find an issue in a Debug build, you can flip to a Release build to see if the issue affects the Release build as well. However, you may decide to go ahead and fix the issue in the Debug build if you decide it's good preventative performance work (that is, fixing the issue reduces the chances of encountering a performance issue later), if you determine the issue is non-CPU intensive (disk or network I/O), or if you would like to speed up the Debug build so your app is snappier during development.

When a performance issue is reported in a Release build, you want to ensure you can reproduce and verify you've fixed the problem. The best way to do this is flip your build to Release mode and run the tools without the Debugger in an environment that closely matches the reported issue.

If you're trying to measure the operational duration, the Debugger-integrated tools will only be accurate to within tens of milliseconds due to the small amount of overhead. If you need a higher level of accuracy, running the tools without the Debugger is a better choice.

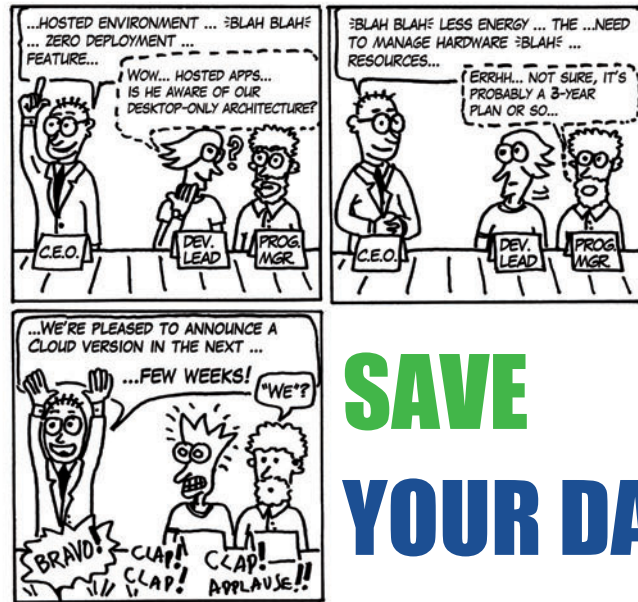
Wrapping Up

You can try out the new Diagnostic Tools debugger window in Visual Studio 2015 by downloading the Visual Studio 2015 RC. Using these new integrated debugging tools can help you improve performance as you debug your apps. ■

DAN TAYLOR is a program manager on the Visual Studio Diagnostics team and has been working on profiling and diagnostic tools for the past two years. Before joining the Visual Studio Diagnostics team, Taylor was a program manager on the .NET Framework team and contributed to numerous performance improvements to the .NET Framework and the CLR.

CHARLES WILLIS has been a program manager on the Visual Studio Diagnostics team since joining Microsoft last year.

THANKS to the following Microsoft technical experts for reviewing this article: Andrew Hall, Daniel Moth and Angelos Petropoulos



**SAVE
YOUR DAY!**



**Use
CodeFluent
Entities**

CodeFluent Entities is a unique product integrated into Visual Studio that allows you to generate database scripts, code (C#, VB), web services and UIs.

*"CodeFluent Entities is a masterpiece software product envisioned and implemented by a group of very talented and experienced people. All that is achieved upon delivering high quality and standardized code and database layers, neatly stitched together and working in unison, lifting the burden from the developers to worry about this aspect on the development process, which could be called plumbing."**

Renato Xavier - Colombaroli - Brazil

* Source : <http://visualstudiogallery.msdn.microsoft.com/B6299BBF-1E11-436D-B618-66E8C16A8410>

To get a license worth \$399 for free
Go to www.softfluent.com/forms/msdn-2015

CodeFluent Entities
tools for developers, by developers

More information: www.softfluent.com Contact us: info@softfluent.com

Use IntelliTrace to Diagnose Issues Faster

Angelos Petropoulos

Think about your typical workflow when you're debugging. Until you've successfully identified the root cause of an issue, you're stuck in a loop of setting breakpoints and repeating testing steps that reproduce the issue. You can now use IntelliTrace to record historical debugging information as the application is running. This helps you break this loop. You can execute the testing steps once to reproduce the issue, then use historical debugging to identify the root cause.

IntelliTrace is the set of historical debugging technologies that extends the debugger in Visual Studio 2015 Enterprise. There's also a standalone component you can use outside of Visual Studio. IntelliTrace records your application execution looking for interesting events. When an interesting event occurs, IntelliTrace automatically records the call stack and local variables while the application continues to run. You can control the events IntelliTrace considers "interesting" through Tools | Options | IntelliTrace | IntelliTrace Events.

IntelliTrace presents you with a history of your app's execution using a timeline (a high-level view of events) and a table that

displays the details of each event. It also gives you access to historical debugging data by extending and integrating with the Visual Studio debugger. This lets you go back in time and see the call stack and local variables of collected events.

IntelliTrace has found a new home in the Diagnostic Tools window in Visual Studio 2015. The Diagnostic Tools window contains the CPU Usage tool and Memory Usage tool, along with IntelliTrace. If your project type and debugging configuration is supported (for up-to-date information, check aka.ms/diagtoolwindow), you'll see a Diagnostic Tools window appear when you start debugging in Visual Studio 2015 (you can press F5 or you can always open it manually using Debug | Show Diagnostic Tools). When you activate Historical Debugging, you'll see something similar to **Figure 1**.

Explore the UI

Following is a list of each component of the IntelliTrace UI and its intended purpose and functionality:

Debugger Events: The Debugger Events details table (see **Figure 2**) is a tabular view of the events IntelliTrace has collected. The columns from left to right are:

1. A pointer to the event for which the debugger is currently showing information; only one row will have the yellow arrow indicating the location of the current instruction pointer and one row might have a pink arrow indicating which historical event you've activated.
2. The icon used to represent this event on the Debugger Events timeline.

This article discusses:

- Historical debugging with IntelliTrace
- How to configure historical debugging
- Analyzing debugging routine results

Technologies discussed:

Visual Studio 2015 Enterprise, IntelliTrace



Extreme Performance Linear Scalability

For .NET & Java Apps

(Available on Microsoft Azure & AWS)

Cache data, reduce expensive database trips, and scale your apps to extreme transaction processing (XTP) with NCache.

In-Memory Distributed Cache

- Extremely fast & linearly scalable with 100% uptime
- Mirrored, Replicated, Partitioned, and Client Cache
- Entity Framework & NHibernate Second Level Cache

ASP.NET Optimization in Web Farms

- ASP.NET Session State storage
- ASP.NET View State cache
- ASP.NET Output Cache provider

Full Integration with Microsoft Visual Studio

- NuGet Package for NCache SDK
- Microsoft Certified for Windows Server 2012 R2



FREE Download

sales@alachisoft.com

US: +1 (925) 236 3830

www.alachisoft.com

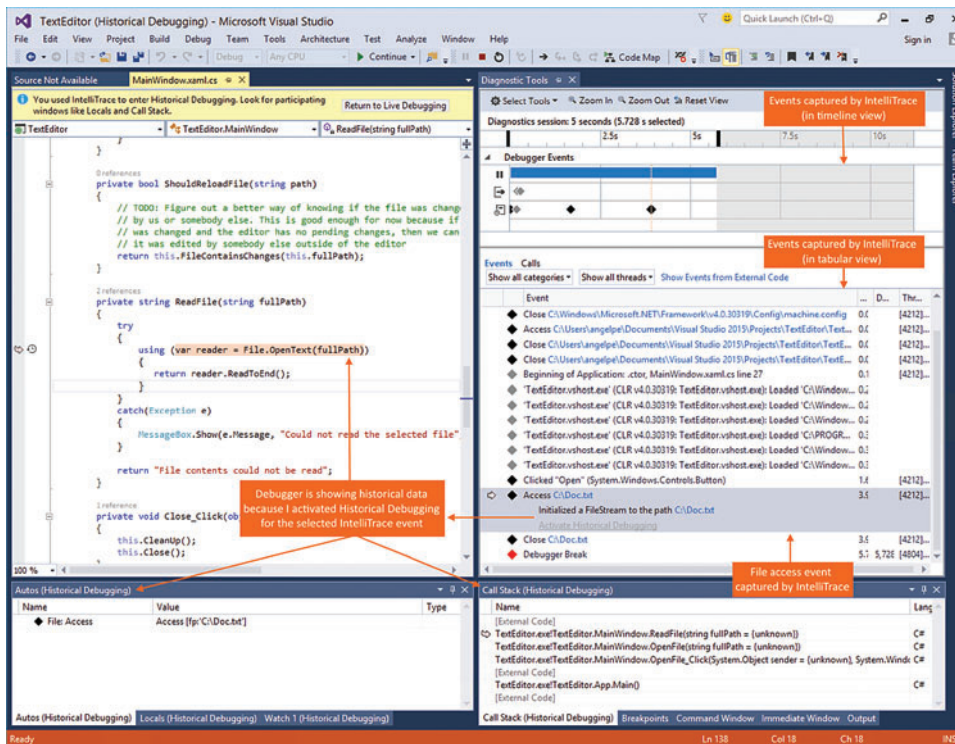


Figure 1 Historical Debugging with IntelliTrace

Show Events from External Code Button: IntelliTrace respects the debugger's Just My Code setting. This means that by default it hides events that originate from non-user code to reduce noise. Clicking on this will bypass the debugger's setting and show events from external code. Most of the time, this will lead to verbose output.

Debugger Events Timeline: This is a graphical view of the events IntelliTrace has collected over time. This is a different view of the same information shown in the Events details table. Use the timeline to get a high-level view, and identify and select areas you want to drill into with the Events details table view. You can filter what you see in the Events details table view by selecting a specific time range.

The Ruler: Above the timeline, there's a ruler that shows you the point in time at which each event

3. A brief description of the event.
4. The number of seconds from the start of the debugging session to the time the event was collected.
5. The duration of the event. (Note: Not all events have a duration.)
6. The thread ID and name that generated the event. (Note: Not all events are associated with a thread)

If you click on an event in the list to expand it, you can then select Activate Historical Debugging and set the debugger to the point where IntelliTrace recorded the selected event.

Category Filter Control: This filtering control lets you hide or show event categories while they're still being collected. If you want to focus on a particular category or you're completely uninterested in another, you can use this to quickly bring them in and out of view. The current list of categories includes: ADO.NET, ASP.NET, Console, Data Binding, Debugger, Environment Variables, Exception, File, Gesture, Lazy Initialization, Output, Registry, Service Model, Threading, Tracing, User Prompt and XAML.

Thread Filter Control: This filtering control lets you hide or show events by the thread from which they were generated, in case you're only interested in diagnosing a specific thread or you're positive executing a specific thread has no issues.

occurred. It also lets you select a specific time range by clicking and dragging. Select a time range to filter the Debugger Events details table.

Break Events Track: Every time a break-related event occurs, it appears on this timeline track. Break events are hitting breakpoints, completed steps, clicking Break All, invoking Debugger.Break or an unhandled exception breaking execution. Think of this as the master timeline track to help you orient where events in the other tracks occurred in the program's execution (because using breakpoints and steps is how you control your application's execution). Clicking on an event in this track applies a time filter that will filter events in the Debugger Events details table. This way you can easily filter only those events that happened when you stepped over a line of code or between you pressing F5 and hitting a breakpoint.

Output Events Track: This track shows events for messages appearing in the Output window. The event categories that appear on this track are: Thrown Exceptions, Program Output (or Console.WriteLine), Module Loaded/Unloaded, Thread Exit and Process Exit. This lets you correlate standard debug output messages with the rest of the debugger's historical information.

IntelliTrace Event Track: Every other event category collected by IntelliTrace appears on this timeline track: ADO.NET, ASP.NET, Console, Data Binding, Environment Variables, File, Gesture, Lazy Initialization, Registry, Service Model, Threading, Tracing, User Prompt and XML.

Diagnostic Tools Toolbar: The toolbar gives you Zoom In and Zoom Out buttons, as well as a Reset View button to reset the timeline back to the default zoom level and clears any existing time selection. This filters all collected data into view. The Select Tools dropdown

Event	Time	Duration	Thread
◆ 'TextEditor.vshost.exe' (CLR v4.0.30319: TextEditor.vshost.exe):...	1.66s		
◆ Clicked "Open" (System.Windows.Controls.Button) The user clicked the btnOpenFile control of type System.Windows.Controls.Button with content "Open". Activate Historical Debugging	9.56s		<No Name>
◆ Clicked "Save" (System.Windows.Controls.Button)	25.30s		<No Name>
◆ Debugger Break	27.85s	27,851ms	Worker Thread

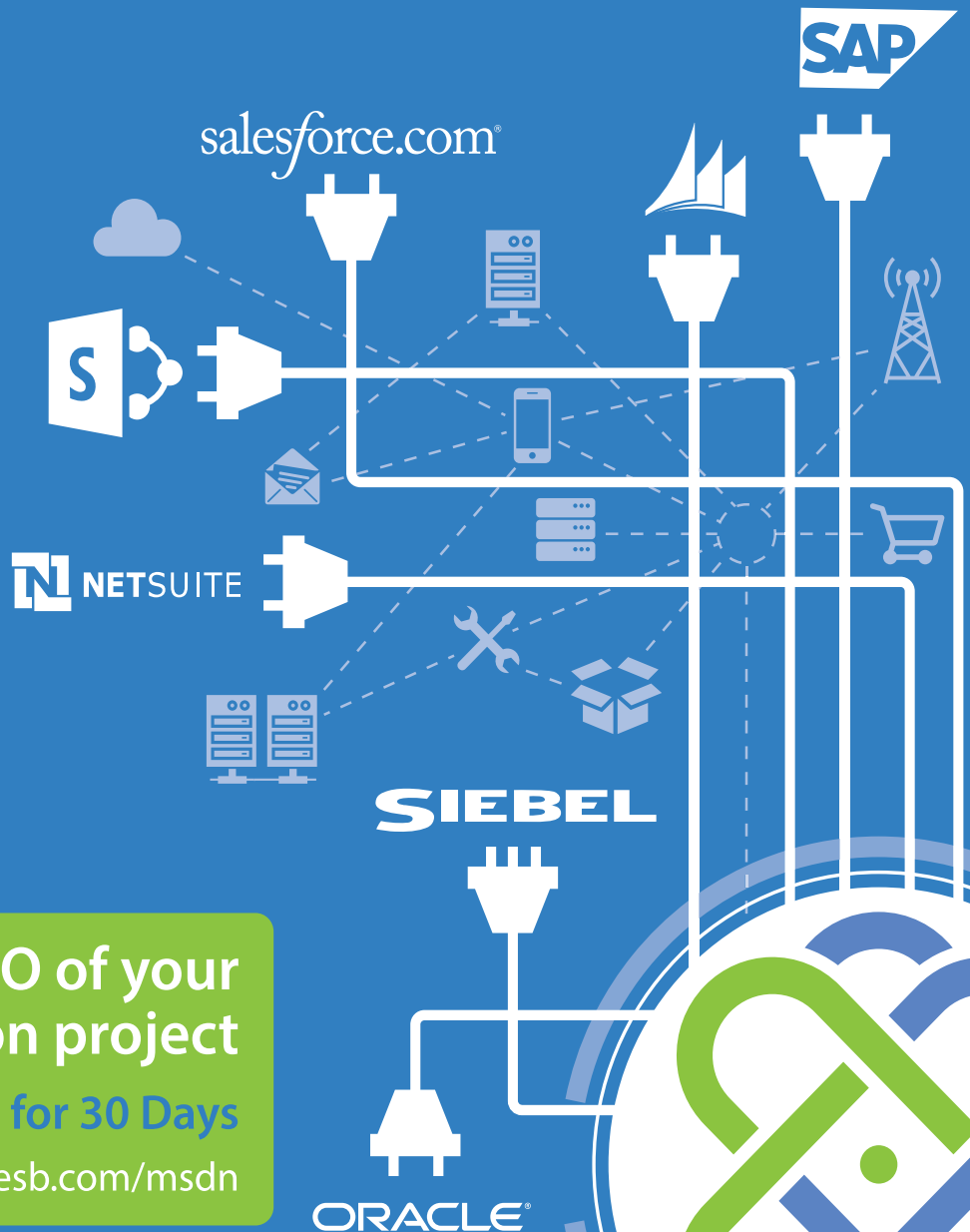
Figure 2 Debugger Events Table

The Simple Solution to Your Complex Integration Challenges

Built by .NET Developers for .NET Developers

Connect

- APIs
- SaaS
- SOA
- LOBs
- IoT
- Workflows
- Hybrid
- On-Premise
- In the Cloud



Be the HERO of your next integration project

Try Neuron ESB FREE for 30 Days

www.neuronesb.com/msdn



neuron  esb


Microsoft Partner
Gold Independent Software Vendor (ISV)

lets you select the tools you would like included in the Diagnostic Tools window, besides IntelliTrace.

You can run multiple diagnostic tools at the same time. The Diagnostic Tools window can host the Memory Usage, CPU Usage tools and IntelliTrace all at the same time. This gives you a holistic view into your application's behavior. For example, **Figure 3** shows how the series of Module Load events increases the memory and CPU usage for an ASP.NET application as it starts up.

Fix a Real Bug with IntelliTrace

Now I'll walk you through fixing a real bug using Live Debugging features of IntelliTrace in Visual Studio 2015 Enterprise. The application I'll debug is a Windows Forms application from CodeProject called SocialClub.

The app maintains a database of members for a social club. The bug is that search behaves erratically after registering a member. To reproduce the bug, I start the application and register a new member. Then I perform a Get All search that's supposed to return all registered members. I expect only one result, but instead I get two (see **Figure 4**). The second search result is unexpected, so that's what I need to fix.

To fix this bug, what should I do next? At this point, my hypothesis is that there's either something wrong with the Get All search function or there's something wrong with the new member registration process. The application has another search mode that takes specific search criteria, so I'll use it to search for the unexpected record returned by Get All (the one with the missing data and Unknown values).

Here are the possible scenarios: If I get no results, it probably means the unexpected record doesn't exist in the database and the issue is with the Get All search function. If I do get a result matching a record with Unknown for Occupation and MaritalStatus, the issue is probably with the registration function entering more records in the database than it should.

So now I perform a search with Unknown for Occupation and Unknown for MaritalStatus, which returns exactly one result: the record I successfully registered as Engineer and Married. That's weird and, unfortunately, it didn't get me any closer to the root

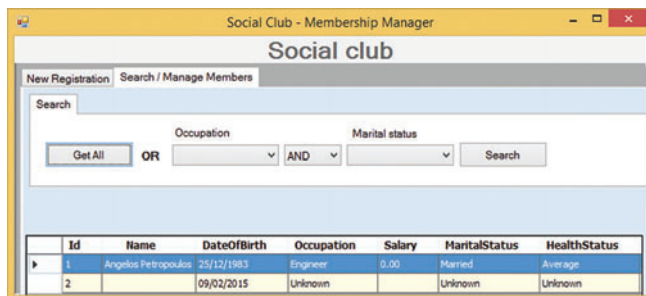


Figure 4 Get All Search Result Contains an Unexpected Record

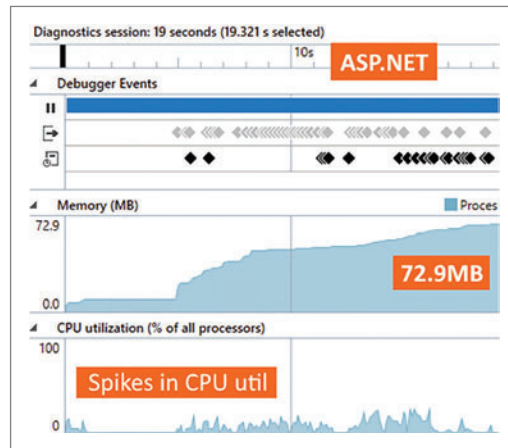


Figure 3 The Diagnostic Tools Window Shows How Your App Is Behaving and Performing

cause of the bug. Instead of spending time setting breakpoints, registering new members and searching for them over and over, I'll see how IntelliTrace can help expedite the investigation.

I want to see the events IntelliTrace has collected, but IntelliTrace events aren't updated until the Debugger breaks the application's execution (meaning it hits a breakpoint). Because I don't have a specific breakpoint in which I'm interested, I simply click Break All on the Visual Studio toolbar. The app is now in a break state with all threads suspended. IntelliTrace is displaying the data it collected in both the timeline and tabular details view of the Diagnostic Tools window.

At this point, I've interacted with the application quite a bit since I started debugging. I've logged in, registered a new member, searched using Get All, and searched with specific search criteria. However, I'm only interested in events that occurred as a direct result of clicking Register. To filter my view to just these events, I hover over the events in the timeline until I find where I clicked Register. Then I drag and select a cluster of events. When I look at my tabular detailed view, after having selected time, I can see the two most recent events listed (other than hitting Break All) are two INSERT statements (see **Figure 5**).

Clicking an event in the list expands it to multiple lines to show the entire executed SQL statement. I can see I have two INSERT statements happening. The second one is inserting a bad record with NULL values. Here are the two SQL statements:

```
Execute Reader "insert [dbo].[ClubMembers]([Name], [DateOfBirth],
[Occupation], [Salary], [MaritalStatus], [HealthStatus], [NumberOfChildren],
[ExpirationDate])values (@0, @1, @2, @3, @4, @5, @6, @7)
select [Id] from [dbo].[ClubMembers] where @@ROWCOUNT > 0 and [Id] =
scope_identity()"
Execute Reader "insert [dbo].[ClubMembers]([Name], [DateOfBirth],
[Occupation], [Salary], [MaritalStatus], [HealthStatus], [NumberOfChildren],
[ExpirationDate])values (null, @0, @1, null, @2, @3, null, @4) select [Id] from
[dbo].[ClubMembers] where @@ROWCOUNT > 0 and [Id] = scope_identity()"
```

You can ignore the SELECT statement that follows the INSERT. That's Entity Framework retrieving the ID of the record it just inserted. The next question is: Why do I get two SQL statements executed for a single click of the Register button? IntelliTrace helps me quickly answer that question by letting me activate Historical Debugging for each of the events (see **Figure 6**) and checking their respective historical call stacks in the Call Stack window.

The first INSERT's historical call stack is:

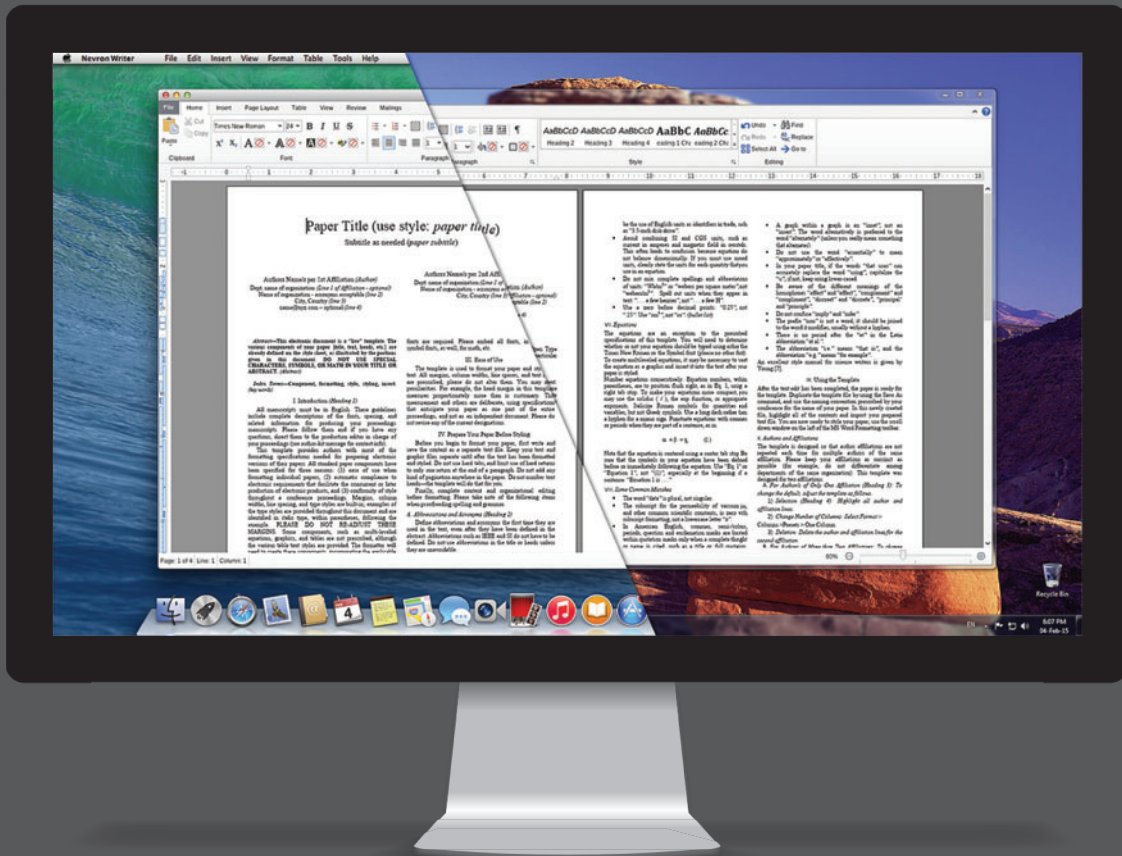
```
John.SocialClub.Data.dll!John.SocialClub.Data.Service.ClubMemberService.Create(...)
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Forms.Membership.
Manage.RegisterMember()
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Forms.Membership.
Manage.Register_Click(...)
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Program.Main()
```

The historical call stack of the second INSERT with the bad record is:

```
John.SocialClub.Data.dll!John.SocialClub.Data.Service.ClubMemberService.Create(...)
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Forms.Membership.
Manage.RegisterMember()
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Forms.Membership.
Manage.btnRegister_MouseClick(...)
John.SocialClub.Desktop.exe!John.SocialClub.Desktop.Program.Main()
```

T NOV TEXT EDITOR for .NET

Available for: WinForms | WPF | Silverlight | Xamarin.Mac | MonoMac



MS Word compatible

Open, edit and save DOCX, RTF and HTML files with great accuracy.



Advanced Editing

Edit paragraphs, sections, tables, columns, lists and more.



Embeddable Widgets

Embed 1D and 2D barcodes, gauges, charts etc.



Export to PDF

Save TXT, RTF, DOCX and even HTML files in PDF.



WYSIWYG

100% identical text layout of screen and printed content. Smooth text zooming.



Tables

Supports tables, nested tables and master cell and advanced styling of everything.

Learn more at www.nevron.com today

www.nevron.com | email@nevron.com | +1 888-201-6088 (Toll free, USA and Canada)

Microsoft, .NET, ASP.NET, SharePoint, SQL Server and Visual Studio are registered trademarks of Microsoft Corporation in the United States and/or other countries. Some Nevron components are only available for certain platforms. For details visit www.nevron.com or send an e-mail to support@nevron.com.



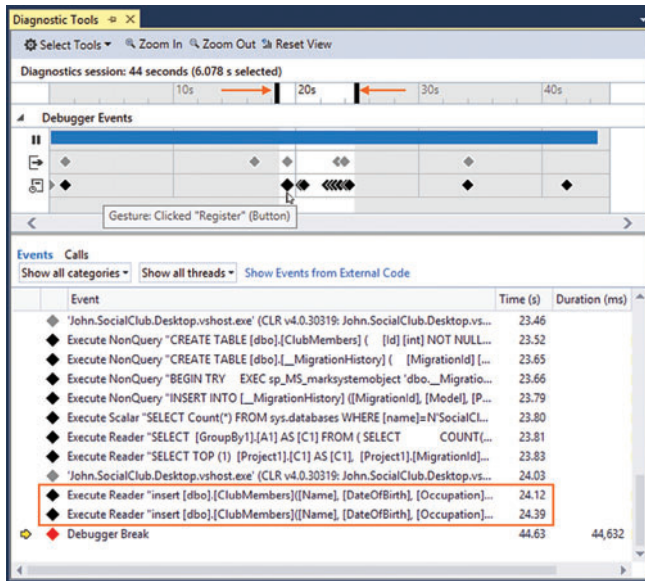


Figure 5 The Tabular Details View Is Filtered to Show Events from the Selected Time Range

Clicking on each frame takes me to the corresponding line of code. After examining the two historical call stacks, I've determined I have two different event handlers subscribed to the same button click: Register_Click(...) and btnRegister_MouseClick(...). Reading the code in those two functions, I quickly deduce that because the form's fields are reset after each new member registration, the first event handler inserts the records to the database correctly. However, the second event handler inserts a record with blank and NULL fields. So I quickly found the bug by using Break All, and then used IntelliTrace to identify and navigate to the offending piece of code.

What If IntelliTrace Events Are Not Enough to Find the Bug?

At this point, as excited as you are about IntelliTrace improving the way you debug, you might be wondering what to do if IntelliTrace doesn't record any interesting events that can lead you to a bug's root cause. Are you out of luck? No, you're not. Don't forget you can control what IntelliTrace events are enabled using Tools | Options | IntelliTrace | IntelliTrace Events. Not all of them are enabled by default, but even enabling all of them might not always be enough for some pesky bugs.

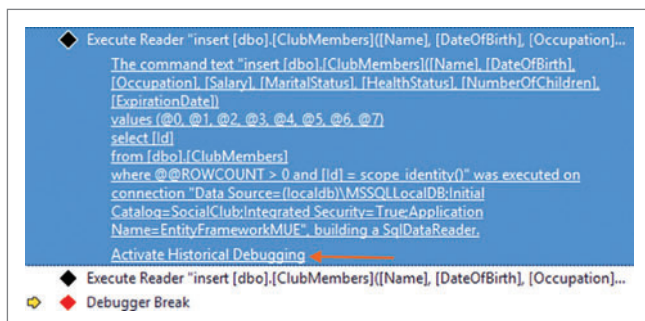


Figure 6 Activate Historical Debugging for the First of Two INSERT Statements

For those tricky issues, you can configure IntelliTrace to record not just events, but also every method call and its parameters. Simply go to Tools | Options | IntelliTrace and select IntelliTrace events and call information. This is a powerful debugging feature, but it comes at a runtime cost. With this setting, IntelliTrace will intercept every method call and record it, which affects the app's performance. That's why it doesn't collect method calls by default. You have to opt in through IntelliTrace settings.

You can view and navigate this new information in two different ways. You can use the sub-tab Calls in the Debugger Events details table, which lists all recorded calls (for more information on the Calls view, go to aka.ms/itracecalls). Another way is to activate Historical Debugging for an event and use the IntelliTrace control within the text editor to navigate back and forth in the application's execution. The controls appear between your code and the instruction pointer. So IntelliTrace has you covered for all your live debugging scenarios.

What If You Can't Reproduce the Bug on a Dev Machine?

This is where Non-Live Debugging with IntelliTrace comes in. So far, I've assumed you know the steps necessary to reproduce the issue you're debugging. That's not always the case. Some of the most difficult and time-consuming bugs are those for which you might not have exact steps to reproduce. IntelliTrace can eliminate this dreaded "no repro" scenario by letting you record the application's execution on a production or test environment. Then you can debug it on your development machine, by exploring the collected information using the same Diagnostic Tools window I've been using here.

IntelliTrace offers a standalone collector you can deploy to other environments to which Visual Studio can't connect. You shouldn't encounter any resistance from your admins because there's no installation involved. It's simply a matter of copying the collector to the target environment. The collector records the application's execution to an .itrace file that you can transfer to your development machine and open with Visual Studio. This scenario is referred to as Non-Live Debugging because you can't control app execution while debugging. For up-to-date information on how to use the IntelliTrace standalone collector, please visit aka.ms/itracecollector.

Wrapping Up

The new IntelliTrace experience and integration with the Diagnostic Tools window have some exciting possibilities. You can keep up-to-date with the latest information on these and other diagnostic-related features by going to aka.ms/DiagnosticsBlog.

ANGELOS PETROPOULOS is a senior program manager on the Visual Studio team. After getting his master's in object-oriented software engineering, he worked as an IT consultant in the United Kingdom. After moving to the United States, he joined the Diagnostic Tools team in Visual Studio and he's now the program manager for IntelliTrace.

THANKS to the following Microsoft technical experts for reviewing this article: Andrew Hall, Daniel Moth, Dan Taylor, Charles Willis

Join us on the Ultimate Code Trip in 2015!



Austin

JUNE 1 - 4
HYATT REGENCY

vslive.com/austin

See pages 48-49 for more info



San Francisco

JUNE 15 - 18
THE FAIRMONT

vslive.com/sf

See pages 50-51 for more info



Redmond

AUGUST 10 - 14
MICROSOFT HEADQUARTERS

vslive.com/redmond

See pages 62-63 for more info



New York

SEPTEMBER 28 - OCTOBER 1
NY MARRIOTT AT BROOKLYN BRIDGE

vslive.com/newyork

See pages 70-71 for more info



Orlando

NOVEMBER 16 - 20
LOEWS ROYAL PACIFIC RESORT

live360events.com

PART OF LIVE! 360

**DETAILS
COMING SOON!**

CONNECT WITH VISUAL STUDIO LIVE!



[@vslive](https://twitter.com/vslive)



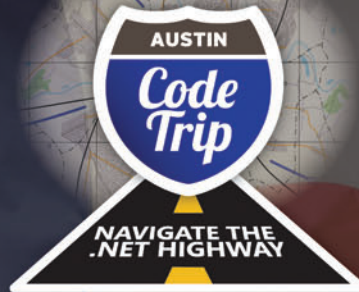
facebook.com - Search "VSLive"



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

TURN THE PAGE FOR
MORE EVENT DETAILS.





DON'T MESS WITH CODE

Visual Studio Live! is pullin' into Austin in June, where the Texas attitude is big, and the code is bigger! For the first time in 8 years, Visual Studio Live! is bringing its unique brand of practical, unbiased, Developer training to the deep heart of Texas. From June 1 - 4, we're offering four days of sessions, workshops and networking events—all designed to make you better at your job.

DEVELOPMENT TRACKS INCLUDE:

- Visual Studio / .NET
- ASP.NET
- JavaScript / HTML5 Client
- Cloud Computing
- Mobile Client
- Database and Analytics
- Windows Client

Sessions are filling up quickly. Register Today!



Scan the QR code to register or for more event details.

Use promo code **VSLMAY5**

vslive.com/austin

ASP.NET	Cloud Computing	Mobile Client	Database and Analytics	JavaScript / HTML5 Client	Visual Studio / .NET	Windows Client
---------	-----------------	---------------	------------------------	---------------------------	----------------------	----------------

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, June 1, 2015 (Separate entry fee required)				
9:00 AM	6:00 PM	M01 Workshop: SQL Server 2014 for Developers - Leonard Lobel	M02 Workshop: Hybrid Mobile Apps with Visual Studio, Cordova, Angular, and Azure - Brian Noyes	M03 Workshop: ALM and DevOps with the Microsoft Stack - Brian Randell		
6:45 PM	9:00 PM	Dine-A-Round with Speakers				

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, June 2, 2015				
8:00 AM	9:00 AM	KEYNOTE: The Future of Application Development - Visual Studio 2015 and .NET 2015 - Jay Schmelzer, Director of Program Management, Visual Studio Team, Microsoft				
9:15 AM	10:30 AM	T01 Building Mobile Cross-Platform Apps with C# and Xamarin - Nick Landry	T02 UX Design Principle Fundamentals for Non-Designers - Billy Hollis	T03 Azure 10-10: Top 10 Azure Announcements in "T-10" Months - Vishwas Lele	T04 A Lap Around Visual Studio 2015 - Robert Green	
10:45 AM	12:00 PM	T05 Building Mobile Cross-Platform Apps in C# with Azure Mobile Services - Nick Landry	T06 Designing and Building UX for Finding and Visualizing Data in XAML Applications - Billy Hollis	T07 Cloud or Not, 10 Reasons Why You Must Know "Websites" - Vishwas Lele	T08 Putting CodedUI Tests on Steroids - Donovan Brown	
12:00 PM	1:30 PM	Lunch — Visit Exhibitors				
1:30 PM	2:45 PM	T09 ASP.NET 5 in All Its Glory - Adam Tuliper	T10 Getting Started with Universal Apps for Windows and Windows Phone - Phillip Japikse	T11 Microsoft Azure SQL Database: SQL Server in the Cloud - Leonard Lobel	T12 Moving DevOps to the Cloud Using Visual Studio Online, Release Management Online, and Azure - Donovan Brown	
3:00 PM	4:15 PM	T13 Hack Proofing Your Web Applications - Adam Tuliper	T14 Building Windows 10 LOB Apps - Robert Green	T15 Database Development with SQL Server Data Tools - Leonard Lobel	T16 What's New in ALM - Brian Randell	
4:15 PM	5:30 PM	Welcome Reception				

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, June 3, 2015				
8:00 AM	9:00 AM	KEYNOTE: User Interaction Design in a NUI World - Tim Huckaby, Founder/Chairman - InterKnowlogy & Actus Interactive Software				
9:15 AM	10:30 AM	W01 Implementing M-V-VM (Model-View-View Model) for WPF - Phillip Japikse	W02 AngularJS 101 - Deborah Kurata	W03 Moving Web Apps to the Cloud - Eric D. Boyd	W04 To Git or Not to Git for Enterprise Development - Benjamin Day	
10:45 AM	12:00 PM	W05 Everything You Always Wanted to Know About REST (But Were Afraid To Ask) - Jon Flanders	W06 AngularJS Forms and Validation - Deborah Kurata	W07 Solving Security and Compliance Challenges with Hybrid Clouds - Eric D. Boyd	W08 Load Testing ASP.NET & WebAPI with Visual Studio - Benjamin Day	
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch — Visit Exhibitors				
1:30 PM	2:45 PM	W09 Building Cross Platform UI with Xamarin.Forms - Walt Ritscher	W10 Securing Angular Apps - Brian Noyes	W11 To Be Announced	W12 Automated Build, Test & Deploy with TFS, ASP.NET, and SQL Server - Benjamin Day	
3:00 PM	4:15 PM	W13 iBeacons and Contextual Location Awareness in iOS and Android Apps - James Montemagno	W14 Build Data-Centric HTML5 Single Page Applications with Breeze - Brian Noyes	W15 Microservices. What's the Big Deal? - Rick Garibay	W16 Stop the Waste and Get Out of (Technical) Debt - Richard Hundhausen	
4:30 PM	5:45 PM	W17 Swift for .NET Developers - Jon Flanders	W18 Take a Gulp, Make a Grunt, and Call Me Bower - Adam Tuliper	W19 A Pragmatic Reference Architecture for the Internet of Things - Rick Garibay	W20 Professional Scrum Development Using Visual Studio 2015 - Richard Hundhausen	
7:00 PM	9:00 PM	Rollin' on the River Bat Cruise				

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, June 4, 2015				
8:00 AM	9:15 AM	TH01 Automated UI Testing for Android and iOS Mobile Apps - James Montemagno	TH02 Busy JavaScript Developer's Guide to ECMAScript 6 - Ted Neward	TH03 SQL 2014 Columnstore Indexes and In-Memory Optimized Tables - Kevin Goff	TH04 What's New in C# 6.0 - Jason Bock	
9:30 AM	10:45 AM	TH05 You Too Can Easily Create an Amazing 3D Application with Unity - Adam Tuliper	TH06 Building Web APIs to Support Your HTML/JavaScript Client - Brian Noyes	TH07 SQL Server Reporting Services - Attendees Choose Topics - Kevin Goff	TH08 Managing the .NET Compiler - Jason Bock	
11:00 AM	12:15 PM	TH09 Strike Up A Conversation with Cortana on Windows Phone - Walt Ritscher	TH10 To Be Announced	TH11 Build Real-Time Websites and Apps with SignalR - Rachel Appel	TH12 Busy Developer's Guide to NoSQL - Ted Neward	
12:15 PM	1:30 PM	Lunch				
1:30 PM	2:45 PM	TH13 Windows, NUI, and You - Brian Randell	TH14 Hate JavaScript? Try TypeScript - Ben Hoelting	TH15 ASP.NET MVC: All Your Tests Are Belong To Us - Rachel Appel	TH16 Microsoft's .NET is Now Open Source and Cross-Platform. Why It Matters. - Mark Rosenberg	
3:00 PM	4:15 PM	TH17 WPF Data Binding in Depth - Brian Noyes	TH18 JavaScript Patterns for the C# Developer - Ben Hoelting	TH19 Building Rich UI with ASP.NET MVC and Bootstrap - Walt Ritscher	TH20 Async and Await Best Practices - Mark Rosenberg	

Sessions and speakers subject to change.





CODE BY THE BAY

Visual Studio Live! returns to **San Francisco June 15 – 18** for the first time since 2009! Bring on the cable cars, Chinatown, Pier 39, Alcatraz, and the Golden Gate Bridge. We can't wait to Code by the Bay!

Join us as we explore the latest features of Visual Studio, JavaScript/HTML5, ASP.NET, Database Analytics, and more over 4 days of sessions and workshops. Code with industry experts, get practical answers to your current challenges, and immerse yourself in what's to come on the .NET horizon.

DEVELOPMENT TRACKS INCLUDE:

- Visual Studio / .NET
- Web Development
- Cloud Computing
- Mobile Client
- Database and Analytics
- Windows Client

REGISTER BY
MAY 13 AND
SAVE \$200!



Scan the QR code to register or for more event details.

Use promo code VSLMAY5

vslive.com/sf

Cloud Computing	Mobile Client	Database and Analytics	Web Development	Visual Studio / .NET	Windows Client
-----------------	---------------	------------------------	-----------------	----------------------	----------------

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, March 16, 2015 (Separate entry fee required)			
9:00 AM	6:00 PM	M01 Workshop: Big Data, Analytics and NoSQL: Everything You Wanted to Learn But Were Afraid to Ask - Andrew Brust	M02 Workshop: Native Mobile App Development for iOS, Android and Windows Using C# - Marcel de Vries	M03 Workshop: ALM and DevOps with the Microsoft Stack - Brian Randell	
6:45 PM	9:00 PM	Dine-A-Round with Speakers			

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, June 16, 2015			
8:00 AM	9:00 AM	KEYNOTE: The Future of Application Development - Visual Studio 2015 and .NET 2015 - Jay Schmelzer, Director of Program Management, Visual Studio Team, Microsoft			
9:15 AM	10:30 AM	T01 Azure 10-10: Top 10 Azure Announcements in "T-10" Months - Vishwas Lele	T02 AngularJS 101 - Deborah Kurata	T03 UX Design Principle Fundamentals for Non-Designers - Billy Hollis	T04 A Lap Around Visual Studio 2015 - Robert Green
10:45 AM	12:00 PM	T05 Cloud or Not, 10 Reasons Why You Must Know "Web Sites" - Vishwas Lele	T06 AngularJS Forms and Validation - Deborah Kurata	T07 Designing and Building UX for Finding and Visualizing Data in XAML Applications - Billy Hollis	T08 To Be Announced
12:00 PM	1:30 PM	Lunch — Visit Exhibitors			
1:30 PM	2:45 PM	T09 Building Mobile Cross-Platform Apps with C# and Xamarin - Nick Landry	T10 ASP.NET 5 in all its Glory - Adam Tuliper	T11 Getting Started with Universal Apps for Windows and Windows Phone - Philip Japikse	T12 To Be Announced
3:00 PM	4:15 PM	T13 Building Mobile Cross-Platform Apps in C# with Azure Mobile Services - Nick Landry	T14 Hack Proofing Your Web Applications - Adam Tuliper	T15 Building Windows 10 LOB Apps - Robert Green	T16 What's New in ALM - Brian Randell
4:15 PM	5:30 PM	Welcome Reception			

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, June 17, 2015			
8:00 AM	9:00 AM	KEYNOTE: Keynote: Cloud Developer - Are You One? - Kris Lankford, Sr. Product Manager, Microsoft			
9:15 AM	10:30 AM	W01 iOS Development - What They Don't Tell You - Jon Flanders	W02 Implementing M-V-VM (Model-View-View Model) for WPF - Philip Japikse	W03 Moving Web Apps to the Cloud - Eric D. Boyd	W04 Stop the Waste and Get Out of (Technical) Debt - Richard Hundhausen
10:45 AM	12:00 PM	W05 Swift for .NET Developers - Jon Flanders	W06 Strike Up a Conversation with Cortana on Windows Phone - Walt Ritscher	W07 Solving Security and Compliance Challenges with Hybrid Clouds - Eric D. Boyd	W08 Best Practices for Using Open Source Software in the Enterprise - Marcel de Vries
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch — Visit Exhibitors			
1:30 PM	2:45 PM	W09 Building Cross Platform UI with Xamarin.Forms - Walt Ritscher	W10 Take a Gulp, Make a Grunt, and Call Me Bower - Adam Tuliper	W11 Introduction to R for C# Programmers - James McCaffrey	W12 Automated Build, Test & Deploy with TFS, ASP.NET, and SQL Server - Benjamin Day
3:00 PM	4:15 PM	W13 Adding Analytics to Your Mobile Apps on Any Platform with Microsoft Application Insights and Hockeyapp - Marcel de Vries	W14 Securing Angular Apps - Brian Noyes	W15 Microservices. What's the Big Deal? - Rick Garibay	W16 Professional Scrum Development Using Visual Studio 2015 - Richard Hundhausen
4:30 PM	5:45 PM	W17 Creating Applications Using Android Studio - Kevin Ford	W18 - Build Data-Centric HTML5 Single Page Applications with Breeze - Brian Noyes	W19 A Pragmatic Reference Architecture for The Internet of Things - Rick Garibay	W20 Load Testing ASP.NET & WebAPI with Visual Studio - Benjamin Day
7:00 PM	9:00 PM	Visual Studio Live! Evening Event			

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, June 18, 2015			
8:00 AM	9:15 AM	TH01 Using Multi-Device Hybrid Apps to Create Cordova Applications - Kevin Ford	TH02 Build Real-Time Websites and Apps with SignalR	TH03 Windows, NUI, and You - Brian Randell	TH04 To Git or Not to Git for Enterprise Development - Benjamin Day
9:30 AM	10:45 AM	TH05 Everything You Always Wanted To Know About REST (But Were Afraid To Ask) - Jon Flanders	TH06 Knocking it Out of the Park with Knockout.JS - Miguel Castro	TH07 Building Rich Data Web APIs with ASP.NET OData - Brian Noyes	TH08 What's New in C# 6.0 - Jason Bock
11:00 AM	12:15 PM	TH09 Comparing Performance of Different Mobile Platforms - Kevin Ford	TH10 To Be Announced	TH11 Power BI 2.0: Analytics in the Cloud and in Excel - Andrew Brust	TH12 Microsoft's .NET is Now Open Source and Cross-Platform. Why it Matters. - Mark Rosenberg
12:15 PM	1:30 PM	Lunch			
1:30 PM	2:45 PM	TH13 Programming WPF with MVVM - Advanced Topics - Miguel Castro	TH14 Busy JavaScript Developer's Guide to ECMAScript 6 - Ted Neward	TH15 Big Data and Hadoop with Azure HDInsight - Andrew Brust	TH16 Managing the .NET Compiler - Jason Bock
3:00 PM	4:15 PM	TH17 Extending XAML To Overcome Pretty Much Any Limitation - Miguel Castro	TH18 ASP.NET MVC: All Your Tests Are Belong To Us	TH19 Busy Developer's Guide to NoSQL - Ted Neward	TH20 Async and Await Best Practices - Mark Rosenberg

Sessions and speakers subject to change.

What Every Programmer Should Know About Compiler Optimizations, Part 2

Hadi Brais

Welcome to the **second** part of my series on compiler optimizations. In the first article (msdn.microsoft.com/magazine/dn904673), I discussed function inlining, loop unrolling, loop-invariant code motion, automatic vectorization and COMDAT optimizations. In this second article, I'm going to look at two other optimizations—register allocation and instruction scheduling. As always, I'll be focusing on the Visual C++ compiler, with a brief discussion of how things work in the Microsoft .NET Framework. I'll be using Visual Studio 2013 to compile the code. Let's get started.

Register Allocation

Register allocation is the process of allocating a set of variables to the available registers so that they don't have to be allocated in memory. This process is usually performed at the level of a whole

function. However, especially when Link-Time Code Generation (/LTCG) is enabled, the process can be performed across functions, which may result in a more efficient allocation. (In this section, all variables are automatic—those whose lifetimes are determined syntactically—unless otherwise mentioned.)

Register allocation is a particularly important optimization. To understand this, let's see how much it takes to access the different levels of memory. Accessing a register takes less than one processor cycle. Accessing the cache is a little slower and takes from few to tens of cycles. Accessing the (remote) DRAM memory is even slower than that. Finally, accessing the hard drive is terribly slow and can take millions of cycles! Also, a memory access increases traffic to shared caches and main memory. Register allocation reduces the number of memory accesses by utilizing the available registers as much as possible.

The compiler tries to allocate a register to each variable, ideally until all instructions involving that variable are executed. If this isn't possible, which is common for reasons I'll discuss shortly, one or more variables have to be spilled into memory so they must be loaded and stored frequently. Register pressure refers to the number of registers that have been spilled due to the unavailability of registers. Larger register pressure means more memory accesses, and more memory accesses can slow not only the program itself, but also bring the whole system to a crawl.

Modern x86 processors offer the following registers to be allocated by compilers: eight 32-bit general-purpose registers, eight

This article discusses:

- Register allocation
- Instruction scheduling
- The volatile and __restrict keywords, and the /favor switch

Technologies discussed:

Visual Studio 2013, Visual C++ Compiler, Microsoft .NET Framework

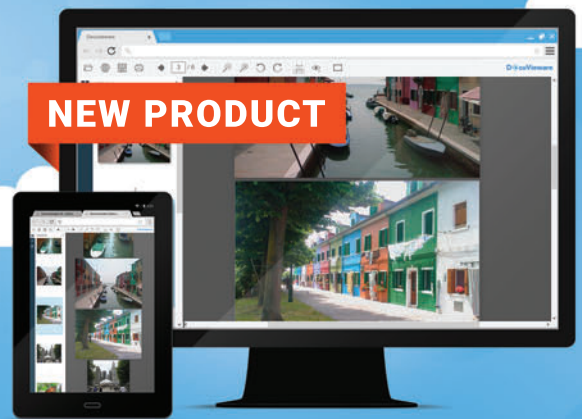
Code download available at:

msdn.microsoft.com/magazine/msdnmag0515

DocuVieware

Universal HTML5 Viewer and Document Management Kit

View, annotate and manage any document, on any device, on any browser through a fully customizable zero-footprint HTML5 Control. Supports 90+ file formats, including PDF, TIFF and SVG.



Why DocuVieware?



Cross-platform, any browser, **zero-footprint** solution (no client-side install).



Super-**easy integration** within existing web applications.



Fast and **crystal-clear rendering** of documents and annotations.



Fully **customizable UI** look and feel, including convenient **snap-ins**.



Mobile devices optimization & **responsive** design.



Built-in **annotations**, **thumbnails**, **bookmarks** and **text search** tools.

Works in all modern browsers ...

IE9+, Chrome, Chrome for Android, Firefox, Firefox for Android, Opera, Safari 5+, Mobile Safari.



... so it works on all devices.

PC, Mac, tablets and smartphones.



80-bit floating-point registers and eight 128-bit vector registers. All x64 processors offer 16 64-bit general-purpose registers, eight 80-bit floating-point registers and at least 16 vector registers—each at least 128 bits wide. Modern 32-bit ARM processors offer 15 32-bit general-purpose registers and 32 64-bit floating-point registers. All 64-bit ARM processors offer 31 64-bit general-purpose registers, 32 128-bit floating-point registers and 16 128-bit vector registers (NEON). All of these are available for register allocation (and you can also add to the list the registers offered by the graphics card). When a local variable can't be allocated to any of the available registers, it has to be allocated on the stack. This happens in almost every function for various reasons, as I'll discuss. Consider the program in **Figure 1**. The program doesn't do anything meaningful, but it serves as a good example to demonstrate register allocation.

Modern compilers are capable of devising a good allocation, but not the optimal one.

Before it allocates the available registers to variables, the compiler first analyzes the use of all declared variables within the function (or across functions in case of /LTCG) to determine which sets of variables are live at the same time, and estimates the number of times each variable is being accessed. Two variables from different sets can be allocated the same register. If there are no suitable registers for some variables of the same set, these variables have to be spilled. The compiler tries to choose the least-accessed variables to be spilled in an attempt to minimize the total number of memory accesses. That's the general idea. However, there are many special cases in which it's possible to find a better allocation. Modern compilers are capable of devising a good allocation, but not the optimal one. It's very, very hard for a mortal to do better, though.

With this in mind, I'll compile the program in **Figure 1** with optimizations enabled and see how the compiler will allocate local variables to registers. There are four variables to be allocated: *n*, *m*, *i* and *j*. I'll assume the target in this case is the x86 platform. By examining the generated assembly code (/FA), I notice the variable *n* has been allocated to the register ESI, the variable *m* has been allocated to ECX, and both *i* and *j* have been allocated to EAX. Note how the compiler cleverly reused EAX for two variables because their lifetimes

Figure 1 Register Allocation Example Program

```
#include <stdio.h>

int main() {
    int n = 0, m;
    scanf_s("%d", &m);
    for (int i = 0; i < m; ++i){
        n += i;
    }
    for (int j = 0; j < m; ++j){
        n += j;
    }
    printf("%d", n);
    return 0;
}
```

do not intersect. Also note that the compiler has reserved a space on the stack for *m* because its address is taken. On the x64 platform, the variable *n* will be allocated to the register EDI, the variable *m* will be allocated to EDX, *i* to EAX, and *j* to EBX. For some reason, the compiler didn't allocate *i* and *j* to the same register this time.

Was that an optimal allocation? No. The problem is in using ESI and EDI. These registers are callee-saved registers, meaning that the called function has to make sure that the values those registers hold at exit are the same at entrance. That's why the compiler had to emit an instruction at the entrance of the function to push ESI/EDI on the stack and another instruction at the exit to pop them from the stack. The compiler could've avoided this on both platforms by using a caller-saved register, such as EDX. Such deficiencies in the register allocation algorithm can sometimes be mitigated by function inlining. Many other optimizations can render the code amenable to a more efficient register allocation, such as dead code elimination, common subexpression elimination and instruction scheduling.

It's actually common that variables have disjoint lifetimes, so allocating the same register to all of them is very economic. But what if you run out of registers to accommodate any of them? You have to spill them. However, you can do that in a clever way. You spill all of them to the same location on the stack. This optimization is called stack packing and it's supported by Visual C++. Stack packing reduces the size of the stack frame and may improve the data cache hit ratio, resulting in better performance.

Unfortunately, things are not that simple. From a theoretical perspective, (near) optimal register allocation can be achieved. However, practically, there are many reasons why this may not be possible:

- The available registers on the x86 and x64 platforms (mentioned earlier) and any other modern platform (such as ARM) can't be used arbitrarily. There are complex restrictions. Each instruction imposes restrictions as to which registers can be used as operands. Therefore, if you want to use an instruction, you have to use the allowed registers to pass to it the required operands. Also, the results of some instructions are stored in predetermined registers whose values are assumed by the instructions to be volatile. There could be a different sequence of instructions that perform the same computation but lets you perform a more efficient register allocation. The problems of instruction selection, instruction scheduling and register allocation are frightfully entangled.
- Not all variables are of primitive types. It's not uncommon to have automatic structures and arrays. Such variables can't be directly considered for register allocation. However, they can be discretely allocated to registers. Current compilers aren't that good yet.
- The calling convention of a function imposes a fixed allocation for some arguments while rendering others ineligible for allocation irrespective of the availability of registers. More on this issue a bit later. Also, the notions of caller-saved and callee-saved registers make things trickier.
- If the address of a variable is taken, the variable better be stored in a location that has an address. A register doesn't have an address, so it has to be stored in memory whether it's available or not.

WPF lives!



➔ **XCEED Business Suite for WPF**

The essential set of WPF controls for all your line-of-business solutions. Includes the industry-leading **Xceed DataGrid for WPF**.
A total of 85 tools!

All of this might seem to you that current compilers are terrible at register allocation. However, they're reasonably good at it and getting better, very slowly. Also, can you imagine yourself writing assembly code while thinking about all of this?

You can help the compiler to potentially find a better allocation by enabling `/LTCG` when targeting x86 architectures. If you specify the `/GL` compiler switch, the generated OBJ files will contain C Intermediate Language (CIL) code rather than assembly code. Function calling conventions aren't incorporated in the CIL code. If a particular function isn't defined to be exported from the output executable, the compiler can violate its calling convention to improve its performance. This is possible because it can identify all the call sites of the function. Visual C++ does take advantage of this by making all arguments of the function eligible for register allocation regardless of the calling convention. Even if register allocation can't be improved, the compiler will try to reorder parameters for a more economic alignment and even remove unused parameters. Without the `/GL` switch, the resulting OBJ files contain binary code in which calling conventions have already been considered. If an assembly OBJ file has a call site to a function in a CIL OBJ file or if the address of the function is taken anywhere or if it's virtual, the compiler can no longer optimize its calling convention. Without `/LTCG`, by default, all functions and methods have external linkage, so the compiler can't apply this technique. However, if a function in an OBJ file has been explicitly defined with internal linkage, the compiler can apply this technique to it, but only within an OBJ file. This technique, referred to by the documentation as a custom calling convention, is important when targeting x86 architectures because the default calling convention, namely `__cdecl`, isn't efficient. On the other hand, the `__fastcall` calling convention on the x64 architecture is very efficient because the first four arguments are passed via registers. For this reason, the custom calling convention is performed only when targeting x86.

All but the simplest instructions are executed in multiple stages, where each stage is handled by a specific unit of the processor.

Note that even if `/LTCG` is enabled, the calling convention of an exported function or method can't be violated because it's impossible for the compiler to locate all call sites, just as in all the cases previously mentioned.

The effectiveness of register allocation depends on the accuracy of the estimated number of accesses to the variables. Most functions contain conditional statements, jeopardizing the accuracy of these estimates. Profile-guided optimization can be used to fine-tune these estimates.

When `/LTCG` is enabled and the target platform is x64, the compiler performs interprocedural register allocation. This means it will consider the variables declared within a chain of functions and try

to find a better allocation depending on the restrictions imposed by the code in each function. Otherwise, the compiler performs global register allocation in which each function is processed separately ("global" here means the whole function).

Both C and C++ offer the register keyword, enabling the programmer to provide a hint to the compiler regarding which variables to store in registers. In fact, the first version of C introduced this keyword and it was useful at that time (circa 1972) because no one knew how to perform register allocation effectively. (A FORTRAN IV compiler developed by IBM Corp. in the late 60s for the S/360 series could perform simple register allocation, though. Most S/360 models offered 16 32-bit general-purpose registers and four 64-bit floating-point registers!) Also, just as with many other features of C, the register keyword makes it easier to write C compilers. Nearly a decade later, C++ was created and it offered the register keyword because C was considered to be a subset of C++. (Unfortunately, there are many subtle differences.) Since the early 80s, many effective register allocation algorithms have been implemented, so the existence of the keyword has created a lot of confusion to this day. Most production languages that have been created since then don't offer such a keyword (including C# and Visual Basic). This keyword has been deprecated since C++11, but not in the latest version of C, C11. This keyword should be used only for writing benchmarks. The Visual C++ compiler does respect this keyword, if possible. C doesn't allow the address of a register variable to be taken. C++, however, does allow it but then the compiler has to store the variable in an addressable location instead of in a register, violating its manually specified storage class.

When targeting the CLR, the compiler has to emit Common Intermediate Language (CIL) code that models a stack machine. In this case, the compiler won't perform register allocation (though if some of the emitted code is native, register allocation will be performed on it, of course) and will postpone it until runtime to be performed by the just-in-time (JIT) compiler (or the Visual C++ back end in the case of .NET Native compilation). RyuJIT, the JIT compiler that ships with the .NET Framework 4.5.1 and later, implements a pretty decent register allocation algorithm.

Instruction Scheduling

Register allocation and instruction scheduling are among the last optimizations performed by the compiler before it emits the binary.

All but the simplest instructions are executed in multiple stages, where each stage is handled by a specific unit of the processor. To utilize all of these units as much as possible, the processor issues multiple instructions in a pipelined fashion such that different instructions are executing at different stages at the same time. This can significantly improve performance. However, if one of these instructions isn't ready for execution for some reason, the whole pipeline stalls. This can happen for many reasons, including waiting for another instruction to commit its result; waiting for data to come from memory or disk; or waiting for an I/O operation to complete.

Instruction scheduling is a technique that can mitigate this problem. There are two kinds of instruction scheduling:

- Compiler-based: The compiler analyzes the instructions of a function to determine those instructions that might

DEVELOPED FOR INTUITIVE USE

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

ceTe Software's DynamicPDF products provide real-time PDF generation, manipulation, conversion, printing, viewing, and much more. Providing the best of both worlds, the object models are extremely flexible but still supply the rich features you need as a developer. Reliable and efficient, the high-performance software is easy to learn and use. If you do encounter a question with any of our components, simply contact ceTe Software's readily available, industry-leading support team.



DynamicPDF

WWW.DYNAMICPDF.COM



TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.631.5006 | +1 410.772.8620

ceTe software

Figure 2 Instruction Scheduling Example Program

```
#include <stdio.h>
#include <time.h>

__declspec(noinline) int compute(){
    /* Some code here */
    return 0;
}

int main() {
    time_t t0 = clock();
    /* Target location */
    int result = compute();
    time_t t1 = clock(); /* Function call to be moved */
    printf("Result (%d) computed in %lld ticks.", result, t1 - t0);
    return 0;
}
```

stall the pipeline. Then it tries to find a different order of the instructions to minimize the cost of the expected stalls while at the same time preserving the correctness of the program. This is called instruction reordering.

- **Hardware-based:** The majority of modern x86, x64 and ARM processors are capable of looking ahead into the stream of instructions (micro-ops, to be accurate) and issuing those instructions whose operands and the required functional unit are available for execution. This is called out-of-order (OoOE or 3OE) or dynamic execution. The result is that the program is being executed in an order that's different from the original.

There are other reasons that might cause the compiler to reorder certain instructions. For example, the compiler might reorder nested loops so that the code exhibits better locality of reference (this optimization is called loop interchange). Another example is to reduce the costs of register spilling by making instructions that use the same value loaded from memory consecutive so that the value is loaded once. Yet another example is to reduce data and instruction cache misses.

As a programmer, you don't have to know how a compiler or a processor performs instruction scheduling. However, you should be aware of the ramifications of this technique and how to handle them.

While instruction scheduling preserves the correctness of most programs, it may produce some non-intuitive and surprising results. **Figure 2** shows an example where instruction scheduling causes the compiler to emit incorrect code. To see this, compile the program as C code (/TC) in Release mode. You can set the target platform to either x86 or x64. Because you're going to examine the resulting assembly code, specify /FA so the compiler emits an assembly listing.

In this program, I want to measure the running time of the compute function. To do this, I usually wrap the call to the function by calls to a timing function such as clock. Then, by computing the difference in the values of the clock, I get an estimate of the time the function took to execute. Note that the purpose of this code is not to show you the best way to measure the performance of some piece of code, but to demonstrate the hazards of instruction scheduling.

Because this is C code and because the program is very simple, it's easy to understand the resulting assembly code. By looking at the assembly code and focusing on the call instructions, you'll note that the second call to the clock function precedes the call to

the compute function (it has been moved to the "target location"), making the measurement completely wrong.

Note that this reordering doesn't violate the minimum requirements imposed by the standard on conforming implementations, so it's legal.

But why would the compiler do that? The compiler thought that the second call to clock didn't depend on the call to compute (indeed, to the compiler, these functions don't affect each other at all). Also, after the first call to clock, it's likely the instruction cache contains some of the instructions of that function and the data cache contains some of the data required by these instructions. Calling compute might cause these instructions and data to be overwritten, so the compiler reordered the code accordingly.

The Visual C++ compiler doesn't offer a switch to turn instruction scheduling off while keeping all other optimizations on. Moreover, this problem can occur due to dynamic execution if the compute function was inlined. Depending on how the execution of the compute function is going and on how far a processor can look ahead, a 3OE processor might decide to begin executing the second call to clock before the compute function completes. Just as with the compiler, the majority of processors don't let you turn off dynamic execution. But to be fair, it's very unlikely that this problem will occur because of dynamic execution. How could you tell if it happened, anyway?

The Visual C++ compiler is actually very careful when performing this optimization. It's so careful that there are many things that prevent it from reordering an instruction (such as call instruction). I have noticed the following situations that caused the compiler to not move the clock function call to a particular location (the target location):

- Calling an imported function from any of the functions being called between the location of the function call and the target location. As this code shows, calling any imported function from the compute function causes the compiler to not move the second call to clock:

```
__declspec(noinline) int compute(){
    int x;
    scanf_s("%d", &x); /* Calling an imported function */
    return x;
}
```

- Calling an imported function between the call to compute and the second call to clock:

```
int main() {
    time_t t0 = clock();
    int result = compute();
    printf("%d", result); /* Calling an imported function */
    time_t t1 = clock();
    printf("Result (%d) computed in %lld.", result, t1 - t0);
    return 0;
}
```

- Accessing any global or static variable from any of the functions being called between the location of the function call and the target location. This holds whether the variable is being read or written. The following shows that accessing a global variable from the compute function causes the compiler to not move the second call to clock:

```
int x = 0;
__declspec(noinline) int compute(){
    return x;
}
```

- Marking t1 as volatile.

DATA QUALITY 101

MASTERING THE FUNDAMENTALS IS
THE KEY TO YOUR SUCCESS.

You can't turn Big Data into Big Value if it's messy. Garbage in, garbage out will always be a problem. Go back to the basics – cleaning all of your data is still the best, first step for success. Melissa Data provides the data quality tools and methods you need to correct, consolidate, and enrich contact data for improved data integration, business intelligence, and CRM initiatives. Always start with quality data – it's the easy way to get straight A's.

- **Verify addresses, phones & emails for over 240 countries**
- **Add geocodes and demographics for better insight**
- **Match duplicate records for a single view of the customer**
- **Identify change-of-address records before mailing**
- **On-premise and Cloud solutions**
- **Free trials with 120-day ROI guarantee**



ADDRESS
VERIFICATION



PHONE
VERIFICATION



NAME
VERIFICATION



EMAIL
VERIFICATION

BETTER DATA MEANS BETTER BUSINESS

MELISSA DATA®

www.MelissaData.com 1-800-MELISSA

There are other situations that prevent the compiler from reordering instructions. It's all about the C++ as-if rule, which says the compiler can transform a program that doesn't contain undefined operations anyway it likes as long as the observable behavior of the code is guaranteed to remain the same. Visual C++ not only adheres to this rule, but also is much more conservative to reduce the time it takes to compile the code. An imported function might cause side effects. Library I/O functions and accessing volatile variables cause side effects.

Volatile, Restrict and /favor

Qualifying a variable with the volatile keyword affects both register allocation and instruction reordering. First, the variable won't be allocated to any register. (Most instructions require some of their operands to be stored in registers, which means the variable will be loaded into a register, but only to execute some of the instructions that use that variable.) That is, reading or writing to the variable will always cause a memory access. Second, writing to a volatile variable has Release semantics, meaning that all memory accesses that occur syntactically before the write to that variable will happen before it. Third, reading from a volatile variable has Acquire semantics, meaning that all memory accesses that occur syntactically after the read from that variable will happen after it. But there's a catch: These reordering guarantees are offered only by specifying the /volatile:ms switch. In contrast, the /volatile:iso switch tells the compiler to adhere to the language standard, which doesn't offer any such guarantees through this keyword. For ARM, /volatile:iso takes effect by default. For other architectures, the default is /volatile:ms. Before C++11, the /volatile:ms switch was useful because the standard didn't offer anything for multi-threaded programs. However, starting with C11/C++11, the use of /volatile:ms makes your code not portable and is strongly discouraged and you should use atomics instead. It's worth noting that if your program works correctly under /volatile:iso, it will work correctly under /volatile:ms. More important, however, if it works correctly under /volatile:ms it may not work correctly under /volatile:iso because the former provides stronger guarantees than the latter.

The /volatile:ms switch implements Acquire and Release semantics. It's not enough to maintain these at compile time; the compiler might (depending on the target platform) emit extra instructions (such as mfence and xchg) to tell a 3OE processor to maintain these semantics while executing the code. Therefore, volatile variables degrade performance not only because the variables aren't cached in registers, but also because of the additional instructions that are being emitted.

The semantics of the volatile keyword according to the C# language specification are similar to those offered by the Visual C++ compiler with the /volatile:ms switch specified. There's a difference, however. The volatile keyword in C# implements Sequentially Consistent (SC) Acq/Rel semantics, while C/C++ volatile under /volatile:ms implements pure Acq/Rel semantics. Remember that C/C++ volatile under /volatile:iso has no Acq/Rel semantics. The details are beyond the scope of this article. In general, memory fences may prevent the compiler from performing many optimizations across them.

It's very important to understand that if the compiler didn't offer such guarantees in the first place, then any corresponding guarantees offered by the processor are automatically void.

The __restrict keyword (or restrict) also affects the effectiveness of both register allocation and instruction scheduling. However, in contrast to volatile, restrict can significantly enhance these optimizations. A pointer variable marked with this keyword in a scope indicates that there's no other variable that points to the same object, created outside the scope and used to modify it. This keyword also might enable the compiler to perform many optimizations on pointers, confidently including automatic vectorization and loop optimizations, and it reduces the generated code size. You can think of the restrict keyword as a top-secret, high-tech, anti-anti-optimization weapon. It deserves a whole article by itself; therefore, it will not be discussed here.

If a variable is marked with both volatile and __restrict, the volatile keyword will take precedence when making decisions regarding how to optimize the code. In fact, the compiler can totally ignore restrict, but must respect volatile.

The /favor switch might enable the compiler to perform instruction scheduling that's tuned to the specified architecture. It also can reduce the generated code size because the compiler might have the ability to not emit instructions that check whether a specific feature is supported by the processor. This in turn leads to improved instruction cache hit ratio and better performance. The default is /favor:blend, which results in code with good performance across x86 and x64 processors from Intel Corp. and AMD.

Wrapping Up

I discussed two important optimizations performed by the Visual C++ compiler: register allocation and instruction scheduling.

Register allocation is the most important optimization performed by the compiler because accessing a register is much faster than accessing even the cache. Instruction scheduling is also important. However, recent processors have outstanding dynamic execution capabilities, making instruction scheduling less significant than it was before. Still, the compiler can see all instructions of a function, no matter how big it is, while a processor can only see a limited number of instructions. Also, the out-of-order execution hardware is quite power hungry because it's always working as long as the core is working. Furthermore, x86 and x64 processors implement a memory model that's stronger than the C11/C++11 memory model and prevents certain instruction reordering that could improve performance. So, compiler-based instruction scheduling is still extremely important for power-limited devices.

Several keywords and compiler switches can affect performance either negatively or positively, so make sure to use them appropriately to ensure your code runs as fast as possible and produces correct results. There are still many more optimizations to talk about—stayed tuned! ■

HADI BRAIS is a Ph.D. scholar at the Indian Institute of Technology Delhi (IITD), researching compiler optimizations for the next-generation memory technology. He spends most of his time writing code in C/C++/C# and digging deep into the CLR and CRT. He blogs at hadibrais.wordpress.com. Reach him at hadi.b@live.com.

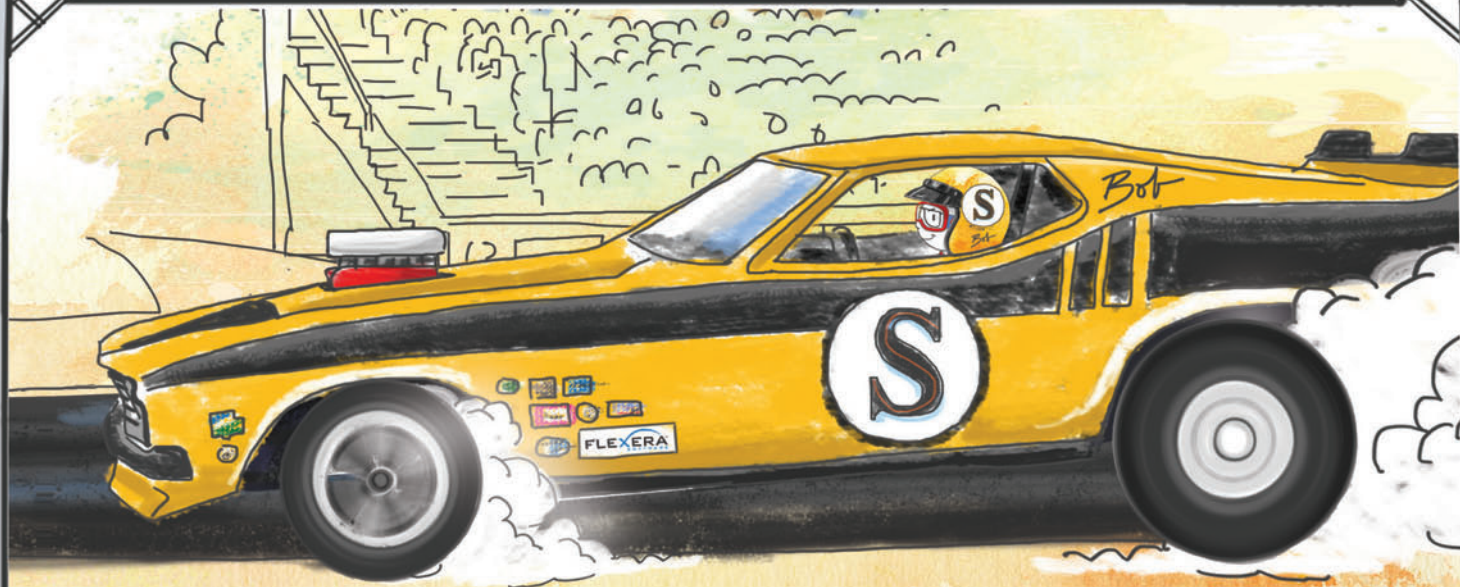
THANKS to the following technical expert for reviewing this article:
Jim Hogg (Microsoft Visual C++ Team)



FLEXERA SOFTWARE®

InstallShield®

GO FOR THE WIN



Take a Fresh Look at InstallShield

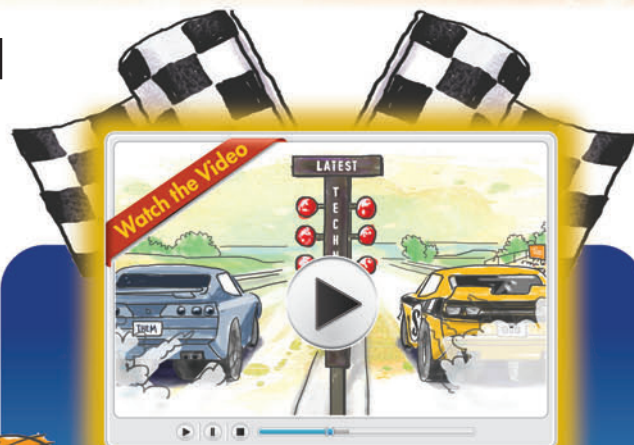
Inside Track – Simplify complex installs for web, cloud, virtual, PC and server

Wow the Crowd – Reliably deliver fast downloads and simple installations

Pole Position – Spend less time building installations and more time creating features that put you in the lead

Team Win – Distributed and agile teams can easily collaborate and save time

*Keep competitors in your rearview mirror.
Test drive InstallShield today.*



Buy Now and Register for a Chance to Win a Microsoft® Surface Pro 3*
flexerasoftware.com/GoForTheWin

* Restrictions apply. Details below.



FLEXERA
SOFTWARE

US/Canada: 1-800-809-5659
International: 1-847-466-4000

Ask to speak to an InstallShield Account Manager

* Contest eligibility requires completion of online form and subsequent purchase of InstallShield Professional, InstallShield Premier, InstallAnywhere Professional or InstallAnywhere Premier between April 6, 2015 and June 30, 2015. New licenses only. Previous purchases ineligible. One entry per customer/organization. Payment/PO must be received by June 30, 2015. Other restrictions may apply. Speak to your account manager for details. If purchased through a reseller, online form still must be completed and Reseller PO must be received by Flexera Software no later than June 30, 2015.



CODE HOME

No code trip would be complete without a stop where it all began, so we're heading to the idyllic **Microsoft Headquarters** in Redmond, WA, **August 10 - 14, 2015!**

Join us as we explore hot topics like Visual Studio, JavaScript/HTML5, ASP.NET, Database and Analytics, and more in over 70+ sessions and workshops. Rub elbows with Microsoft insiders, have lunch with Blue Badges, visit the company store, and experience code at the source.

JUST ADDED!

A FULL Track of Microsoft Sessions

10+ additional Microsoft-led sessions are being added to the agenda.

Go to vslive.com/redmond for more details!



Register NOW and Save \$400!



Scan the QR code to register or for more event details.

Use promo code **REDMAY2**

vslive.com/redmond

Design	Cloud Computing	Mobile Client	Database and Analytics	Web Development	Visual Studio / .NET	Windows Client
--------	-----------------	---------------	------------------------	-----------------	----------------------	----------------

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, August 10, 2015 (Separate entry fee required)				
8:00 AM	12:00 PM	M01 – Workshop: Building Universal Apps for Desktop, Store, and Phone – <i>Philip Japikse</i>		M02 – Workshop: UX Design Process Essentials – Steps and Techniques – <i>Billy Hollis</i>		M03 – Workshop: ALM and DevOps with the Microsoft Stack – <i>Brian Randell</i>
12:00 PM	2:30 PM	Lunch @ The Mixer – Visit the Microsoft Company Store & Visitor Center				
2:30 PM	6:00 PM	M01 – Workshop Continues		M02 – Workshop Continues		M03 – Workshop Continues
7:00 PM	9:00 PM	Dine-A-Round Dinner				

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, August 11, 2015				
8:30 AM	9:30 AM	Keynote: To Be Announced				
9:45 AM	11:00 AM	T01 – Azure 10-10: Top 10 Azure Announcements in “T-10” Months – <i>Vishwas Lele</i>	T02 – Hack Proofing Your Web Applications – <i>Adam Tuliper</i>	T03 – UX Design Principle Fundamentals for Non-Designers – <i>Billy Hollis</i>	T04 – A Lap Around Visual Studio 2015 – <i>Robert Green</i>	
11:15 AM	12:30 PM	T06 – Cloud or Not, 10 Reasons Why You Must Know “Web Sites” – <i>Vishwas Lele</i>	T07 – Take a Gulp, Make a Grunt, and Call Me Bower – <i>Adam Tuliper</i>	T08 – Designing and Building UX for Finding and Visualizing Data in XAML Applications – <i>Billy Hollis</i>	T09 – What’s New in ALM – <i>Brian Randell</i>	
12:30 PM	2:30 PM	Lunch – Visit Exhibitors				
2:30 PM	3:45 PM	T11 – Building Mobile Cross-Platform Apps with C# and Xamarin – <i>Nick Landry</i>	T12 – AngularJS 101 – <i>Deborah Kurata</i>	T13 – Windows, NUI and You – <i>Brian Randell</i>	T14 – Hosting ASP.NET Applications Cross Platform with Docker – <i>Adam Tuliper</i>	
3:45 PM	4:15 PM	Sponsored Break – Visit Exhibitors				
4:15 PM	5:30 PM	T16 – Building Mobile Cross-Platform Apps in C# with Azure Mobile Services – <i>Nick Landry</i>	T17 – AngularJS Forms and Validation – <i>Deborah Kurata</i>	T18 – Building Windows 10 LOB Apps – <i>Robert Green</i>	T19 – SOLID Design Patterns for Mere Mortals – <i>Philip Japikse</i>	
5:30 PM	7:00 PM	Microsoft Ask the Experts & Exhibitor Reception				

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, August 12, 2015				
8:00 AM	9:00 AM	Keynote: To Be Announced				
9:15 AM	10:30 AM	W01 – iOS Native Dev Talk – <i>Jon Flanders</i>	W02 – Implementing M-V-VM (Model-View-View Model) for WPF – <i>Philip Japikse</i>	W03 – Moving Web Apps to the Cloud – <i>Eric D. Boyd</i>	W04 – Enhancing Application Quality Using Visual Studio 2015 Premium Features – <i>Anthony Borton</i>	
10:45 AM	12:00 PM	W06 – Building Cross Platform UI with Xamarin.Forms – <i>Walt Ritscher</i>	W07 – To Be Announced		W08 – Solving Security and Compliance Challenges with Hybrid Clouds – <i>Eric D. Boyd</i>	W09 – Load Testing ASP.NET & WebAPI with Visual Studio – <i>Benjamin Day</i>
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch – Visit Exhibitors				
1:30 PM	2:45 PM	W11 – Swift for .NET Developers – <i>Jon Flanders</i>	W12 – Strike Up a Conversation with Cortana on Windows Phone – <i>Walt Ritscher</i>	W13 – To Be Announced		W14 – To Git or Not to Git for Enterprise Development – <i>Benjamin Day</i>
2:45 PM	3:15 PM	Sponsored Break – Exhibitor Raffle @ 2:55 pm (Must be present to win)				
3:15 PM	4:30 PM	W16 – Creating Applications Using Android Studio – <i>Kevin Ford</i>	W17 – Securing Angular Apps – <i>Brian Noyes</i>	W18 – Makers and the Internet of Things: Connecting Apps, Sensors & the Cloud – <i>Nick Landry</i>	W19 – Not Your Grandfather’s Build – A Look at How Build Has Changed in 2015 – <i>Anthony Borton</i>	
8:00 PM	10:00 PM	Lucky Strike Evening Out Party				

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, August 13, 2015				
8:00 AM	9:15 AM	TH01 – Using Multi-Device Hybrid Apps to Create Cordova Applications – <i>Kevin Ford</i>	TH02 – Build Data-Centric HTML5 Single Page Applications with Breeze – <i>Brian Noyes</i>	TH03 – Database Development with SQL Server Data Tools – <i>Leonard Label</i>	TH04 – Stop the Waste and Get Out of (Technical) Debt – <i>Richard Hundhausen</i>	
9:30 AM	10:45 AM	TH06 – Everything You Always Wanted to Know About REST (But Were Afraid to Ask) – <i>Jon Flanders</i>	TH07 – Build Real-Time Websites and Apps with SignalR – <i>Rachel Appel</i>	TH08 – Programming the T-SQL Enhancements in SQL Server 2012 – <i>Leonard Label</i>	TH09 – What’s New in C# 6.0 – <i>Jason Bock</i>	
11:00 AM	12:15 PM	TH11 – To Be Announced		TH12 – Knocking it Out of the Park with Knockout.JS – <i>Miguel Castro</i>	TH13 – Power BI 2.0: Analytics in the Cloud and in Excel – <i>Andrew Brust</i>	TH14 – Async and Await Best Practices – <i>Mark Rosenberg</i>
12:15 PM	2:15 PM	Lunch @ The Mixer – Visit the Microsoft Company Store & Visitor Center				
2:15 PM	3:30 PM	TH16 – Extending XAML to Overcome Pretty Much any Limitation – <i>Miguel Castro</i>	TH17 – ASP.NET MVC: All Your Tests Are Belong To Us – <i>Rachel Appel</i>	TH18 – Busy Developer’s Guide to NoSQL – <i>Ted Neward</i>	TH19 – Microsoft’s .NET is Now Open Source and Cross-Platform. Why it Matters. – <i>Mark Rosenberg</i>	
3:45 PM	5:00 PM	TH21 – Agile Database Development – <i>Richard Hundhausen</i>	TH22 – Busy JavaScript Developer’s Guide to ECMAScript 6 – <i>Ted Neward</i>	TH23 – Big Data and Hadoop with Azure HDInsight – <i>Andrew Brust</i>	TH24 – Managing the .NET Compiler – <i>Jason Bock</i>	

START TIME	END TIME	Visual Studio Live! Post-Conference Workshops: Friday, August 14, 2015 (Separate entry fee required)				
8:00 AM	12:00 PM	F01 – Workshop: SQL Server for Developers – <i>Andrew Brust and Leonard Label</i>			F02 – Workshop: Service Oriented Technologies: Designing, Developing, & Implementing WCF and the Web API – <i>Miguel Castro</i>	
12:00 PM	1:00 PM	Lunch				
1:00 PM	5:00 PM	F01 – Workshop Continues			F02 – Workshop Continues	

Speakers and sessions subject to change

2D Game Engines for the Web

Michael Oneppo

Imagine there's a developer who wants to write a game. This developer looks around to see what tools are available, and is disappointed by the options. So, this developer then decides to create a custom engine that will fulfill all the needs for the current game concept. It will also fulfill all future game development requirements anyone will ever possibly have. And, so, the developer never actually writes a game.

It's a tragedy that happens frequently. Fortunately, this story has spawned dozens of game development engines for all possible platforms. These encompass a wide range of methodologies, tools and licensing models. For example, [JavaScripting.com](#) currently lists 14 dedicated open source game libraries. That doesn't even include physics engines, audio libraries and specialized input systems for games. With that kind of selection, there's bound to be a game engine that will work well for any game project you have in mind.

This article will walk through three popular open source 2D game engines for the Web: Crafty, Pixi and Phaser. To compare and contrast these libraries, I've ported the Ping game from my

first article (msdn.microsoft.com/magazine/dn913185) to each one to see how they feel. Keep in mind by restricting things to 2D, I'm leaving out a number of 3D game engines for the Web. I'll cover those in the future. For now, I'll focus on 2D and the opportunities there.

Crafty

Crafty (craftyjs.com) is meant primarily for tile-based games, although it works well for a wide variety of 2D games, including my Ping game. The foundation of Crafty is an object model/dependency system it calls components.

Components are like classes you define with specific attributes and actions. Each instance of a component is called an entity. Components are meant to be heavily mixed to make complex, feature-rich entities. For example, if you want to make a sprite sheet in Crafty, use the `sprite` function to generate components that represent your sprites. Here's how I define the sprite types from an image:

```
Crafty.sprite("sprites.png", {
  PlayerSprite: [0, 128, 128, 128],
  OpponentSprite: [0, 0, 128, 128],
  BallSprite: [128, 128, 64, 64],
  UpSprite: [128, 0, 64, 64],
  DownSprite: [128, 64, 64, 64],
  LeftSprite: [192, 0, 64, 64],
  RightSprite: [192, 64, 64, 64]});
```

Now, if you want to make a ball entity that uses the image of the ball on the sprite sheet, include the `BallSprite` component when you create the entity, using the `e` function, as follows:

```
Crafty.e("Ball, BallSprite");
```

This won't draw it to the screen yet. You have to tell Crafty you want this entity to live in 2D (so it will have an "x" and "y" position), and you want it to draw using DOM elements:

```
Crafty.e("Ball, 2D, DOM, BallSprite");
```

This article discusses:

- Three popular 2D game engines
- How to link actions to game sprites
- Selecting the environment that suits your needs

Technologies discussed:

Visual Studio Pro 2013

Code download available at:

msdn.microsoft.com/magazine/msdnmag0515

Spreadsheets Made Easy.



Fastest Calculations

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



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.



Windows
Forms



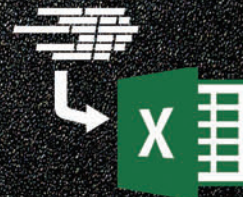
Silverlight



WPF

Powerful Controls

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



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

If you want to draw your elements in a canvas, it's as simple as replacing the Document Object Model (DOM) component with Canvas.

Entities act like independent objects on your screen by reacting to events. Like many JavaScript libraries, Crafty entities have a bind function for reacting to events. It's important to remember these are Crafty-specific events. For example, if you want an entity to do something in every frame, have it react to the EnterFrame event. This is exactly what's needed to move the ball around based on who's holding it and the appropriate physics, if necessary. **Figure 1** shows ball entity initialization with an EnterFrame event function that covers the motion of the ball.

Crafty is meant primarily
for tile-based games, although it
works well for a wide variety
of 2D games.

If you look closely, you'll see "this" used frequently to refer to entity properties. The built-in Crafty "this.x" and "this.y" define the ball position. When you create the ball with the 2D component, this functionality is added. The statement, "this.velocity," is custom to my code. You can see it defined as a property using the attr function. The same is true with this.owner, which I use to figure out if either player is holding the ball.

Crafty has several built-in components, but I don't use them a huge amount in the Ping example. Here are just a few:

- **Gravity:** Crafty has a simple gravity engine that automatically pulls an entity downward. You can define any number of element types to stop its motion.

Figure 1 The Ball Entity Definition

```
Crafty.e("Ball, 2D, DOM, BallSprite, Collision")
.attr({ x: width/2, y: height/2, velocity: [0,0], owner: 'User' })
.bind('EnterFrame', function () {
  // If there's an owner, have the ball follow the owner.
  // Nudge the ball to the left or right of the player
  // so it doesn't overlap.
  if (this.owner) {
    var owner = Crafty(this.owner);
    switch(this.owner) {
      case 'User':
        this.x = owner.x + 128;
        break;
      case 'Opponent':
        this.x = owner.x - 64;
    }
    this.y = owner.y + 32;
    return;
  }

  // Bounce the ball off the ceiling or floor.
  if (this.y <= 0 || this.y >= (height - 64))
    this.velocity[1] = -this.velocity[1];

  // Move the ball based on its velocity, which is defined in
  // pixels per millisecond.
  var millis = 1000 / Crafty.timer.FPS();
  this.x += this.velocity[0] * millis;
  this.y += this.velocity[1] * millis;
})
```

- **TwoWay/FourWay/MultiWay Motion:** With these components, you get a variety of arcade-style motion input methods. TwoWay is for platformers, with left, right and jump options bindable to any keys you want. FourWay allows top-down motion in the cardinal directions. MultiWay allows custom-defined inputs, each with an arbitrary direction.
- **Particles:** Crafty includes a fast particle system, which you can only use with Canvas drawing. As soft, round, single-color images, the particles are particularly suited for smoke, fire and explosions.

The code for Ping, implemented in Crafty, is available in the code download accompanying this article. Take a look to get a full picture of how I ported things, particularly the AI for the opponent.

Pixi.js

Pixi.js (pixijs.com) is the bare-bones, close-to-the-metal 2D Web renderer. Pixi.js can forego the 2D canvas and dive straight into WebGL, which gives it a significant performance gain. A Pixi.js game has two key pieces: a stage that describes your game layout in a scene graph, and a renderer, which actually draws the elements in the stage to the screen using canvas or WebGL. By default, Pixi.js will use WebGL if it's available:

```
$(document).ready(function() {
  // The stage to play your game.
  stage = new PIXI.Stage(0xFFFFFFFF);
  lastUpdate = 0;

  // autoDetectRenderer() finds the fastest renderer you've got.
  renderer = PIXI.autoDetectRenderer(innerWidth, innerHeight);
  document.body.appendChild(renderer.view);
});
```

Pixi.js has a scene graph and transformations, but it's important to note you need to render the scene yourself. So, for my Ping game, I keep the rendering loop going using requestAnimationFrame, as follows:

```
function update(time) {
  var t = time - lastUpdate;
  lastUpdate = time;
  ball.update(t);
  ai.update();

  // New: You actually have to render the scene
  renderer.render(stage);
  requestAnimationFrame(update);
}
```

Unlike Crafty, Pixi.js doesn't dictate much about how to structure your game, so I can pretty much use the same code from the original Ping game, with slight modifications.

The big difference you'll see is that Pixi.js either lets you load sprites as images from files one by one, or as a tiled image using a special tool called TexturePacker. This tool will automatically merge a group of individual images into a single, optimized file with an accompanying JSON file that describes the image locations. I used TexturePacker to make the sprite sheet. **Figure 2** shows the updated ready function, loading the sprite images.

The loader is asynchronous, as it requires a function to call for when it's done loading. The remaining game initializations, like creating the player, opponent and ball, are in the new startGame function. To use these newly initialized sprites, use the PIXI.Sprite.fromFrame method, which takes an index into the array of sprites in the JSON file.

facebook



Microsoft SharePoint 2010



Linked in



twitter

SEE THE WORLD AS A DATABASE

amazon web services

bing

amazon web services

Microsoft Excel 2010

Microsoft SQL Server

OData Open Data Protocol



ADO.NET ▪ JDBC ▪ ODBC ▪ SQL SSIS ▪ ODATA
MYSQL ▪ EXCEL ▪ POWERSHELL

Microsoft SQL Server

Linked in

SAP

OData Open Data Protocol

Salesforce



facebook



Microsoft SharePoint 2010

amazon web services



Microsoft Visual Studio Java ODBC Microsoft SQL Server Microsoft Excel Microsoft BizTalk MySQL OData

Work With Relational Data, Not Complex APIs or Services

Whether you are a developer using ADO.NET, JDBC, OData, or MySQL, or a systems integrator working with SQL Server or Biztalk, or even an information worker familiar with ODBC or Excel – our products give you bi-directional access to live data through easy-to-use technologies that you are already familiar with. If you can connect to a database, then you will already know how to connect to Salesforce, SAP, SharePoint, Dynamics CRM, Google Apps, QuickBooks, and much more!



Give RSSBus a try today and see what mean:

visit us online at www.rssbus.com to learn more or download a free trial.

rssbus

INTEGRATION YOUR WAY

One last difference Pixi.js provides is a touch and mouse input system. To make any sprite input-ready, I set the interactive property to “true” and directly add some event handlers to the sprite. For instance, for the up/down buttons, I bound events to move the player up and down, as shown in **Figure 3**.

It’s important to note I had to manually add each sprite to the scene to make it visible. You can check out my version of Ping built in Pixi.js in the code download accompanying this article.

Phaser

Phaser (phaser.io) is a full-featured game engine that uses Pixi under the hood to perform all rendering tasks. Phaser has a long list of features, including frame-based sprite animation, music and audio, a game state system and physics with three different contributing libraries.

Because I already ported Ping onto Pixi.js, porting to Phaser was straightforward. Phaser streamlines object creation based on Pixi.js:

```
ball = game.add.sprite(x, y, 'sprites');
```

While Phaser can be more concise, there are places where things are more complex. For instance, the last parameter in the previous code, “sprite” refers to an image loaded at startup:

```
game.load.image('sprites', 'sprites.png');
```

Phaser has a different sprite sheet system for images from Pixi.js. You can break an image into tiles. To use it, call something like this:

```
game.load.spritesheet('sprites', 'mySprites.png',  
    spriteWidth, spriteHeight, totalSprites);
```

For my example, I don’t actually use this function. The code I just showed you assumes all sprites are the same size, but the Ping sprite sheet has sprites of 64 pixels and 128 pixels. As such, I had to crop each sprite image manually. To crop down the image for the ball, I set a cropping rectangle on the sprite:

```
ball.cropRect = new Phaser.Rectangle(0, 64, 64, 64);  
ball.updateCrop();
```

Figure 2 Loading a Sprite Sheet in Pixi.js

```
$(document).ready(function() {  
    // Create a new instance of a Pixi stage.  
    stage = new PIXI.Stage(0xFFFFFFFF);  
    lastUpdate = 0;  
  
    // Create a renderer instance.  
    renderer = PIXI.autoDetectRenderer(innerWidth, innerHeight);  
    document.body.appendChild(renderer.view);  
  
    var images = ["sprites.json"];  
    var loader = new PIXI.AssetLoader(images);  
    loader.onComplete = startGame;  
    loader.load();  
});
```

Figure 3 Bind Events to Move Game Player Up and Down

```
up = new PIXI.Sprite.fromFrame(3);  
up.interactive = true;  
stage.addChild(up);  
up.touchstart = up.mousedown = function() {  
    player.move(-distance);  
}  
  
down = new PIXI.Sprite.fromFrame(4);  
down.interactive = true;  
stage.addChild(down);  
down.touchstart = down.mousedown = function() {  
    player.move(distance);  
}
```

Hopefully this shows some of the flexibility of Phaser. That flexibility manifests itself in other ways, as well. You can make your game event-based, or check for events and respond to them sequentially in an update function. For instance, to handle a key input, you can respond to an event:

```
game.input.keyboard.onDownCallback = function(key) {  
    console.log(key + " was pressed.");  
}
```

or check if the key is pressed in your update loop:

```
function update() {  
    if (game.input.keyboard.isDown('a'.charCodeAt(0)) {  
        console.log("'A' key was pressed");  
    }  
}
```

This goes for a lot of events in Phaser, using sequential or asynchronous coding depending on your preference or need. On occasion, Phaser provides only the latter form of event handling. A good example is the Arcade physics system, where you must explicitly make a function call for every update to check for collisions between objects:

```
game.physics.arcade.collide(player, ball, function() {  
    ball.owner = player;  
});
```

This code determines if the ball has come in contact with the player sprite, and gives control of the ball to the player when that happens. Check out my implementation of Ping on Phaser with the code download for this article.

Wrapping Up

If you’re looking for a dedicated 2D JavaScript game engine, there are a lot of options to suit your needs. Here are two important factors to keep in mind when picking a framework:

Use Only the Features You Need: Don’t be dazzled by lots of features, all-encompassing monolithic frameworks, and other bells and whistles. Unless the bells and whistles address specific aspects of the game you want to make, they’ll probably get in the way. Also consider how the features are componentized. Will the whole system still work if you remove the physics system? Does the framework rely on the physics system to provide other functionality?

Understand How Each Engine Works: Try to understand how you can customize or extend the framework. If you’re making something slightly out of the ordinary, it’s likely you’ll need to write custom code that relies heavily on the framework you’ve chosen. At that point, you can either write a new, independent feature or extend the framework functionality. If you extend the framework itself instead of adding to it, you’ll end up with more maintainable code.

The frameworks covered here address many of these issues, so they’re definitely worth checking out. There are dozens of engines, libraries and frameworks out there, so do some research before embarking on your next game development adventure. ■

MICHAEL ONEPPO is a creative technologist and former program manager at Microsoft on the Direct3D team. His recent endeavors include working as CTO at the technology nonprofit Library for All and exploring a master’s degree at the NYU Interactive Telecommunications Program.

THANKS to the following Microsoft technical expert for reviewing this article:
Justin Garrett

We know how application development can be.



INTRODUCING
Atalsoft MobileImage®
MOBILE CAPTURE SDK
Try it free for 30 days with full support
ATALASOFT.COM

Let DotImage take some of the bite out of your challenges.

Atalsoft DotImage®

Connecting the dots is a no-brainer. DotImage image-enables your .NET-based web application faster, more cost effectively, and less painfully than if done on your own. This proven SDK is versatile, with options including OCR capabilities, WingScan compatibility, and support for a range of formats. Coupled with dedicated assistance from our highly knowledgeable and skilled engineers, DotImage helps your business connect with powerful information hidden inside your documents, making the big picture much easier to see.



THE CODE THAT NEVER SLEEPS

Visual Studio Live! is hitting the open road on the ultimate code trip to help you navigate the .NET Highway. The next stop? NYC, and we're geared up to be back in the big apple for the first time in 2012.

From September 28 - October 1, Visual Studio Live! is bringing its unique brand of practical, unbiased, Developer training to Brooklyn, offering four days of sessions, workshops and networking events - all designed to help you avoid road blocks and cruise through your projects with ease.





NAVIGATE THE .NET HIGHWAY

DEVELOPMENT TOPICS INCLUDE:

- VISUAL STUDIO/.NET
- JAVASCRIPT/HTML5 CLIENT
- ASP.NET
- MOBILE CLIENT
- WINDOWS CLIENT
- DATABASE AND ANALYTICS
- CLOUD COMPUTING
- SHAREPOINT/OFFICE



Register Now and Save \$300!



USE PROMO CODE NYMAY2
SCAN THE QR CODE TO REGISTER
OR FOR MORE EVENT DETAILS.

VSLIVE.COM/NEWYORK

Join us on the Ultimate Code Trip in 2015!

LAS VEGAS Code Trip	AUSTIN Code Trip	SAN FRANCISCO Code Trip	REDMOND Code Trip	NEW YORK Code Trip	ORLANDO Code Trip
March 16-20	June 1-4	June 15-18	August 10-14	Sept. 28- Oct. 1	Nov. 16-20

FOLLOW US



twitter.com/vslive - @VSLive



facebook.com - Search "VSLive"



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



End-to-End Testing in Modern Web Sites and Apps

Software is more complex than ever before. Whether it's an app for Windows, iOS, Web, Android, an Internet of Things (IoT) device or a smartwatch—that software is doing a lot. Consequently, the software must be accurate. It must work according to specs. So it's up to developers to test their software.

Without testing, it's difficult to verify whether the code does what it should. It's also harder to isolate code to fix bugs. While it's important to test, it's also important to test for the right reasons, not just for the sake of testing or bragging rights about 100 percent coverage. Small smart tests always win over meaningless tests that cover the entire code base. This article will cover how to unit test server- and client-side code, as well as how to automate UI testing with coded UI tests (CUITs). These three basic areas of testing will test the major aspects of your software's functionality.

Unit by Unit

Unit tests help reduce bugs and provide documentation for the code they're testing. Regression, user acceptance, UI automation and other quality assurance measures are also important. Budgets are often tight and teams are small, so many times you have to choose one over another. It's up to you to decide how much quality assurance goes into your app. At the very minimum, unit testing and UI tests will help keep quality high.

Without testing, it's difficult to verify whether the code does what it should. It's also harder to isolate code to fix bugs.

There are a number of open source and commercial unit-testing software packages available. Many of the popular testing suites publish a NuGet package, so you can easily download them directly into Visual Studio. A few of the popular C# testing frameworks are xUnit, NUnit, MSTest and TestDriven.NET. For JavaScript testing, there's qUnit, JSUnit, and YUI Test.

While unit testing is often the default type of test, there are other ways to test your software. Those include UI automation or CUITs, user acceptance testing, regression testing, integration testing and all types of specialty tests. Wikipedia lists dozens of ways to test software at bit.ly/14066q8.

Good unit tests only test one function per test. The single responsibility principle (SRP) is one way to ensure your code is modular. SRP means the code focuses on one thing and does that thing well, whether it's implementation code or the test itself. Unit tests shouldn't directly access databases, files or other resources. They should rely on mock objects that simulate those resources and, just as with any other code, unit tests should be small.

Another way to write good tests is by adhering to test-driven development (TDD) principles. TDD is a way to develop code around a suite of tests. The developer creates tests that fail, then writes proper code to conform to the test, then refactors. I call it, "Red, Green, Refactor." This technique helps you structure and write code consistently. And consistency is a virtue in software development. Although TDD is outside the scope of this article, look into it and try it out with your team.

Figure 1 Testing a Sample of xUnit Code

```
// Unit tests that test customer-related features
public class CustomerTests
{
    [Theory]
    [InlineData(990)]
    [InlineData(10010)]
    [InlineData(50010)]
    [InlineData(100010)]
    public void VerifyCustomerStatus(double TotalSpent) {
        // Arrange
        var status = new Status();
        var testStatus=OrderForm.Models.Status.StatusLevel.None;

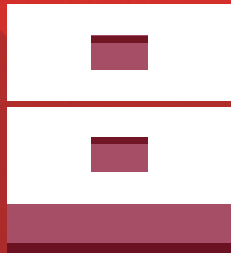
        // Act
        var currentStatus = status.GetCurrentStatus(TotalSpent);

        // Assert
        Assert.True(testStatus <= currentStatus);
    }
}
// Logic that sets the customer status
public class Status
{
    public StatusLevel GetCurrentStatus(decimal ForAmount)
    {
        var level = StatusLevel.None;
        var amt = Convert.ToInt32(ForAmount);
        if (amt > 1000 && amt <= 5000) { level = StatusLevel.Silver; }
        else if (amt > 5000 && amt <= 10000) { level = StatusLevel.Gold; }
        else if (amt > 10000) { level = StatusLevel.Platinum; }

        return level;
    }
    public enum StatusLevel
    {
        None = 0,
        Silver = 1,
        Gold = 2,
        Platinum = 3
    }
}
```

Jet **BRAINS**

R#



ReSharper

jetbrains.com/msdn

The legendary extension
to Visual Studio.*



* WARNING! PROLONGED USE MAY CAUSE ADDICTION.

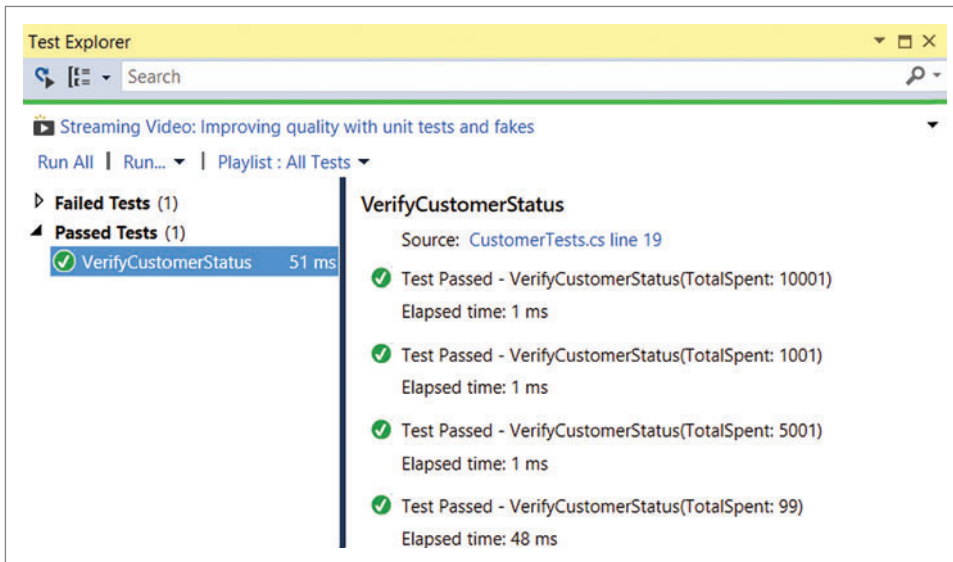


Figure 2 The VerifyCustomerStatus Test Method with Four Data Points in Test Explorer

This article will cover unit testing with xUnit and qUnit. I'll also look at UI automation testing with CUITs. This way you can become familiar with the most important types of tests you'll need.

Test Server-Side C# with xUnit

You'll be pleased to find out how easy it is to integrate a unit-testing package into Visual Studio. Many frameworks are available as a NuGet package. For example, if you'd like to use xUnit, just launch the NuGet Package Manager, search for "xUnit," then click to install both the core package and its test runner package. You can use xUnit to test ASP.NET Web sites, Windows Forms, Windows Presentation Foundation (WPF), Windows Store and Windows Phone apps, plus any language that compiles to the Microsoft Intermediate Language, or MSIL. You can even unit test F# using xUnit—that's all the Microsoft development technologies.

To use xUnit, add a Class Library project to your solution. For simplicity's sake, you might want to keep to common naming conventions and use something like ProjectName.Tests, where "ProjectName" is the name of the project you're testing. In your test, make sure you add a reference to that project. Then name and add a class file to the test, and a class for testing something.

Figure 3 The qUnit HTML Test Harness

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>qUnit Unit Testing</title>
  <link href="Content/qunit.css" rel="stylesheet" />
</head>
<body>
  <h1 id="qunit-header">qUnit Test Suite</h1>
  <h2 id="qunit-banner"></h2>
  <div id="qunit-testrunner-toolbar"></div>
  <h2 id="qunit-userAgent"></h2>
  <ol id="qunit-tests"></ol>
  <script src="Scripts/qunit.js"></script>
  <script src="Scripts/logic.js"></script>
  <script src="Scripts/tests.js"></script>
</body>
</html>
```

Figure 1 shows a sample of an xUnit test class with one test, along with the code it tests. The code in Figure 1 determines a customer's status based on how much they've spent. The tests in Figure 1 ensure the code is correct.

Within unit test methods, you'll find the popular arrange, act, assert (or AAA) pattern, which is also shown in Figure 1. First arrange the test conditions, then act on those conditions and, last, assert the output.

You can use AAA on anything, but if you want quality software, you must have meaningful tests. Testing frameworks tend to come with a fleet of assertions, such as Equal, NotEqual, Same, NotSame,

Contains, DoesNotContain, InRange, IsNotEmpty and several others. Of course, the exact naming or methods won't be the same, but each framework gives you several from which to choose.

Good unit tests only test one thing per test.

You may have noticed the [Theory] and [InlineData] attributes decorating the test class, instead of the popular [Test] attribute. When test methods have no parameters, in xUnit you can use the [Fact] attribute to mark the method as a test method. These data annotations let the test accept the TotalSpent parameter you see in the test method's signature in Figure 1. One test will run for each [InlineData] attribute on a method, and display the results of each test in Visual Studio Text Explorer, as shown in Figure 2.

You can use a context menu on tests in Test Explorer at any time to reveal the options to run, debug, analyze, profile and so on. Any time you change code or tests, you must compile and run the tests.

Figure 4 QUnit Tests the getCurrentStatus Function and the Logic Itself in JavaScript

```
// Tests to test customer statuses
test('VerifyCustomerStatus', function () {
  var silver = getCurrentStatus(1001);
  equals(silver, "Silver", "Pass");

  var gold = getCurrentStatus(5001);
  equals(gold, "Gold", "Pass");

  var platinum = getCurrentStatus(10001);
  equals(platinum, "Platinum", "Pass");
});
// Logic to set customer statuses
function getCurrentStatus(amt) {
  if (amt > 1000 && amt <= 5000) { return "Silver"; }
  else if (amt > 5000 && amt <= 10000) { return "Gold"; }
  else if (amt > 10000) { return "Platinum"; }
  return "None";
}
```


Debug Like a Boss

Enhance Your C# Debugging Experience

Simplify

```
2 250
"X@"
if (email.Length < MaxLength &&
    email.Length > 0 &&
    IsValid(email))
{
    SendThankYouLetter(email);
}
```

Make sense of complex expressions and predict the next result

Reveal

Index	FullName	Id
[0]	"Louise Midgett"	1
[1]	"Shirley Lewis"	2
[2]	"Wayne Campbell"	3
[3]	"Ben Cribbs"	4
[4]	"Roberta Deputy"	5

Count = 30

Search:

Visualize objects and focus on data that actually matters

Search

Find a needle in a haystack of data

Compare

View differences between objects

Trace

Log, view and filter debug messages

Exception Trail

Easily navigate inner exceptions

Quick Actions

Automate your everyday debugging

Show All Instances

Track down an object in memory



Supports Visual Studio 2010, 2012, 2013



Get an exclusive DISCOUNT



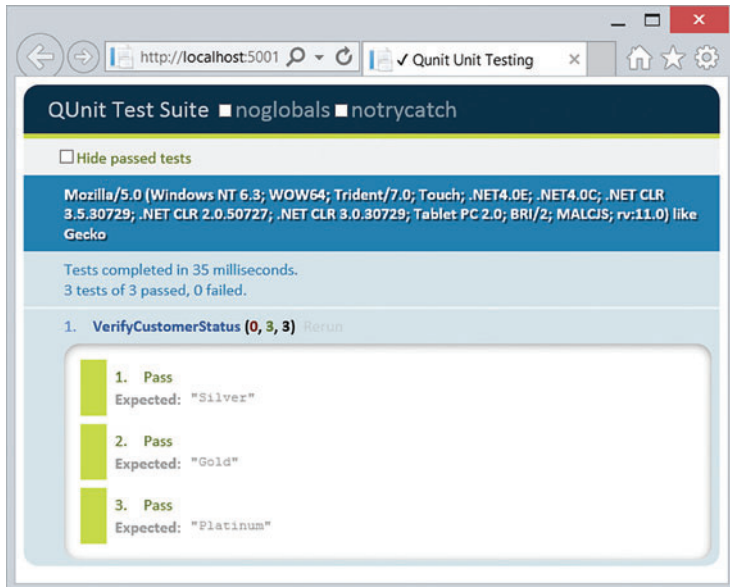


Figure 5 Results of qUnit Tests in the Browser

You can set the option to run tests automatically on each build by going to the Visual Studio Test menu, selecting Test Settings, then selecting Run Tests After Build.

You won't get far with unit testing before you bump into the need for mock objects. Generally, when you attempt to test a database, you don't want to modify the actual live data. Therefore, you can use mocking as a way to simulate the actual database, Web service or business object. Mocking is outside the scope of this article, so when you get to that point, search the Web for Moq, Telerik JustMock or Rhino Mocks. All are popular mocking software titles with full documentation and support.

Test Client-Side JavaScript with qUnit

QUnit is a client-side JavaScript testing framework. It's lightweight, simple and easy to use. You can use qUnit to test any JavaScript or jQuery. Because you're dealing with client-side JavaScript, there may be activity in the UI happening you don't need to test. You can skip aspects such as link hovers or changes to aesthetic UI. You can also use UI automation testing to cover UI testing. Your business

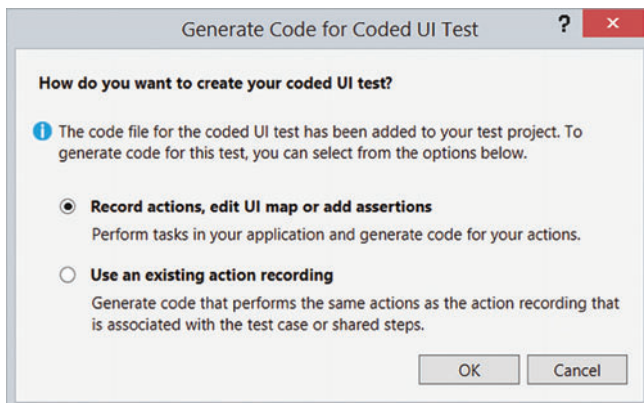


Figure 6 The Dialog Visual Studio Displays to Record or Edit Coded UI Tests

rules, API code and server-side validation must all have good coverage. Any JavaScript that performs logic and validation needs testing, as well.

Unlike the compiled language tests that integrate directly with Visual Studio, you'll have to build your qUnit test harness. It's fairly easy, though, and takes roughly 17 lines of code, which you can copy from Figure 3.

Figure 3 shows references to qunit.css, then qunit.js, logic.js and tests.js. The order is important. You must load qunit.js before your logic, and your logic before the tests. You can name logic.js whatever you want, as it's your own code. There may be multiple .js files between qunit.js and the tests. The HTML structure of the test harness page contains a few heading tags, as well as an ordered list that displays the individual test results. All the styles for these are in the qunit.css file. As a best practice, put only the JavaScript references at the end of the Web page immediately before the closing </body> tag, as shown in Figure 3.

To test with qUnit, you can place tests in a .js file and name it tests.js. In here, there's a call to the qUnit test function, which

accepts the string name of the function to call, and an anonymous function that actually performs the test. The name passed into the test function shows up as the name of the test in the test runner, as shown in Figure 4. Instead of passing in parameters to test, use the dynamic nature of JavaScript to call the code in question multiple times using multiple asserts. Figure 5 shows the output of multiple asserts in the VerifyCustomerStatus test function.

When you need to automate testing for native UIs, turn to CUITs.

If you're writing Windows Store or Windows Phone apps using the Windows Library for JavaScript (WinJS), fear not. There's a qUnitMetro you can download at bit.ly/1F2Rs07. This package lets qUnit run in the app store container just as it would in a Web page. The coding and the way you write tests are identical to qUnit, because it is indeed qUnit.

Test the UI with CUITs

When you need to automate testing for native UIs, turn to CUITs. CUITs seem to be a little-known testing feature in Visual Studio. You can create CUITs in Visual Studio for Web, Windows Store and Windows Phone apps. Visual Studio CUITs can test any UI on Windows. You don't even need the source code to test a UI. You can configure the tests to target any running program.

There are Visual Studio templates for your basic program UIs—Web, Windows Store and Windows Phone. When you create a project from one of these templates, Visual Studio will



Figure 7 The Tool to Record and Edit Assertions

AWSInsider.net

Your independent resource for news, how-tos, tips and more on the Amazon Web Services cloud platform.



launch you directly into the first dialog windows, shown in **Figure 6**, asking if you want to record new tests or edit existing ones. Select “Record actions, edit UI map or add assertions,” and a tool window will appear (see **Figure 7**). This same dialog and tool window will appear every time you add a new CUIT file to a project (File | New Item). At this point, you can click record, and then start interacting with your Web page or the app UI.

When you're done, click the Add and Generate button on the bottom of the window (see **Figure 8**), add detailed and descriptive metadata, and let Visual Studio generate the code.

When you create a Coded UI project and add a Coded UI Test to it, Visual Studio generates a fair amount of code. However, Visual Studio is quite smart about how it generates the test code. The output is clean, especially for generated code. Visual Studio generates the code by recording your actions while using a particular program. If you click record on the toolbox in **Figure 7**, the recorder tracks what you do to generate code that performs the same action. You're free to write this code, but it's easier to have it generated.

Once you're done with the Coded UI Test Builder, Visual Studio creates a few files, the first being the Coded UI test. This is a .cs file and contains a class with methods that perform the same steps a user would take to use a UI. The other files are map files: UIMap.uitest, UIMap.cs and UIMap.designer.cs. Unlike traditional Visual Studio

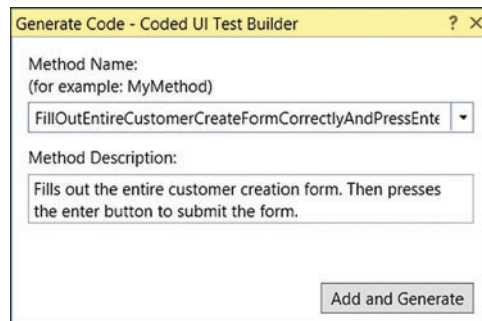


Figure 8 Dialogs That Create Coded UI tests

designers that are notorious for generating too much code, this one behaves much better. It's easy to modify tests without the designer getting in your way. At any time, you can modify the designer by right-clicking on the .uitest file and selecting Edit with Coded UI Test Builder from the options. This launches the toolbox you saw in **Figure 7**. Again, you're free to write your own tests manually. Before deciding, take a look at some of the generated code that performs these four tests:

1. It tests creating a new customer correctly
2. It tests creating a new customer without entering a name
3. It tests creating a new customer without entering an address
4. It tests creating a new customer by entering an invalid postal code format

You can find the four tests in the designer file, and code in the Coded UI test class calls them in succession, as you can see in **Figure 9**.

It's easy to modify tests
without the designer
getting in your way.

Coded UI tests are just like any other test. They're for interacting with the UI just as a user would. Fortunately, the Coded UI tool is great for recording your actions and generating code that does the same thing. At any time, you can discard and regenerate the tests, or modify the code without worry. Also like unit tests, you can run the Coded UI tests from Visual Studio Test Explorer. They'll show up next to the unit tests.

Improve the Quality of Your Software

There are real benefits from the various forms of testing. It's important to test even small Windows Store or Windows Phone apps, especially if you're selling them in an app store. Tested apps are apps that work reliably—they have documentation and an audit trail.

You can unit test with libraries like xUnit and qUnit, and use automation testing for Web or native apps with CUITs. That's not to mention other types of important testing such as user acceptance testing, integration testing, systems testing and so on. Adding these once you have a solid set of unit and UI tests will significantly improve the quality of your software. ■

RACHEL APPEL is a consultant, author, mentor and former Microsoft employee with more than 20 years of experience in the IT industry. She speaks at top industry conferences such as Visual Studio Live!, DevConnections, Mix and more. Her expertise lies within developing solutions that align business and technology focusing on the Microsoft dev stack and open Web. For more about Appel, visit her Web site at rachelappell.com.

THANKS to the following technical expert for reviewing this article:
Oren Novotny

Figure 9 Partial Contents of the Coded UI Test and Map Designer Files

```
// In the Coded UI Test C# file
[CodedUITest]
public class CodedUITest1
{
    [TestMethod]
    public void CodedUITestMethod1()
    {
        this.UIMap.CreateNewCustomerCorrectly();
        this.UIMap.CreateNewCustomerWithoutName();
        this.UIMap.CreateNewCustomerWithoutAddress();
        this.UIMap.CreateNewCustomerWithoutValidPostalCode();
    }
    // Some other framework code for Coded UI
}
// In the Coded UI UIMap.Designer.cs file
public void CreateNewCustomerWithoutName()
{
    #region Variable Declarations
    // Click 'Create New' link
    Mouse.Click(uiCreateNewHyperlink, new Point(51, 8));
    // Type '' in 'Name' text box
    uiNameEdit.Text = this.CreateNewCustomerMissingNameParams.UINameEditText;
    // Type '{Tab}' in 'Name' text box
    Keyboard.SendKeys(uiNameEdit,
        this.CreateNewCustomerMissingNameParams.UINameEditSendKeys,
        ModifierKeys.None);
    // Type '234 Lane St' in 'Address' text box
    uiAddressEdit.Text =
        this.CreateNewCustomerMissingNameParams.UIAddressEditText;
    // Type '{Tab}' in 'Address' text box
    Keyboard.SendKeys(uiAddressEdit,
        this.CreateNewCustomerMissingNameParams.UIAddressEditSendKeys,
        ModifierKeys.None);
    }
}
```

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

MICROSOFT HEADQUARTERS,
REDMOND, WA

AUGUST 3 - 7, 2015

ENGINEERED FOR YOU @ THE SOURCE

Join us August 3 – 7, 2015 for TechMentor 2015:
Datacenter Edition, focused entirely on making your datacenter
more modern, capable, and manageable through 5 days of
immediately usable IT education.

THE AGENDA FEATURES:

- ✦ 75-minute topic overview breakout sessions
- ✦ 3 hour content-rich deep dives
- ✦ "Hands on" labs with your own laptop

CONTENT AREAS INCLUDE:

- ✦ Windows PowerShell
- ✦ Infrastructure Foundations
- ✦ Runbook Automation
- ✦ Configuration and Service Management
- ✦ Datacenter Provisioning and Deployment

SAVE \$400!

WHEN YOU REGISTER BY MAY 20

USE PROMO CODE **TMMAY1**



Scan the QR code to register or for more event details.

EVENT SPONSOR:



PLATINUM SPONSOR:



GOLD SPONSORS:



SUPPORTED BY:



TECHMENTOREVENTS.COM/REDMOND

PRODUCED BY: 1105 MEDIA



Gone Viral

If you want to sleep tonight, close this magazine now.

Couldn't do it, could you? OK, read on, and then stay awake along with me.

We've wiped out smallpox, probably the greatest public health triumph in human history. The last endemic case occurred in 1978. Wikipedia describes it in the past tense: "Smallpox was an infectious disease" But you and I, my fellow geeks, are bringing it back.

We were able to eradicate smallpox because the virus had no host other than humans, no animal reservoirs as rabies or influenza have. But Leonard Adleman, professor of computer science and molecular biology at USC, and the "A" in RSA, wrote a column last fall (nyti.ms/19QH68q) and said, "... the smallpox virus has now found a second host. It is not the pig. In fact, it is not even what we think of as a living thing. It is the computer."

Think you'll sleep tonight if you slam this magazine shut right now, or close your browser and erase your history like your boss just knocked on your door? Too late, my friend. Too late.

"Smallpox has miraculously and unconsciously saved itself through an extraordinary act of evolution," Adleman wrote. "After thousands of years, it was on the verge of extinction; it existed in one small girl, and just before that girl's immune system killed its last living member, a sample was taken and stored in a lab. Years later, that sample was used by another lab to sequence the viral genome. The sequence was placed on a computer, infecting a new 'species' that had just come into existence." That new host species is our creation, yours and mine.

You can't find the smallpox genome online today, at least not reliably, at least not yet. But do you seriously think it will stay buried for long? One word: Snowden. Would WikiLeaks publish it? Probably. And if not, then some other organization.

How bad can that be, I hear you wondering. Constructing an actual virus from a genetic sequence is difficult, isn't it? No, not enormously. It was done with polio in 2002 (see 1.usa.gov/1N9e2a2). And like all technology, it's getting easier and cheaper all the time. Again, Adleman:

"In my lab there was a machine much like a soda dispenser, only in this case the reservoirs were filled with chemicals. If I typed

in a short word of my choice using the letters A, T, C and G, the machine would squirt one chemical after another into a test tube. When it was done, the test tube would contain trillions of molecules of DNA. Each would look like a necklace, with molecules of adenine, thymine, cytosine and guanine (the building blocks of DNA) strung according to the word I had typed."

It's easy to mock this fear. Will Kaspersky updates now include cowpox? Or will Clippy pop up while I'm writing this: "I see you're synthesizing a deadly bioweapon. Can I help?" Or vaccine rejectionist Jenny McCarthy say, "Let's have smallpox parties so our kids can acquire immunity naturally"?

You can't find the smallpox genome online today, at least not reliably, at least not yet. But do you seriously think it will stay buried for long?

You could say, hey, no biggie. This isn't Ebola. We've had a safe, cheap, reliable smallpox vaccine for 200 years. I remember getting a vaccination at age six, then another around 15—not a huge deal. It doesn't even need refrigeration any more.

All true. But look at the panic in the United States last fall about Ebola reaching our shores—all from four cases and one death. I don't want to be in New York City the day a guy infected with smallpox rides the subway.

It's coming. We're bringing it. Unintentionally, as a side effect of beneficial activities, but we're the ones unlocking it. And I'm having real trouble sleeping, knowing that our creations are resurrecting this deadly enemy of humanity. ■

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.

I hope to be at the Microsoft Ignite conference in Chicago May 4-8. You'll find this magazine in your attendee bags. Look for me to get it autographed, unless you'd rather leave it undamaged. I'll be the one with bags under my eyes from no sleep. And if you've read this far, so will you.

Build Experiences

for Your Users with Enterprise-level
Developer Tools and Solutions



Wijmo

A New Generation of
JavaScript Controls
wijmo.com



Spread

Versatile Spreadsheet
Data and UI Components
spread.grapecity.com



Xuni

Cross-Platform Data
Visualization Controls
componentone.com/xuni



ComponentOne Studio

.NET Tools for Serious
Application Development
componentone.com



ActiveReports

Reporting Platform for
Essential Business Needs
activereports.grapecity.com

The GrapeCity family of products provides developers, designers, and architects the ultimate collection of easy-to-use tools for building sleek, high-performing, feature complete applications. With over 25 years of experience, we understand your needs and have developed a comprehensive line of products that includes innovative UI controls, cross-platform data visualization controls, enterprise-level reporting, analysis, and spreadsheet controls for Windows, web, and mobile platforms.

Learn more and receive
free 30-day trials at
tools.grapecity.com





650 CONTROLS

...

12 PLATFORMS

...

A \$9,975 VALUE

**ESSENTIAL
STUDIO
ENTERPRISE
EDITION**

COMMUNITY LICENSE

ABSOLUTELY FREE

WHAT DO YOU GET?

- ✓ A license that never expires.
- ✓ Free access to our entire product offering.
- ✓ A license to build commercial applications.

WHO QUALIFIES?

Individual developers, or up to five users at companies with less than \$1 million USD in annual gross revenue, are eligible for the Community License.



www.syncfusion.com/communitylicense

 **Syncfusion®**



ASPOSE

Powerful File APIs that are easy and intuitive to use

Native APIs for
.NET, Java & Cloud

Adding File Conversion and Manipulation
to Business Systems

DOC, XLS, JPG, PNG, PDF,
BMP, MSG, PPT, VSD, XPS &
many other formats.

 www.aspose.com

Working with Files?



- ✓ CONVERT
- ✓ PRINT
- ✓ CREATE
- ✓ COMBINE
- ✓ MODIFY

100% Standalone - No Office Automation

Scan for
20% Savings!



ASPOSE

Your File Format APIs

ASPOSE.TOTAL

Every Aspose component combined in
ONE powerful suite!



Powerful File Format APIs

- ▶ **Aspose.Words**
DOC, DOCX, RTF, HTML, PDF, XPS & other document formats.
 - ▶ **Aspose.Cells**
XLS, XLSX, XLSM, XLTX, CSV, SpreadsheetML & image formats.
 - ▶ **Aspose.BarCode**
JPG, PNG, BMP, GIF, TIF, WMF, ICON & other image formats.
 - ▶ **Aspose.Pdf**
PDF, XML, XLS-FO, HTML, BMP, JPG, PNG & other image formats.
 - ▶ **Aspose.Email**
MSG, EML, PST, EMLX & other formats.
 - ▶ **Aspose.Slides**
PPT, PPTX, POT, POTX, XPS, HTML, PNG, PDF & other formats.
 - ▶ **Aspose.Diagram**
VSD, VSDX, VSS, VST, VSX & other formats.
- ... and many others!*

Aspose.Total for .NET

Aspose.Total for Java

Aspose.Total for Cloud

Get your FREE evaluation copy at www.aspose.com

.NET

Java

Cloud

Aspose.Cells

Work with spreadsheets and data without depending on Microsoft Excel

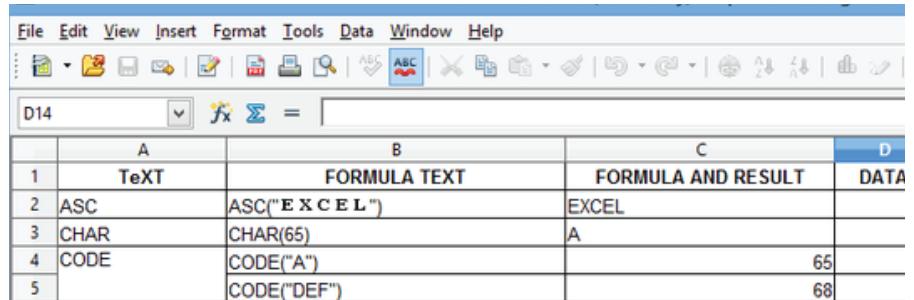
- Solution for spreadsheet creation, manipulation and conversion.
- Import and export data.

ASPOSE.CELLS IS A PROGRAMMING API that allows developers to create, manipulate and convert Microsoft Excel spreadsheet files from within their own applications. Its powerful features make it easy to convert worksheets and charts to graphics or save reports to PDF.

Aspose.Cells speeds up working with Microsoft Excel files. The API is a flexible tool for simple tasks such as file conversion, as well as complex tasks like building models. Developers control page layout, formatting, charts and formulas. They can read and write spreadsheet files and save out to a wide variety of image and text file formats.

Fast and reliable, Aspose.Cells saves time and effort compared to using Microsoft Office Automation.

A flexible API for simple and complex spreadsheet programming.



	A	B	C	D
1	TeXT	FORMULA TEXT	FORMULA AND RESULT	DATA
2	ASC	ASC("EXCEL")	EXCEL	
3	CHAR	CHAR(65)	A	
4	CODE	CODE("A")		65
5		CODE("DEF")		68

Aspose.Cells lets developers work with data sources, formatting, even formulas.

Common Uses

- Building dynamic reports on the fly.
- Creating Excel dashboards with charts and pivot tables.
- Rendering and printing spreadsheets and graphics with high fidelity.
- Exporting data to, or importing from, Excel spreadsheets.
- Generating, manipulating and editing spreadsheets.
- Converting spreadsheets to images or other file formats.

Key Features

- A complete spreadsheet manipulation solution.
- Flexible data visualization and reporting.
- Powerful formula engine.
- Complete formatting control.

Supported File Formats

XLS, XLSX, XLSM, XMPS, XLTX, XLTM, ODS, SpreadsheetML, tab delim., CSV, TXT, PDF, HTML, and many image formats including TIFF, JPEG, PNG and GIF.

Format support varies across platforms.

Platforms



Pricing Info					
	Standard		Enhanced		
Developer Small Business	\$999	\$1498	Site Small Business	\$4995	\$7490
Developer OEM	\$2997	\$4494	Site OEM	\$13986	\$20972

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Aspose.Cells for

.NET, Java, Cloud & more

File Formats

XLS XLSX TXT PDF HTML CSV TIFF PNG JPG BMP SpreadsheetML and many others.

Spreadsheet Manipulation

Aspose.Cells lets you create, import, and export spreadsheets and also allows you to manipulate contents, cell formatting, and file protection.

Creating Charts

Aspose.Cells comes with complete support for charting and supports all standard chart types. Also, you can convert charts to images.

Graphics Capabilities

Easily convert worksheets to images as well as adding images to worksheets at runtime.

Get your FREE Trial at
<http://www.aspose.com>

No Office Automation

Aspose.Cells does not require Microsoft Office to be installed on the machine in order to work.

Aspose.Words

Program with word processing documents independently of Microsoft Word

- Solution for document creation, manipulation and conversion.
- Advanced mail merge functionality.

ASPOSE.WORDS IS AN ADVANCED PROGRAMMING API

that lets developers perform a wide range of document processing tasks with their own applications. Aspose.Words makes it possible to generate, modify, convert, render and print documents without Microsoft Office Automation. It provides sophisticated and flexible access to, and control over, Microsoft Word files.

Aspose.Words is powerful, user-friendly and feature rich. It saves developers time and effort compared to using Microsoft Office Automation and makes gives them powerful document management tools.

Aspose.Words makes creating, changing and converting DOC and other word processing file formats fast and easy.

Generate, modify, convert, render and print documents without Microsoft Office Automation.

	Table			
	Column 1	Column 2	Column 3	Column 4
Row 1	Cell 1	Cell 2	Cell 3	Cell 4
Row 2	Cell 1	Cell 2	Cell 3	
Row 3	Cell 1	Cell 2		

Aspose.Words has sophisticated controls for formatting and managing tables and other content.

Common Uses

- Generating reports with complex mail merging; mail merging images.
- Populating tables and documents with data from a database.
- Inserting formatted text, paragraphs, tables and images into Microsoft Word documents.
- Adding barcodes to documents.
- Inserting diagrams and watermarks into Word documents.
- Formatting date and numeric fields.

Key Features

- A complete Microsoft Word document manipulation solution.
- Extensive mail merge features.
- Complete formatting control.
- High-fidelity conversion, rendering and printing.

Supported File Formats

DOC, DOCX, ODT, OOXML, XML, HTML, XHTML, MHTML, EPUB, PDF, XPS, RTF, and a number of image formats, including TIFF, JPEG, PNG and GIF.

Format support varies across platforms.

Platforms



	Pricing Info					
	Standard	Enhanced		Standard	Enhanced	
Developer Small Business	\$999	\$1498	Site Small Business	\$4995	\$7490	
Developer OEM	\$2997	\$4494	Site OEM	\$13986	\$20972	

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Case Study: Aspose.Words for .NET

ModulAcht e.K. - using Aspose.Words for .NET to convert from DOCX to PDF.

MODULACHT IS A SOFTWARE DEVELOPMENT TEAM WHICH CREATES INDIVIDUAL SOFTWARE for small businesses. Mostly we develop web applications including web UI and web-services, but we are also familiar with Windows Forms and Windows Services applications based on .NET.

Problem

For our main customer, we are developing the operating system they will use to administer the buying and selling of cars. With a need to generate documents easily, one of the main requirements was to have an easy-to-use template system.

“The really quick and competent support of Aspose helped us to solve some initial problems.”

Looking for a Solution

We searched on the internet for DOCX to PDF converters, which is not as easy as it sounds. After filtering all the Interop wrappers only a handful of components remained to be tested. At the end only Aspose.Words for .NET created a result which really looks like the input DOCX. The really quick and competent support of Aspose helped us to solve some initial problems.

Implementation

Aspose.Words for .NET was the 4th component we tested. On our development machine, everything worked great, but after moving the code on to our test-server-machine, the resulting PDF did not look like the original DOCX file. Adjusting the settings didn't help so we decided give the support team of Aspose a try.

After a short discussion in the live chat we started a new thread including a description, the input and the output file, in the Aspose.Words forum. Within less than 24 hours one of the support-team member told us that we would

have to check whether the font we used in the DOCX file was available on the server machine, which it was not. After changing the font, the

whole PDF looks exactly the same as the DOCX file.

Outcome

Choosing Aspose.Words for .NET meant an intuitive and easy to use software component and also getting a really friendly and straightforward software partner which is ready to help if you need help.

Next Steps

After getting our Test-Driver ready we will implement the template engine in our customer's software. Aspose.Words for .NET functionality will be used on many different places in this software to convert files into the PDF format.

This is an extract from a case study on our website. For the full version, go to: www.aspose.com/corporate/customers/case-studies.aspx



After converting, our PDF looks exactly the same as the DOCX file.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926



Open, Create, Convert, Print & Save Files

from within your *own* applications.

> ASPOSE.TOTAL

allows you to process these file formats:

- Word documents
- Excel spreadsheets
- PowerPoint presentations
- PDF documents
- Project documents
- Visio documents
- Outlook emails
- OneNote documents



**DOC XLS PPT PDF EML
PNG XML RTF HTML VSD
BMP & barcode images.**



Contact Us:

US: +1 888 277 6734

EU: +44 141 416 1112

AU: +61 2 8003 5926

sales@aspose.com

Helped over 11,000 companies and over 250,000 users work with documents in their applications.



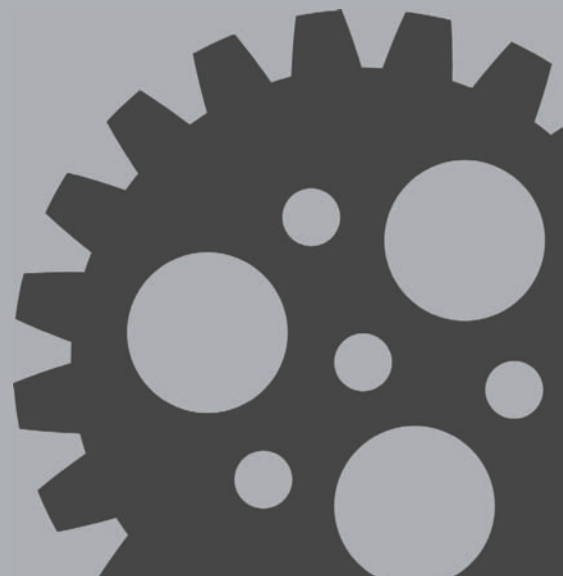
.NET, Java, and Cloud

Your File Format APIs



GET STARTED NOW

- Free Trial
- 30 Day Temp License
- Free Support
- Community Forums
- Live Chat
- Blogs
- Examples
- Video Demos



Adding File Conversion and Manipulation to Business Systems

How often do people in your organization complain that they can't get information in the file format and layout they want? Converting documents from one format to another without losing layout and formatting should be simple, but it can be frustrating for both users and developers.

EXTRACTING DATA FROM A DATABASE AND DELIVERING IT TO THE SALES TEAM AS A REPORT, complete with charts and corporate branding, is fine. Until the sales team says that they want it as a Microsoft Excel file, and could you add a dashboard?

Using information from online forms in letters that can be printed and posted is easy. But what if you also want to add tracking barcodes and archive a digital copy as a PDF?

Ensuring that your business system supports all the different Microsoft Office file formats your users want can be difficult. Sometimes the native file format support of your system lets you down. When that is the case, use tools that extend that capability. A good tool can save you time and effort.

Document Conversion Options

Building your own solution: Time-consuming and costly, this option is only sensible if the solution you develop is central to your business.

Using Microsoft Office

Automation: Microsoft Office

Automation lets you use Microsoft Office programs server-side. It is not how the Office products were designed to be used. It can work well but you might notice issues with the stability, security and speed of the system, as well as cost.

Using an API: The API market has lots of free and commercial solutions, some very focused, some feature-rich. An API integrates with your code and gives you access to a range of new features.

barcodes and OCR. The APIs are optimised for stability, speed and ease of use. Our APIs save users weeks, sometimes months, of effort.

Aspose creates APIs that work independently of Microsoft Office Automation.



Finding the Right Tool

To find the product that's right for you, take a systematic approach:

- List must-have and nice-to-have features.
- Research the market.
- Ask for recommendations.
- Select a few candidates .
- Run trials.
- Evaluate
 - ease of use,
 - support and documentation,
 - performance, and
 - current and future needs.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Aspose.BarCode

A complete toolkit for barcode generation and recognition

- Generate barcodes with customer defined size and color.
- Recognize a large number of barcode types from images.

ASPOSE.BARCODE IS A ROBUST AND RELIABLE BARCODE GENERATION AND RECOGNITION API that allows developers to add barcode generation and recognition functionality to their applications quickly and easily.

Aspose.BarCode supports most established barcode specifications. It can export generated barcodes to multiple image formats, including BMP, GIF, JPED, PNG and TIFF.

Aspose.BarCode gives you full control over every aspect of the barcode

image, from background and bar color, through image quality, rotation angle, X-dimension, captions, and resolution.

Aspose.BarCode can read and recognize most common 1D and 2D barcodes from any image and at any angle. Filters help developers

Robust and reliable barcode generation and recognition.



Aspose.BarCode offers a large number of symbologies and formatting options.

clean up difficult to read images to improve recognition.

Common Uses

- Generating and recognizing barcode images.
- Printing barcode labels.
- Enhancing workflow by adding barcode functionality.
- Using recognition functions to drive real-life work processes.

Key Features

- Barcode generation and recognition.
- Comprehensive support for 1D and 2D symbologies.
- Image processing for improved recognition.

Supported File Formats

JPG, TIFF, PNG, BMP, GIF, EMF, WMF,

EXIP and ICON.

Format support varies across platforms.

Supported Barcodes

Linear: EAN13, EAN8, UPCA, UPCE, Interleaved2of5, Standard2of5, MSI, Code11, Codabar, EAN14(SCC14), SSCC18, ITF14, Matrix 2 of 5, PZN, Code128, Code39 Extended, Code39 Standard, OPC, Code93 Extended, Code93 Standard, IATA 2 of 5, GS1Code128, ISBN, ISMN, ISSN, ITF6, Pharmacode, DatabarOmniDirectional, VIN, DatabarTruncated, DatabarLimited, DatabarExpanded, PatchCode, Supplement 2D: PDF417, MacroPDF417, DataMatrix, Aztec, QR, Italian Post 25, Code16K, GS1DataMatrix **Postal:** Postnet, Planet, USPS OneCode, Australia Post, Deutsche Post Identcode, AustralianPosteParcel, Deutsche Post Leticode, RM4SCC, SingaporePost, SwissPostParcel

Platforms



	Pricing Info				
	Standard	Enhanced	Standard	Enhanced	
Developer Small Business	\$599	\$1098	Site Small Business	\$2995	\$5490
Developer OEM	\$1797	\$3294	Site OEM	\$8386	\$15372

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926









Aspose *for* Cloud

The easiest API to Create, Convert & Automate Documents in the cloud.



**Convert
Create
Render
Combine
Modify**

without installing anything!

<p>Aspose.Words for Cloud </p> <p>Create and convert docs Manipulate text Render documents Annotate</p>	<p>Aspose.Cells for Cloud </p> <p>Create spreadsheets Convert spreadsheets Manipulate cells and formulas Render spreadsheets</p>
<p>Aspose.Slides for Cloud </p> <p>Create presentations Manage slides Edit text and images Read and convert</p>	<p>Aspose.Pdf for Cloud </p> <p>Create and convert PDFs Manipulate text, images Add pages, split, encrypt Manage stamps</p>
<p>Aspose.Email for Cloud </p> <p>Create, update, and convert messages Extract attachments Use with any language</p>	<p>Aspose.BarCode for Cloud </p> <p>Generate barcodes Read barcodes Set attributes Multiple image formats</p>

Free Evaluation at www.aspose.com

Aspose.Email

Work with emails and calendars without Microsoft Outlook

- Complete email processing solution.
- Message file format support.

ASPOSE.EMAIL IS AN EMAIL PROGRAMMING API that allows developers to access and work with PST, EML, MSG and MHT files. It also offers an advanced API for interacting with enterprise mail systems like Exchange and Gmail.

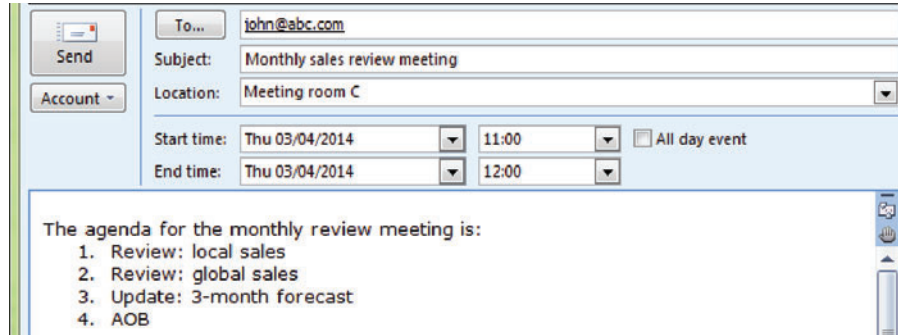
Aspose.Email can work with HTML and plain text emails, attachments and embedded OLE objects. It

allows developers to work against SMTP, POP, FTP and Microsoft Exchange servers. It supports mail

merge and iCalendar features, customized header and body, searching archives and has many other useful features.

Aspose.Email allows developers to focus on managing email without getting into the core of email and network programming. It gives you the controls you need.

Aspose.
Email works
with HTML
and plain
text emails,
attachments
and embedded
OLE objects.



Aspose.Email lets your applications work with emails, attachments, notes and calendars.

Common Uses

- Sending email with HTML formatting and attachments.
- Mail merging and sending mass mail.
- Connecting to POP3 and IMAP mail servers to list and download messages.
- Connecting to Microsoft Exchange Servers to list, download and send messages.
- Create and update tasks using iCalendar.
- Load from and save messages to file or stream (EML, MSG or MHT formats).

Key Features

- A complete email processing solution.
- Support for MSG and PST formats.
- Microsoft Exchange Server support.
- Complete recurrence pattern solution.

Supported File Formats

MSG, MHT, OST, PST, EMLX, TNEF, and EML.

Format support varies across platforms.

Platforms



	Pricing Info				
	Standard	Enhanced	Standard	Enhanced	
Developer Small Business	\$599	\$1059	Site Small Business	\$2995	\$5490
Developer OEM	\$1797	\$3294	Site OEM	\$8386	\$15372

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Aspose.Pdf

Create PDF documents without using Adobe Acrobat

- A complete solution for programming with PDF files.
- Work with PDF forms and form fields.

ASPOSE.PDF IS A PDF DOCUMENT CREATION AND MANIPULATION API

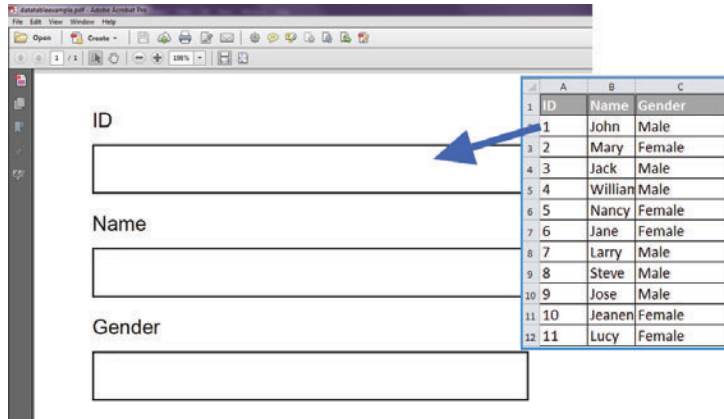
that developers use to read, write and manipulate PDF documents without using Adobe Acrobat. Aspose.Pdf is a sophisticated product that integrates with your application to add PDF capabilities.

Aspose.Pdf offers a wealth of features that lets developers compress files, create tables, work with links,

add and remove security, handle custom fonts, integrate with external data sources, manage bookmarks, create table of contents, create forms and manage form fields.

Read, write and manipulate PDF documents independently of Adobe Acrobat.

It helps developers add, work with attachments, annotations and PDF form data, add, replace or remove text and images, split, concatenate,



Aspose.Pdf can be used to automatically complete PDF forms with external data.

extract or inset pages, and print PDF documents.

Common Uses

- Creating and editing PDF files.
- Inserting, extracting, appending, concatenating and splitting PDFs.
- Working with text, images, tables, images, headers, and footers.
- Applying security, passwords and signatures.
- Working with forms and form fields.

Key Features

- PDF creation from XML or XLS-FO documents.
- PDF form and field support.
- Advanced security and encryption.
- High-fidelity printing and conversion.
- Supported File Formats
- PDF, PDF/A, PDF/A_1b, PCL, XLS-FO, LaTeX, HTML, XPS, TXT and a range of image formats.

Format support varies across platforms.

Platforms



	Pricing Info				
	Standard	Enhanced		Standard	Enhanced
Developer Small Business	\$799	\$1298	Site Small Business	\$3995	\$6490
Developer OEM	\$2397	\$3894	Site OEM	\$11186	\$18172

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Aspose.Pdf

.Net, Java & Cloud

File Formats

PDF XPS ePUB HTML XML XLS TXT DOC XSL-FO & other image file formats.

Create and Manipulate PDFs

Create new or edit/manipualte existing PDFs.

Form Field Features

Add form fields to your PDFs. Import and export form fields data from select file formats.

Table Features

Add tables to your PDFs with formatting such as table border style, margin and padding info, column width and spanning options, and more.

Get started today at www.aspose.com



Conversion is Fast And High-Fidelity



Aspose.Note for .NET

Aspose.Note for .NET is an API that lets developers convert Microsoft OneNote pages to a variety of file formats, and extract the text and document information.

Conversion is fast and high-fidelity. The output looks like the OneNote page, no matter how complex the formatting or layout.

Aspose.Note works independently of Office Automation and does not require Microsoft Office or OneNote to be installed.

Product	Benefit	Supported Platforms
Aspose.Note for .NET	Modify, convert, render and extract text and images from Microsoft OneNote files without relying on OneNote or other libraries.	.NET Framework 2.0, 3.0, 3.5, 4.0, 4.0 CP

Features

File Formats and Conversion		Rendering and Printing	Document Management
Microsoft OneNote 2010, 2010 SP1, 2013	Load, Save	Save as Image (BMP, GIF, JPG, PNG)	<ul style="list-style-type: none">• Extract text• Get the number of pages in a document.• Get page information.• Extract images.• Get image information from a document.• Replace text in document.
PDF	Save	Save as PDF	
Images (BMP, GIF, JPG, PNG)	Save		

Aspose.Imaging

Create Images from scratch.

- Load existing images for editing purposes.
- Render to multiple file formats.

ASPOSE.IMAGING IS A CLASS LIBRARY

that facilitates the developer to create Image files from scratch or load existing ones for editing purpose. Also, Aspose.Imaging provides the means to save the created or edited Image to a variety of formats. All of the above mentioned can be achieved without the need of an Image Editor. It works independent of other applications and although Aspose.Imaging allows you to save to Adobe PhotoShop® format (PSD), you do not need PhotoShop installed on the machine.

Aspose.Imaging is flexible, stable and powerful. It's many features and image processing routines should meet most imaging requirements. Like all Aspose file format components, Aspose.

Imaging introduces support for an advanced set of drawing features along with the core functionality. Developers can

Create images from scratch. or load existing ones...



Aspose.Imaging allows creation and manipulation of images.

draw on Image surface either by manipulating the bitmap information or by using the advanced functionality like Graphics and Paths.

Common Uses

- Create images from scratch.
- Load and Edit existing images.
- Export images to a variety of formats.
- Adding watermark to images.
- Export CAD drawings to PDF & raster image formats.
- Crop, resize & RotateFlip images.
- Extract frames from multipage TIFF image.

Key Features

- Create, edit, and save images
- Multiple file formats
- Drawing features
- Export images

Supported File Formats

BMP, JPG, TIFF, GIF, PNG, PSD, DXF, DWG, and PDF.

Platforms



	Pricing Info				
	Standard	Enhanced	Standard	Enhanced	
Developer Small Business	\$399	\$898	Site Small Business	\$1995	\$4490
Developer OEM	\$1197	\$2694	Site OEM	\$5586	\$12572

The pricing info above is for .NET.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Aspose.Slides

Work with presentations without using Microsoft PowerPoint

- Complete solution for working with presentation files.
- Export presentations and slides to portable or image formats.

ASPOSE.SLIDES IS A FLEXIBLE PRESENTATION MANAGEMENT API that helps developers read, write and manipulate Microsoft PowerPoint documents. Slides and presentations can be saved to PDF, HTML and image file formats without Microsoft Office Automation.

Aspose.Slides offers a number of advanced features that make it easy to perform tasks such as

Aspose.Slides gives you the tools you need to work with presentation files.

rendering slides, exporting presentations, exporting slides to SVG and printing. Developers use Aspose.Slides to build customizable slide decks, add or remove standard graphics and automatically publish presentations to other formats.

Aspose.Slides gives developers the tools they need to work with presentation files. It integrates quickly and saves time and money.



Aspose.Slides has advanced features for working with every aspect of a presentation.

Common Uses

- Creating new slides and cloning existing slides from templates.
- Handling text and shape formatting.
- Applying and removing protection.
- Exporting presentations to images and PDF.
- Embedding Excel charts as OLE objects.
- Generate presentations from database.

external content.

- Wide support for input and output file formats.

Supported File Formats

PPT, POT, PPS, PPTX, POTX, PPSX, ODP, PresentationML, XPS, PDF and image formats including TIFF and JPG.

Format support varies across platforms.

Key Features

- A complete presentation development solution.
- Control over text, formatting and slide elements.
- OLE integration for embedding

Platforms



	Pricing Info				
	Standard	Enhanced	Standard	Enhanced	
Developer Small Business	\$799	\$1298	Site Small Business	\$3995	\$6490
Developer OEM	\$2397	\$3894	Site OEM	\$11186	\$18172

The pricing info above is for .NET: prices for other platforms may differ. For the latest, contact sales.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

Support Services

Get the assistance you need, when you need it, from the people who know our products best.

- Use experienced Aspose developers for your projects
- Get the level of support that suits you and your team

NO ONE KNOWS OUR PRODUCTS AS WELL AS WE DO.

We develop them, support them and use them. Our experience is available to you, whether you want us to develop a solution for you, or you just need a little help to solve a particular problem.

Consulting

Aspose's developers are expert users of Aspose APIs. They understand how to use our products and have hands-on experience of using them for software development. Aspose's developers are skilled not just with Aspose tools but in a wide range of programming languages, tools and techniques.

When you need help to get a project off the ground, Aspose's developers can help.

Aspose's file format experts are here to help you with a project or your support questions



Work with the most experienced Aspose developers in the world.

Consulting Benefits

- Use Aspose engineers to work on your products
- Get peace of mind from a fully managed development process
- Get a custom-built solution that meets your exact needs

Support Options

Free

Everyone who uses Aspose products have access to our free support. Our software developers are on stand-by to help you succeed with your project, from the evaluation to roll-out of your solution.

Priority

If you want to know when you'll hear back from us on an issue, and know that your issue is prioritized, Priority Support is for you. It provides a more formal support structure and has its own forum that is monitored by our software engineers.

Enterprise

Enterprise customers often have very specific needs. Our Enterprise Support option gives them access to the product development team and influence over the roadmap. Enterprise Support customers have their own, dedicated issue tracking system.



Pricing Info

Each consulting project is evaluated individually; no two projects have exactly the same requirements.

To see the Priority and Enterprise support rates, refer to the product price list, or contact our sales team.

www.aspose.com

EU: +44 141 416 1112

US: +1 888 277 6734
sales@aspose.com

Oceania: +61 2 8003 5926

We're Here to Help You

Aspose has 4 Support Services to best suit your needs

Free Support

Support Forums with no Charge

Priority Support

24 hour response time in the week,
issue escalation, dedicated forum

Enterprise Support

Communicate with product
managers, influence the roadmap

Sponsored Support

Get the feature you need built now

Technical Support is an issue that Aspose takes very seriously. Software must work quickly and dependably. When problems arise, developers need answers in a hurry. We ensure that our clients receive useful answers and solutions quickly.

Email • Live Chat • Forums

Contact Us

US Sales: +1 888 277 6734
sales@aspose.com

EU Sales: +44 141 416 1112

AU Sales: +61 2 8003 5926

