THE MICROSOFT JOURNAL FOR DEVELOPERS

DECEMBER 2014 VOL 29 NO 12

msdn magazine

Your Next Great ASP.NET App Starts Here

Deliver interactive touch-enabled user experiences for WebForms and MVC with elegant, high-performance UI controls and extensions from DevExpress.

Download your free 30-day trial at: DevExpress.com/ASP

Waikiki Beach Hotel

800 Waikiki Beach Rd Honolulu, HI, 96801

Situated directly on Waikiki Beach you are 100 yards away from the fun that awaits you whenever you visit Honolulu. The hotel offers free shuttle service from Honolulu International Airport and we provide you with free towels when using our pool.

Jun 7 - Jun 9, 2014
2 nights, 1 room, 2 adults

What our guests say...

I finally saved up enough money to stay here and I tell ya - this place deserves all the praise it gets. Wondeful!
Jan 13, 2014

If you've never experienced the Waikiki Beach then I suggest you fly over to Hawaii and see why it's the best.
Feb 15, 2014

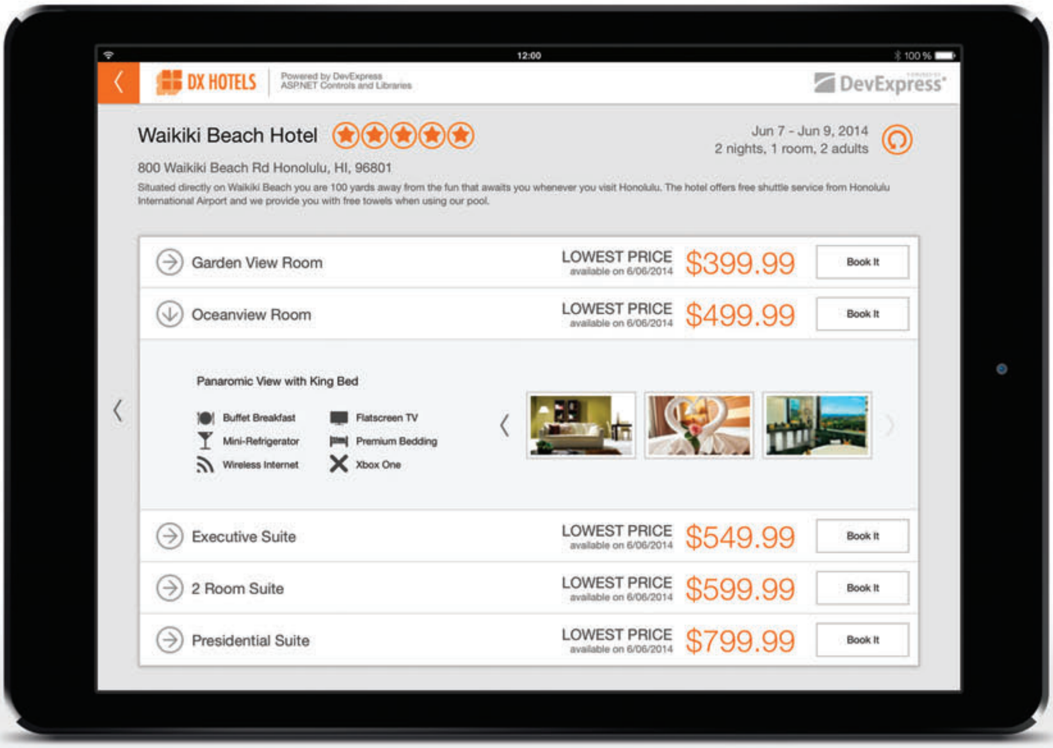
We stayed here on c What a hotel! It's e it's totally worth it.
Mar 19, 2014

DevExpress®

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

Become a UI Superhero

Learn More at DevExpress.com/Superhero



Waikiki Beach Hotel ★★★★★

Jun 7 - Jun 9, 2014
2 nights, 1 room, 2 adults

800 Waikiki Beach Rd Honolulu, HI, 96801

Situated directly on Waikiki Beach you are 100 yards away from the fun that awaits you whenever you visit Honolulu. The hotel offers free shuttle service from Honolulu International Airport and we provide you with free towels when using our pool.

Garden View Room	LOWEST PRICE available on 6/06/2014	$399.99	Book It
Oceanview Room	LOWEST PRICE available on 6/06/2014	$499.99	Book It

Panaromic View with King Bed

- Buffet Breakfast
- Mini-Refrigerator
- Wireless Internet
- Flatscreen TV
- Premium Bedding
- Xbox One

Executive Suite	LOWEST PRICE available on 6/06/2014	$549.99	Book It
2 Room Suite	LOWEST PRICE available on 6/06/2014	$599.99	Book It
Presidential Suite	LOWEST PRICE available on 6/06/2014	$799.99	Book It

DevExpress®

WIN ASP WPF SL | XAF | CR

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

THE MICROSOFT JOURNAL FOR DEVELOPERS

DECEMBER 2014 VOL 29 NO 12

msdn magazine

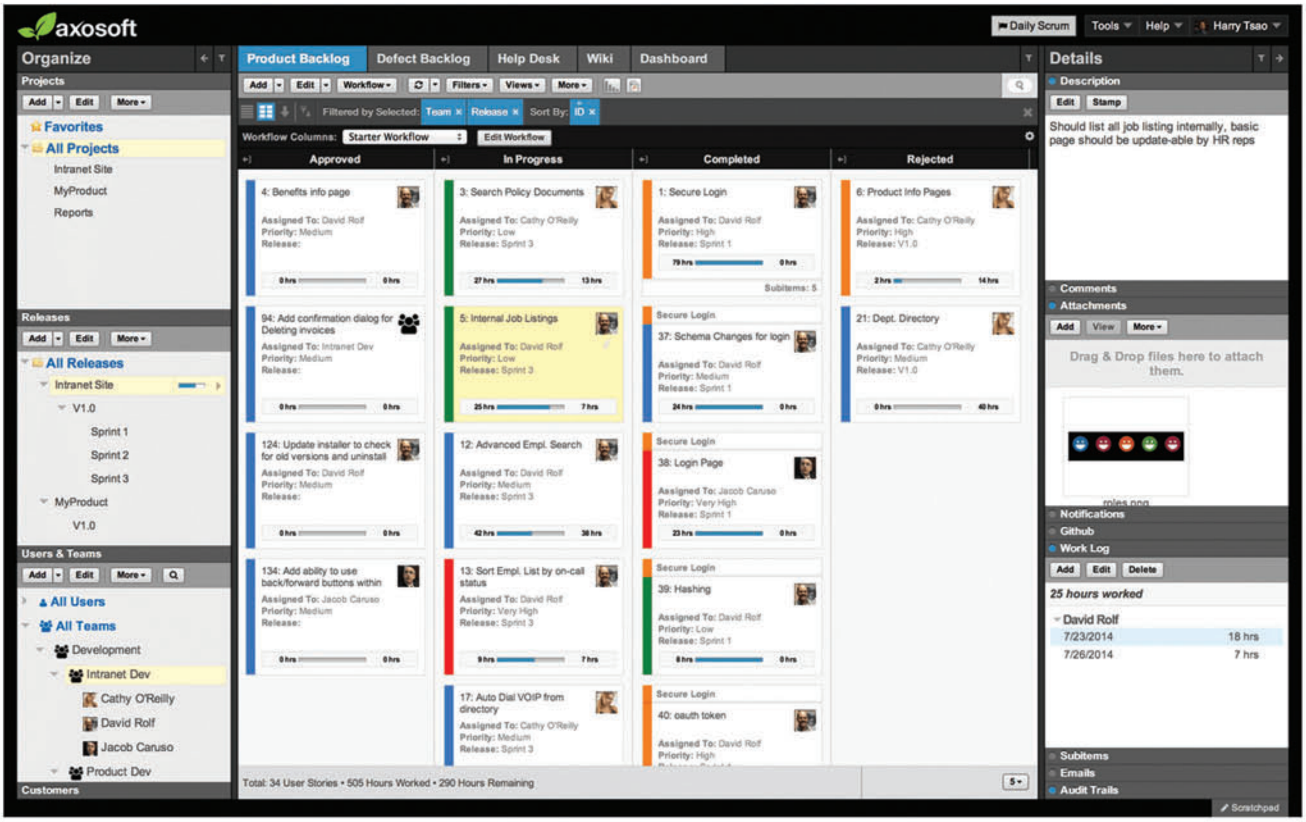
COLUMNS

Microsoft

Don't let your dev team get bogged down by complicated processes.

axosoft

Transform your team into a lean, mean Scrum machine with Axosoft!



Approved

4: Benefits info page
Assigned To: David Rolf
Priority: Medium
Release:
0 hrs 0 hrs

94: Add confirmation dialog for Deleting invoices
Assigned To: Intranet Dev
Priority: Medium
Release:
0 hrs 0 hrs

124: Update installer to check for old versions and uninstall
Assigned To: David Rolf
Priority: Medium
Release:
0 hrs 0 hrs

134: Add ability to use back/forward buttons within
Assigned To: Jacob Caruso
Priority: Medium
Release:
0 hrs 0 hrs

In Progress

3: Search Policy Documents
Assigned To: Cathy O'Reilly
Priority: Low
Release: Sprint 3
27 hrs 13 hrs

5: Internal Job Listings
Assigned To: David Rolf
Priority: Low
Release: Sprint 3
25 hrs 7 hrs

12: Advanced Empl. Search
Assigned To: David Rolf
Priority: Medium
Release: Sprint 3
42 hrs 38 hrs

13: Sort Empl. List by on-call status
Assigned To: David Rolf
Priority: Very High
Release: Sprint 3
9 hrs 7 hrs

17: Auto Dial VOIP from directory
Assigned To: Cathy O'Reilly
Priority: Medium
Release: Sprint 3

Completed

1: Secure Login
Assigned To: David Rolf
Priority: High
Release: Sprint 1
79 hrs 0 hrs
Subitems: 5

Secure Login
37: Schema Changes for login
Assigned To: David Rolf
Priority: Medium
Release: Sprint 1
24 hrs 0 hrs

Secure Login
38: Login Page
Assigned To: Jacob Caruso
Priority: Very High
Release: Sprint 1
23 hrs 0 hrs

Secure Login
39: Hashing
Assigned To: David Rolf
Priority: Low
Release: Sprint 1
8 hrs 0 hrs

Secure Login
40: oauth token
Assigned To: David Rolf
Priority: High

Rejected

6: Product Info Pages
Assigned To: Cathy O'Reilly
Priority: High
Release: V1.0
2 hrs 14 hrs

21: Dept. Directory
Assigned To: Cathy O'Reilly
Priority: Medium
Release: V1.0
0 hrs 40 hrs

Total: 34 User Stories • 505 Hours Worked • 290 Hours Remaining

Should list all job listing internally, basic page should be update-able by HR reps

25 hours worked

David Rolf

7/23/2014	18 hrs
7/26/2014	7 hrs

Empower management & team members with total visibility into your development process.

Axosoft Scrum provides key project insights with a Kanban card view that clearly illustrates an item's workflow status, while burndown charts and custom reports help management identify further opportunities for improvement. Out-of-the-box the tool is configured with Scrum best practices in mind! Get started free at **Axosoft.com/MSDNscrum**.

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.

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

Ask about fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

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

msdn magazine

DECEMBER 2014 VOLUME 29 NUMBER 12

KEITH BOYD Director
MOHAMMAD AL-SABT Editorial Director/mmeditor@microsoft.com
KENT SHARKEY Site Manager

MICHAEL DESMOND Editor in Chief/mmeditor@microsoft.com
LAFE LOW Features Editor
SHARON TERDEMAN Features Editor
DAVID RAMEL Technical Editor
WENDY HERNANDEZ Group Managing Editor

SCOTT SHULTZ Vice President, Art and Brand Design
JOSHUA GOULD Art Director

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

RedmondMediaGroup

Henry Allain President, Redmond Media Group
Becky Nagel Vice President, Digital Strategy
Michele Imgrund Vice President, Lead Services Division
Tracy Cook Director, Client Services & Webinar Production
Irene Fincher Director, Audience Development & Lead Generation Marketing

ADVERTISING SALES: 818-674-3416/dlabianca@1105media.com

Dan LaBianca Chief Revenue Officer
Chris Kourtoglou Regional Sales Manager
Danna Vedder Regional Sales Manager/Microsoft Account Manager
David Seymour Director, Print & Online Production
Anna Lyn Bayaua Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA INC

Rajeev Kapur Chief Executive Officer
Henry Allain Chief Operating Officer
Richard Vitale Senior Vice President & Chief Financial Officer
Michael J. Valenti Executive Vice President

Erik A. Lindgren Vice President, Information Technology & Application Development

Jeffrey S. Klein Chairman of the Board

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, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@1105media.com

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

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

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.

Microsoft

BPA WORLDWIDE

Printed in the USA

OCR — BARCODE — VIEWERS & CONVERTERS

PDF — **DOCUMENT** — FORMS

OFFICE FORMATS

DICOM — HL7

WORKSTATION — STORAGE SERVER

PACS – **MEDICAL** — 3D

LEADTOOLS® V19

THE WORLD LEADER IN IMAGING SDKs

DVR — CAPTURE

PLAYBACK – **MULTIMEDIA** – CODECS

H.265 H.264

MPEG-2 TS RTSP

COMPRESSION — **IMAGING** – FORMATS

IMAGE PROCESSING — TWAIN

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

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

Medical
Imaging SDKs

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

Multimedia
Imaging SDKs

- *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

LEAD TECHNOLOGIES INCORPORATED

SALES@LEADTOOLS.COM
800.637.1840

EDITOR'S NOTE

MICHAEL DESMOND

The Greatest Toy of All Time

I am a child of the 1970s. I was raised on American steel, Led Zeppelin and The Six Million Dollar Man. I marveled at Pong, was a beast at Rock 'Em Sock 'Em Robots, and never understood all the panic and concern over lawn darts. Those things were awesome.

But like any child of a particular decade, I tend to wax nostalgic about the places, events and things I experienced growing up. Which is why I never forgave Bruce Springsteen for "Born in the USA," and still regard the old, handheld Mattel Electronics Football game as the greatest toy of all time.

> You know the universal head-down, hands-forward posture of commuters staring at their smartphones? Mattel Football invented that.

For a kid weaned on meatspace fare like tabletop hockey and the ball-in-a-tilting maze game Labrynth, Mattel's handheld Football was a revelation. Here was a simple, compact, devilishly compelling game that could be played absolutely anywhere. You directed a tiny, calculator-quality, LED dash up a rectangular football field populated with other dashes that maneuvered to "tackle" you. If one of those dashes caught you, you were down. You can see a YouTube video of the game being played at bit.ly/1OEOUXp.

Like so many great innovations, no one got it. At first. According to an interview with Howard Cohen (bit.ly/111U1zP), one of the leaders of the Mattel Football project, the original production run of 500,000 units was halted after just 100,000 were made. Early sales figures from Sears, Roebuck, didn't support continued production. And then, sales took off. Before long, Mattel was pumping out a half million of the handheld game consoles each week.

Mattel Football was a classic case of savvy reuse. The game was powered by a Rockwell calculator chip modified to display the on-screen action. The players were rendered as red LED dashes instead of dots, Cohen said in his interview with the Handheld Games Museum, "because they are basically the little segments of the number 8 on a calculator display." Other quirks abounded. For instance, a limitation in the chip constrained the digital football field to a length of 90 yards. I'm pretty sure none of us ever noticed.

What Innovation Looks Like

In an age of touchscreen tablets and cloud apps it's hard to believe, but this is what innovation looks like. Working from the foundations of available technology and platforms, Mattel created something wholly new and exciting that both exceeded and reset expectations. For me, as a child of the 70s, there was life before Mattel Football and life after it.

Like so many great innovations, Mattel Football wasn't the first of its kind. A little-remembered handheld game called Mattel Auto Race preceded Football by a year. It proved out the repurposed calculator chip and display hardware used in the Mattel handhelds, and showed that Rockwell programmers could knock out a game experience in just 511 bytes of ROM. But it was Football that nailed the experience—and foretold the future. You know the universal head-down, hands-forward posture of commuters staring at their smartphones (bit.ly/1qAnV4g)? Mattel Football invented that.

Mark Lesser was the Rockwell developer who wrote both Auto Race and Football, as well as other, early handheld games. In a 2007 interview with Digital Press (the 30th anniversary of Football's release), he described working in "a primitive assembly language that was ad hoc to the specific chip being programmed" (bit.ly/1Ay7TBA). His take on what made Mattel Football so compelling should echo forward for anyone crafting modern experiences today.

"Even with the crudest graphics and sounds there are hooks that carry sufficient and enduring interest," Lesser said. "So much of the appeal of a game relates to the way it is tuned—the harmony between the elements. Complexity is not required to create fun gameplay."

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

© 2014 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.

DOMAINS | MAIL | HOSTING | eCOMMERCE | SERVERS

NEW HOSTING

POPULAR APPS - NOW EVEN BETTER!

WORKS WITH MySQL
php
Magento Open Source eCommerce
WORDPRESS
Joomla!
python
TYPO3
Drupal
PRESTASHOP The Best E-Commerce Experience
1&1

WordPress & Powerful App Platforms!

- Supports over 140 popular apps including WordPress, Drupal™, Joomla!™, TYPO3, and more
- Security & version upgrade notifications
- Trial version available for all applications
- App Expert Support
- 1&1 CDN powered by CloudFlare™ and 2 GB guaranteed RAM for peak performance

Powerful Tools

- PHP 5.5, Perl, Python, Ruby
- 1&1 Mobile Website Builder
- NetObjects Fusion® 2013 included

Successful Marketing

- Facebook® advertising credits
- Listing in business directories
- 1&1 Search Engine Optimization
- 1&1 E-Mail Marketing Manager

State-of-the-Art Technology

- Maximum availability (Geo-Redundancy)
- 300 Gbit/s network connection

All Inclusive

- **1 FREE** domain: .com, .net, .org, .biz, .info
- Unlimited Power: webspace, traffic, mail accounts, MySQL databases
- Secure e-mail addresses with virus and spam protection
- Linux or Windows operating system

COMPLETE PACKAGES FOR PROFESSIONALS

Starting at $0.99 per month*

1 TRIAL 30 DAY MONEY BACK GUARANTEE ★★★★★

1 MONTH FLEXIBLE PAYMENT OPTIONS ★★★★★

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

1 (877) 461-2631

1&1®

1and1.com

* Offer valid for a limited time only. The $0.99/month pricing for the 1&1 Unlimited Hosting Package applies during promotional period only. Regular price of $8.99/month applies after promotional period. Some features listed are only available with package upgrade. Visit www.1and1.com for full promotion details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. ©2014 1&1 Internet. All rights reserved. Rubik's Cube® used by permission of Rubik's Brand Ltd.

Effective Image Handling in Responsive Web Sites

Responsive Web design (RWD)—creating one unique Web experience through a fluid and adaptive Web layout—is undoubtedly one of those milestones that will change the flow of things in the real world.

As a Web developer, you must be ready to live side-by-side with a given fact and its exact opposite. On one hand, RWD is a de facto standard. It's becoming a must-have for all Web sites. On the other hand, RWD is still an in-progress methodology, with best practices still being defined and redefined. In this article, I aim to provide a quick snapshot of the state of RWD, focusing on the most critical aspects: overall performance and particular image handling.

RWD Performance

The primary purpose of RWD is to make content easy and enjoyable to consume on devices, regardless of size and shape. So far, so good, but how would you do that?

RWD requires quite a different approach to project approval and development. The canonical visual checkpoint when your customers look at static mockups and give you the green light is a much less defined step. Instead of static mockups, you'll likely present wireframes. Overall, a responsive site is easy to sell to customers, but it's challenging to explain why it ultimately turns out to be more expensive than having a classic non-responsive site.

A few years of experience have helped consolidate a few practices and a few frameworks such as Twitter Bootstrap have emerged to help with the responsive site implementation. But how about performance?

Performance is currently the sore point of RWD. A responsive site works beautifully when viewed through powerful desktop browsers. Their performance, though, tends to degrade when consumed via smaller devices such as smartphones. RWD isn't effective in the same way for just any type of Web site. A portal of news and videos is easier to reshape to make content usable on smaller screens than a site that has a lot of forms and requires interaction, such as a travel-booking site. In the latter case, the screen size is a more important factor.

The key question to ask a potential customer before starting on an RWD implementation is how much its business cares about the capabilities of the various devices. If devices serve a business purpose such as searching or purchasing products or services, then you can't just differentiate content based on plain screen width and orientation. A resized desktop browser window and a smartphone may be able to display content using the same layout, but they run on different hardware and have vastly different computing capabilities. If the business cares about the choice of devices, those differences should be taken into careful account before making any decisions.

The One-Site Metaphor

The "one-site" metaphor is at the foundation of RWD: There's one unique set of content pages, one back-end logic, one URL and one brand. There are two ways you can approach this type of RWD site development. For each view and regardless of the device, you serve the same HTML markup and make a bunch of different CSS files available to the device. Each CSS file is bound to a media query expression the browser evaluates at run time when the window size changes. The applied CSS might change on the fly, which changes the appearance of the content. This approach relies extensively on CSS3 media queries, a World Wide Web Consortium (W3C) standard defined at bit.ly/1oVBf89. Here's how you would define two CSS files to be applied at two different screen sizes:

```
<link type="text/css"
  rel="stylesheet"
  href="view480.css"
  media="only screen and (max-width: 480px)">

<link type="text/css"
  rel="stylesheet"
  href="view800.css"
  media="only screen and (max-width: 800px)">
```

The other option for developing responsive sites involves using CSS3 Media Queries with JavaScript code to appropriately reshape the Document Object Model (DOM). This approach is more powerful because it gives you more control over the layout. It also adds the opportunity to download extra content specific for the screen size. To detect changes of the screen size, you can use either the window resize event or the matchMedia event. The matchMedia event is a JavaScript implementation of media queries. When the browser detects a change in the specified media query, it fires a call to your code:

```
if (window.matchMedia) {
  mq800px = window.matchMedia("(min-width: 800px)"),
  mqPortr = window.matchMedia("(orientation: portrait)");

  mq800px.addListener(mq800px_handler);
  mqPortr.addListener(mqPortr_handler);
}
```

When the browser detects a portrait orientation or a minimum width of 800 pixels, the code will invoke your handlers. You can use JavaScript and CSS media queries within the same site. They apply to individual HTML views or pages.

So the key pattern behind RWD is one set of content with multiple views, whether you reshape that using CSS or some ad hoc JavaScript. If you go the CSS route, you're subject to what you can do with CSS in a Web page. You can move and reposition elements, you can reflow content within differently sized containers, and float blocks of content horizontally or vertically. You can also simply hide unwanted content.

GROUPDOCS
Your Document Collaboration APIs

30 days free trial

Your Document Collaboration APIs

Professional APIs that allow developers to empower their apps with document collaboration capabilities.

GroupDocs.Viewer

Native-text, high-fidelity HTML5 document viewer with support for over 50 file formats.

GroupDocs.Annotation

A powerful API that lets developers annotate Microsoft Office, PDF and other documents within their apps.

GroupDocs.Conversion

Universal document converter for fast conversion between more than 50 file formats.

GroupDocs.Signature

Electronic signature API that allows users to place signatures on documents within your apps.

GroupDocs.Comparison

A diff view API that allows end users to quickly find differences between two revisions of a document.

GroupDocs.Assembly

Merges data entered by users through online forms Into both Microsoft Office and PDF documents.

100% Standalone - No Office Automation

.NET Libraries | Java Libraries | Cloud APIs | Cloud Apps

SALES INQUIRIES
+1 214 329 9760

sales@groupdocs.com

groupdocs.com

With JavaScript, you can do all of that and then some. You can make more sophisticated changes to the view layout, create DOM subtrees from scratch or download new content from remote endpoints. With CSS, the DOM is the largest possible. With JavaScript, it can only grow to the extent of becoming the largest possible that was designed.

The size of RWD solutions isn't an issue when viewing sites on desktop browsers. It may not even be an issue on most tablets. Those tend to be powerful enough to run any required JavaScript and are often used through a solid Wi-Fi connection. The issue is having RWD sites on smartphones with 3G connections. In those cases, it's not so much the amount of markup and JavaScript code, it's the related events and their impact on threading that bogs things down. The most painful aspect of RWD is images.

Handling Images

The old img element references images. They're downloaded entirely and displayed as specified by width and height attributes. If you need a large background image for a desktop or numerous large images for a carousel, you can't simply use CSS to adjust width and height. That would have no impact on the download size. Hiding images via CSS doesn't help, either, as the image is downloaded anyway.

There are a few tricks you can apply while waiting for a new HTML element to be available. However, a new HTML element is a ways off in terms of widespread browser support. There are some experiments going, but they're not for prime time yet. The direction seems to be having an element whose structure blinks at the HTML5 structure of the video element, as follows:

```
<picture>
  <source media="(min-width: 400px)" srcset="foo-sm.jpg, foo-sm-2x.jpg 2x">
  <source media="(min-width: 800px)" srcset="foo-md.jpg, foo-md-2x.jpg 2x">
  <img src="foo.jpg">
</picture>
```

When using picture instead of img, you provide a range of options for the same logical image. You provide a set of images for each media query scenario and within each scenario you might provide multiple images to address different pixel densities. In that example, when the screen width is at least 800 pixels, the browser might choose a regular image with an appropriate size or a larger copy made-to-measure for a higher density or even to satisfy art direction requirements and provide a different image or a crop of the original image when a given media query kicks in. The embedded img element indicates the fallback and is logically equivalent to what is in use today. You can experiment with this approach using the JavaScript polyfill found at bit.ly/1aVEoxb.

Another approach is to use server-side logic, such as an HTTP handler. The handler will receive the request and decide which image to download. Frankly, this approach can be a challenge. The server-side logic needs some clues to select the most appropriate image. Those clues can only come from the browser and should be put into the HTTP request, whether with a query string or headers. This requires some scripting work to be done when the images are downloaded upon page loading. This is doable, but tricky.

While waiting for the picture element, image size and other aspects of the RWD specification to become standard and available on all browsers (only Chrome and Opera currently provide some support), stop and reflect on the exact problem you're trying to solve.

As long as the user is on a desktop browser, the size of the image isn't a big deal. Just point img to the largest one. If the browser window is resized, the same large image is resized. At that point, it no longer has an impact on performance. You want to change an image on smaller windows to focus on particular aspects. In that case, one trick might be to reference the image as the background of a div set via CSS instead of a plain img element. The advantage there is media queries will select just the image you want. You might get multiple images downloaded on the desktop, but that could be a negligible point. There could be other issues with background images if users try to print the page.

Using background images helps when the RWD site displays on mobile devices. Images are guaranteed to be of the proper size. Either way, you need to address the use of images in RWD when you plan to view the site on non-desktop devices.

How would you detect which type of device is being used? Detecting devices isn't a deadly sin. Reliably detecting devices is difficult, however, and can be a mess if you do it yourself. You can use a popular Modernizr plug-in for some forms of client-side device detection (bit.ly/1tfJhtf). This way, you can programmatically modify the DOM to try to get ad hoc images. That approach is reasonable, but doesn't scale with the number of devices and may become unreliable at some point. Device detection is a serious matter and requires expertise.

A Look at WURFL Image Tailor

One new interesting approach to image handling is the Wireless Universal Resource File (WURFL) Image Tailor (WIT) component. Backed by the full WURFL engine—the same device detection engine used by Facebook—WIT performs a quick server-side analysis of the user agent. It determines the form factor of the requesting device and serves a resized version of the original image. WIT is a free service that just requires a fix to the image URL:

```
<img src="//wit.wurfl.io/[full-url-to-your-image]">
```

You append the full image URL to the WIT Web site URL so you can download the image from the original Web site, resize the image and return pixels to the requesting site. Images are cached on the WIT end. That keeps the number of requests to the bare minimum. A bunch of supported parameters let you control aspects of the resizing such as cropping, dimensions and returned format.

WIT has pros and cons. On the upside, it relieves you of the burden of dealing with multiple versions of the same image. All it requires is a slightly modified version of the URL in the plain old img element. Plus, you can start using it in a matter of seconds.

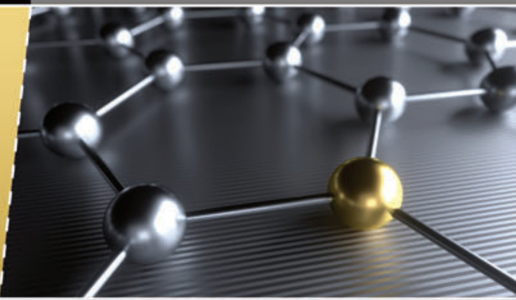
On the downside, it acts as a proxy and doesn't specifically address scenarios where you care also about pixel density and not just size. In any case, there's no reason for not giving it a try. You'll find it at wurfl.io/#wit.

The landscape of image handling within the context of RWD is fluid. Some compromise between RWD and devices will have to be forthcoming to ensure things work effectively everywhere. ■

Dino Esposito *is the co-author of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2014) and "Programming ASP.NET MVC 5" (Microsoft Press, 2014). A technical evangelist for the Microsoft .NET Framework and Android platforms at JetBrains and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents.wordpress.com and on Twitter at twitter.com/despos.*

Thanks to the following technical expert for reviewing this article:
Jon Arne Saeteras

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

CERTIFIED FOR Windows Server® 2008

Windows® 7

Windows 8

Windows Server 2012 Certified

AMYUNI

All development tools available at

www.amyuni.com

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

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

DATA POINTS

JULIE LERMAN

A Pattern for Sharing Data Across Domain-Driven Design Bounded Contexts, Part 2

In the October 2014 Data Points (msdn.microsoft.com/magazine/dn802601), I wrote about a pattern for mirroring data from one database to another when you're using multiple Domain-Driven Design (DDD) bounded contexts (BC), with each BC isolated to its own database. The scenario was that the Customer Management BC allows users to manage customer data, inserting, updating and deleting customer details. The second BC is for an ordering system that needs access to two critical pieces of customer information: the customer's identifier key and name. Because these systems are in two separate BCs, you can't reach from one into the other to share data.

DDD is for solving complex problems, and simplifying problems in the domain often means moving the complexity outside of the domain. So the Customer Management BC need not be aware of this subsequent sharing of data. In my previous column, I employed the publish/subscribe pattern to solve the problem by leveraging Domain Events, a message queue (RabbitMQ) and a service. There were a lot of moving parts.

Figure 1 **Persistence Class Raises Events After Data Is Persisted**

```
public class CustomerAggregateRepository {

public bool PersistNewCustomer(Customer customer) {
  using (var context = new CustomerAggregateContext()) {
    context.Customers.Add(customer);
    int response = context.SaveChanges();
    if (response > 0) {
      PublishCustomerPersistedEvent(customer, true);
      return true;
    }
    return false;
  }
}

public bool PersistChangeToCustomer(Customer customer) {
  using (var context = new CustomerAggregateContext()) {
    context.Customers.Attach(customer);
    context.Entry(customer).State = EntityState.Modified;

    int response = context.SaveChanges();
    if (response > 0) {
      PublishCustomerPersistedEvent(customer, false);
      return true;
    }
    return false;
  }
}

   private void PublishCustomerPersistedEvent(Customer customer, bool isNew) {
     CustomerDto dto = CustomerDto.Create(customer.Id, customer.Name);
     DomainEvents.Raise(new CustomerUpdatedEvent(dto, isNew));
   }
 }
```

Code download available at msdn.microsoft.com/magazine/msdnmag1214.

One shortcut I intentionally took to avoid overwhelming the explanation with too many concepts was that of triggering the series of events directly from the customer class by publishing the event to a message queue.

This month, I want to enhance the solution in two ways. First, I want to publish the message from a more logical spot in the workflow: after first confirming that the customer (whether new or modified) has been successfully persisted into the customer system's database. I also want to make sure the event is published only in response to relevant events. Publishing the event after a new customer is created makes sense. But I'd also like to tackle the hypothetical condition where I might need to be more discriminating about when updates are published to the message queue. In this case, I'd like to ensure the relevant message is published only when the customer's name has changed. If other data is modified that doesn't impact the customer name, I won't publish the message.

These two changes will make the solution more applicable to real-world scenarios.

Customer Created or Updated Becomes Customer Persisted

The current solution raises a notification when a customer is created or when the customer's name is fixed. The constructor and this FixName method both call PublishEvent:

```
public void FixName(string newName){
    Name = newName;
    ModifiedDate = DateTime.UtcNow;
    PublishEvent(false);
  }
```

PublishEvent triggers the workflow that results in the message being published to the queue:

```
private void PublishEvent(bool isNew){
    var dto = CustomerDto.Create(Id, Name);
    DomainEvents.Raise(new CustomerUpdatedEvent(dto, isNew));
  }
```

You can check the October column for details about that solution. Rather than raising the events from the class, I want to raise the events after I know either the new customer instance or the fix to the customer name has been successfully persisted into the database.

This means removing the PublishEvent method and the calls to it from the Customer class.

My data layer has a class containing the data access logic for the customer aggregate. I've moved the PublishEvent method into this class, renaming it to PublishCustomerPersistedEvent. In my methods

Working with Files?

Try Aspose File APIs

30 DAYS FREE TRIAL

Save Time and Effort

Creating, Converting, Combining Editing & Printing Files

Aspose.Words
DOC, DOCX, RTF, HTML, PDF...

Aspose.Email
MSG, EML, PST, EMLX...

Aspose.Pdf
PDF, XML, XLS-FO, HTML, BMP...

Aspose.BarCode
JPG, PNG, BMP, GIF, TIFF, WMF...

Aspose.Cells
XLS, XLSX, XLSM, XLTX, CSV...

Aspose.Imaging
PDF, BMP, JPG, GIF, TIFF, PNG..

Aspose.Slides
PPT, PPTX, POT, POTX, XPS...

Aspose.Tasks
XML, MPP, SVG, PDF, TIFF, PNG...

... and many more

100% Standalone - No Office Automation

Scan for 20% Savings!

ASPOSE
Your File Format APIs

US: +1 888 277 6734
sales@aspose.com

EU: +44 141 416 1112
sales.europe@aspose.com

AU: +61 2 8003 5926
sales.asiapacific@aspose.com

that persist customers to the database, I call the new event after SaveChanges is complete (see **Figure 1**).

With this move, I also needed to move the infrastructure for publishing messages into the data layer project. **Figure 2** shows the relevant projects (CustomerManagement.Core and CustomerManagement.Infastructure) I created for the previous column, alongside the projects after I moved this event into the data layer. The CustomerUpdatedEvent, DTO and Service are now in the Infrastructure project. It was satisfying to move infrastructure logic outside of the domain. I had been bothered by the code smell of needing it in the core domain.

I have two tests I use to verify that successful inserts and updates do, in fact, publish the correct messages into the queue. A third test verifies that a failed update doesn't attempt to publish any messages to the queue. You can see these tests in the download solution that accompanies this article.

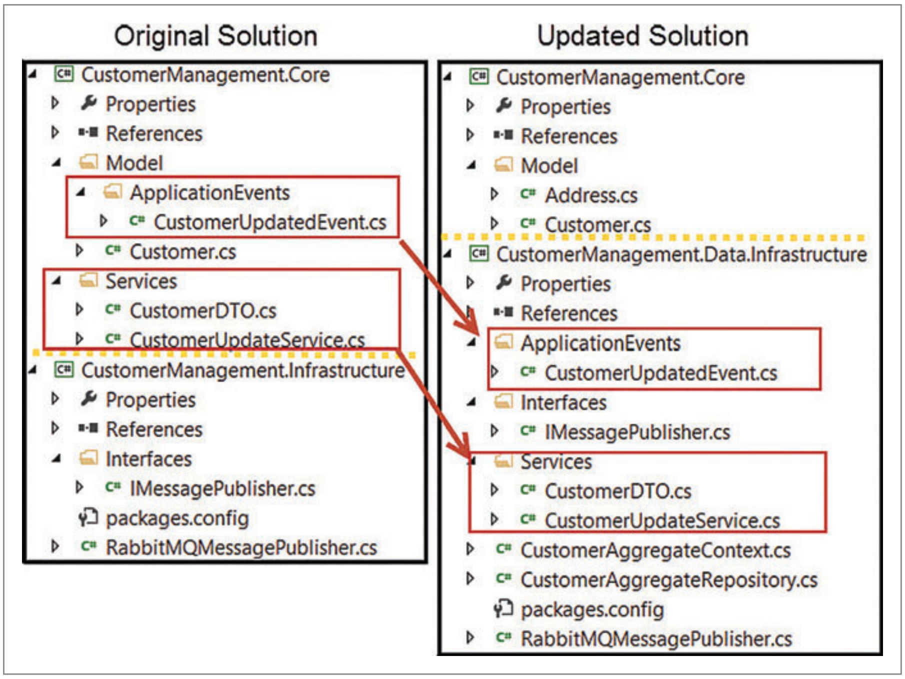


Figure 2 **Project Structure Before and After Moving Event Publishing into the Data Layer**

In the Data Layer, Not in the Domain Model

It's a really simple change, but I did struggle with this move—not the technical aspect—but in justifying moving the event out of my BC and into my data layer. I had a debate about this on a dog walk. (It is not abnormal for me to be walking through my woods, or up and down my road, talking to myself. Fortunately, I live in a quiet place where nobody will question my sanity.) The debate ended with the following conclusion: Publishing the event does not belong in the BC. Why? Because the BC cares only about itself. The BC does not care what other BCs or services or applications want or need. So, "I need to share my persisted data with the Ordering System," is not a valid concern of the BC. It isn't a domain event, but an event related to persistence, which I'll put in the Application Event bucket.

Fixing a New Problem Created by the Move

There is one issue caused by moving the publish event into my persistence layer. The PersistChangeToCustomer method is used to persist other edits to the Customer entity, as well. For example, the Customer entity also has the ability to add or update the customer's shipping and billing addresses. The addresses are value objects and creating or replacing them with a new set of values reflects a change to the customer.

I need to call PersistChangeToCustomer when either of these addresses change. But in that case, there's no point in sending a message to the queue saying that the customer name has changed.

So how do you let this persistence layer know the customer name didn't change? An instinctive solution is to add a flag property such as NameChanged. But I don't want to have to rely on adding Booleans as needed to track the detailed state. I considered raising an event from the customer class, but not one that would trigger another message to a queue. I don't want a message that just says, "Don't send a message." But how to capture the event?

Once again, Jimmy Bogard comes to the rescue with another brilliant solution. His May 2014 blog post, "A Better Domain Events Pattern" (bit.ly/1vUG3sV), suggests collecting events rather than raising them immediately, then letting the persistence layer grab that collection and handle the events as needed. The point of his pattern is to remove the static DomainEvents class, which doesn't allow you to control when events are raised and can therefore cause side effects. This is a newer line of thinking with regard to domain events. My refactoring will coincidentally avoid that problem, but I admittedly am still tied to the static DomainEvents class. As always, I will continue to learn and evolve my practices.

The BC does not care what other BCs or services or applications want or need.

I love Bogard's approach, but I'm going to steal the idea and use it a little differently than his implementation. I don't need to send a message to the queue; I only need to read the event. And it's a great way to capture this event in the customer object without creating various random state flags. For example, I can avoid the awkwardness of having to include a Boolean that says, "The name was fixed," to be set to true or false, as needed.

Bogard uses an ICollection<IDomainEvent> property called Events property in an IEntity interface. If I had more than one entity in my domain, I'd do the same or perhaps add it to an Entity base

Desktop. Web. Mobile. Your next great app starts here.

From interactive Desktop applications, to immersive Web and Mobile solutions, development tools built to meet your needs today and ensure your continued success tomorrow.

Download your free 30-day trial today and Experience the DevExpress Difference.

www.devexpress.com/try

DevExpress®

WIN ASP WPF SL VCL | XAF | CR

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

class. But in this demo, I'll just put the new property directly into my Customer object. I've created a private field and exposed the Events as read-only so only the Customer can modify the collection:

```
private readonly ICollection<IDomainEvent> _events;

public ICollection<IDomainEvent> Events {
  get { return _events.ToList().AsReadOnly(); }
}
```

Next, I'll define the relevant event: CustomerNameFixedEvent, which implements the IDomainEvent interface I used in Part 1 of this column. CustomerNameFixedEvent doesn't need much. It will set the DateTimeEventOccurred property that's part of the interface:

```
public class CustomerNameFixedEvent : IDomainEvent{

  public CustomerNameFixedEvent(){
    DateTimeEventOccurred = DateTime.Now;
  }

  public DateTime DateTimeEventOccurred { get; private set; }   }
}
```

Now, whenever I call the Customer.FixName method, I can add an instance of this event to the Events collection:

```
public void FixName(string newName){
  Name = newName;
  ModifiedDate = DateTime.UtcNow;
  _events.Add(new CustomerNameFixedEvent());
}
```

This gives me something that's much more loosely coupled than a state property. Additionally, I can add logic to it in the future as my domain evolves without modifying the schema of my Customer class. And I can take advantage of it in my persistence method.

The PersistChangeToCustomer method now has new logic. It will check for this event in the incoming Customer. If that event exists, it will fire off the message to the queue. **Figure 3** shows the full method again with the new bit of logic in it—a check for event type before publishing.

The method still returns a Boolean showing whether the Customer was successfully saved, indicated by the response to SaveChanges being greater than 0. If that's the case, then the method will check for any CustomerNameFixedEvents in the Customer.Events and publish the message about the customer being persisted as it did earlier. If for some reason the SaveChanges fails, most likely that will be indicated by an exception. However, I'm also looking at the value of response, so I can return false from the method. The calling logic will decide what to do about a failure—perhaps trying the save again or sending a notification to the end user or some other system.

Because I'm using Entity Framework (EF), it's worth noting that you can configure EF6 to retry SaveChanges on transient connection errors. But that will have already played out by the time I get a response from SaveChanges. Check my December 2013 article, "Entity Framework 6, Ninja Edition" (msdn.microsoft.com/magazine/dn532202), for more information about DbExecutionStrategy.

Figure 3 **PersistChangeToCustomer Method Checking Event Type**

```
public bool PersistChangeToCustomer(Customer customer) {
    using (var context = new CustomerAggregateContext()) {
      context.Customers.Attach(customer);
      context.Entry(customer).State = EntityState.Modified;

      int response = context.SaveChanges();
      if (response > 0) {
        if (customer.Events.OfType<CustomerNameFixedEvent>().Any()) {
          PublishCustomerPersistedEvent(customer, false);
        }
        return true;
      }
      return false;
    }
  }
```

I added a new test to verify the new logic. This test creates and persists a new customer, then edits the customer's BillingAddress property and persists that change. My queue receives a message that the new Customer was created, but it gets no message in response to the update that changes the address:

```
[TestMethod]
public void WillNotSendMessageToQueueOnSuccessfulCustomerAddressUpdate() {
  Customer customer = Customer.Create("George Jetson", "Friend Referral");
  var repo = new CustomerAggregateRepository();
  repo.PersistNewCustomer(customer);
  customer.CreateNewBillingAddress
    ("123 SkyPad Apartments", "", "Orbit City", "Orbit", "n/a", "");
  repo.PersistChangeToCustomer(customer);
  Assert.Inconclusive(@"Check status of RabbitMQ Manager for a create message,
    but no update message");
}
```

Stephen Bohlen, who reviewed this article, suggests the "test spy pattern" (xunitpatterns.com/TestSpy.html) as an alternative way of verifying that the messages did make it into the queue.

A Solution in Two Parts

How to share data across bounded contexts is a question many developers learning about DDD ask. Steve Smith and I hinted at this capability in our Pluralsight course, Domain-Driven Design Fundamentals (bit.ly/PS-DDD), but didn't demonstrate it. We've been asked numerous times to elaborate on how to pull this off. In the first article in this little series, I leveraged many tools to construct a workflow that would allow data stored in one BC's database to be shared with a database used by a different BC. Orchestrating a publish-subscribe pattern using message queues, events and an Inversion of Control container let me achieve the pattern in a very loosely coupled fashion.

This month, I expanded the sample in response to the question: When does it make sense, with a DDD focus, to publish the message? Originally, I triggered the data-sharing workflow by publishing the event from the Customer class as it created new customers or updated customer names. Because all of the mechanics were in place, it was easy in this article to move that logic to a repository, allowing me to delay publishing the message until I was sure the customer data had been successfully persisted into the database.

There's always more fine-tuning to be done, and perhaps different tools you might prefer to use. But you should now have a handle (no pun intended, but I have to leave it here now) on approaching the DDD pattern of allowing bounded contexts to work with their own independent databases. ■

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 Microsoft technical expert for reviewing this article: *Stephen Bohlen*

/n software

We didn't invent the Internet...

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

TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- E-Business
  AS2, 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 …

Microsoft .NET · Microsoft BizTalk Server · Microsoft SQL Server · Windows Mobile · activex · Microsoft SharePoint 2010 · CodeGear DELPHI · CodeGear C++BUILDER · Java · php

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

Microsoft .NET, | Java | ActiveX | C++ | Delphi | C++ Builder | PHP | Objective-C | Windows | Windows Mobile | Mac OS X | Linux/Unix | iPhone

Roaming Data Between Windows Store Apps and Windows Phone Store Apps

Tony Champion

There's an entirely different set of standards and expectations for today's apps than the desktop apps of a decade ago. One of these expectations is that apps work and share data across multiple devices. If a user has the same app installed on their desktop and laptop, they'll expect both apps to maintain the same configuration and work on the same set of data. Taking it a step further, if the same app is available on multiple devices, users will expect to be able to share data across devices.

In the world of data-driven apps, you can handle these expectations primarily on the back-end database. An app on multiple devices can query the same remote database, and give the user access to the same data. However, supporting apps with a remote database adds a tremendous overhead in terms of architecture, development and maintainability. It's simply not necessary for all apps. Even apps that require databases might not want to support app-specific information in a database designed for multiple purposes.

This article discusses:

- Sharing data between mobile apps
- Preparing data and devices to sync roaming data
- Determining the type of data you can sync

Technologies discussed:

Windows Runtime

Windows Store and Windows Phone Store apps address this need with the concept of roaming data. Every app automatically receives a small amount of cloud storage per user. The user can use this to save information about the app and share it across multiple installs of the app. Windows Phone Store apps and Windows Store apps take this another step and let the user share data between apps on different devices. Here, I'll look at the uses of roaming data in your apps and how to use this data effectively across devices.

This article will use universal apps to demonstrate sharing roaming data across devices, however, these same techniques work in separate Windows Phone Store and Windows Store apps. Roaming data also works just as well across HTML- and XAML-based apps.

Use Roaming Data

One of the nicer aspects of using roaming data in your apps is that it's automatically available and it requires no configuration or other settings. Simply use the roaming data API available to let your app take advantage of roaming data.

What to Roam The first question most developers ask is what type of data is good for roaming settings. Keep in mind there's a tight size limit on the amount of data that can roam. That means you need to do some advanced planning.

Some of the most common things to roam are user-defined preferences and settings—aspects such as colors, fonts, options and so on. These values might take on different meanings if you're sharing between Windows Store apps and Windows Phone Store apps, but providing a similar experience and between platforms will be a big win for your app.

Current navigation within your app can be a powerful feature. If a user pulls up a report on a Windows Phone and then logs into a computer at work, why shouldn't the Windows Store version of the app jump straight to the same report? If a user is watching a video on a desktop, why shouldn't the Windows Phone version jump to the same video?

Temporary data can be another candidate for roaming. If a user is in the middle of typing an e-mail on a laptop, they should be able to continue that same e-mail at their desktop.

Large datasets typically aren't candidates for roaming data. However, you could put a key to the data set in roaming data and pull the large dataset down to the new client based on the key you share.

The list goes on, but the goal is the same. Roaming data should let the user always feel connected to your app.

Enable Roaming Data There are a couple of prerequisites for your apps to successfully sync data between devices. First, users have to log on to the device using a Microsoft account. Roaming settings are associated with an app and a Microsoft user account. If the user isn't using a Microsoft account, the data is missing part of its key.

Figure 1 **Convert from ApplicationDataCompositeValue to Person Class**

```
public class Person
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public int Age { get; set; }

  public static Person FromComposite(ApplicationDataCompositeValue composite)
  {
    return new Person()
    {
      FirstName = (string)composite["firstName"],
      LastName = (string)composite["lastName"],
      Age = (int)composite["age"]
    };
  }

  public static ApplicationDataCompositeValue ToComposite(Person person)
  {
    var composite = new ApplicationDataCompositeValue();
    composite["firstName"] = person.FirstName;
    composite["lastName"] = person.LastName;
    composite["age"] = person.Age;

    return composite;
  }
}
```

Figure 2 **Obtain and Store Personal Information in RoamingSettings**

```
var person = new Person()
{
  FirstName = "Tony",
  LastName = "Champion",
  Age = 38
};

roamingSettings.Values["personalInfo"] = Person.ToComposite(person);

if (roamingSettings.Values.ContainsKey("personalInfo"))
{
  var composite =
    (ApplicationDataCompositeValue)roamingSettings.Values["personalInfo"];
  var roamingPerson = Person.FromComposite(composite);
}
```

Second, it's important the user hasn't disabled roaming data capabilities for the device. A user can do this manually or it might be a device policy applied by system administrators. If roaming data is disabled, the data won't sync.

If roaming data isn't enabled on a device, data is still available locally. Therefore, your app doesn't have to worry about checking to see if roaming data is syncing and using a different workflow.

> Some of the most common things to roam are user-defined preferences and settings—aspects such as colors, fonts, options and so on.

Explore the API

You'll find the API for roaming data in the Windows.Storage.ApplicationData object. Each app maintains a single instance of ApplicationData you can reference with the Current static property:

```
var appData = Windows.Storage.ApplicationData.Current;
```

The API doesn't include a mechanism to force roaming data to synchronize. That process is left to the device itself to manage.

You can use two types of roaming data. You'll find the first in the RoamingSettings property of ApplicationData, which is an ApplicationDataContainer that manages key/value pairs. The settings are managed in the RoamingSettings.Values property and you can access them as a string indexed array. The keys can be any alphanumeric string up to 255 characters long. The value can be an object as long as it's a supported Windows Runtime data type. This means you can't store custom objects in roaming settings.

You can access roaming settings through the indexed Values property. Add or update a setting by changing the key indexed Values property to the new value. Use the Values.Remove method to remove a setting. The following code shows an example of creating, reading and removing a roaming setting:

```
var roamingSettings = ApplicationData.Current.RoamingSettings;

// Create setting
roamingSettings.Values["setting1"] = "My First Setting";

// Read setting
var settings1 = roamingSettings.Values["setting1"];

// Remove setting
roamingSettings.Values.Remove("setting1");

// Clear all settings
roamingSettings.Values.Clear();
```

Storing simple Windows Runtime data types will work for some instances. However, there are times when storing an entire object makes sense. There are a couple of ways to store classes in roaming data, but you can use an ApplicationDataCompositeValue to store more complex objects in RoamingSettings.

An ApplicationDataCompositeValue is a collection of key/value pairs stored together. This is a great way to group items that will remain synced as a single unit. The following code shows an example of creating an ApplicationDataCompositeValue and adding it to RoamingSettings:

```
var compositeValue = new ApplicationDataCompositeValue();
compositeValue["firstName"] = "Tony";
compositeValue["lastName"] = "Champion";
compositeValue["age"] = 38;

roamingSettings.Values["personalInfo"] = compositeValue;
```

The one downfall to this approach is there's no mechanism to automatically go to and from a complex object to an ApplicationDataCompositeValue. One approach to resolve this is to create helper functions for your classes that will handle the conversion for you. **Figure 1** shows a Person class with two static methods—ToCompositeSetting and FromCompositeSetting. These methods convert the data stored in the previous example into a Person object, which will make things like data binding much less complex.

Figure 2 uses the new Person class to obtain and store the personal information in RoamingSettings.

There's a special key in the roaming settings you can use for data you need to sync immediately. Adding HighPriority to any setting will have it synced as quickly as possible. This is great for items like a current book page number, a paused video frame and anything else that will help provide a connected experience between devices. For example, if a user is watching a movie on your Windows Store app, you could provide the Windows Phone Store app with the supporting data appropriate to the user's current location in the movie:

```
var roamingSettings = ApplicationData.Current.RoamingSettings;

var composite = new ApplicationDataCompositeValue();
composite["movieId"] = myVidPlayer.MovieId;
composite["position"] = myVidPlayer.CurrentTime;

roamingSettings.Values["HighPriority"] = composite;
```

Files are the second type of roaming data. The ApplicationData object contains a RoamingFolder property that returns a StorageFolder instance where your app can read and write files to be synced.

Figure 3 **Update Version of Roaming Data**

```
void SetVersionHandler(SetVersionRequest request)
{
  SetVersionDeferral deferral = request.GetDeferral();

  if (request.CurrentVersion < 1)
  {
    // Handle upgrade from 0 to 1
  }

  if (request.CurrentVersion < 2)
  {
    // Handle upgrade from 1 to 2
  }

  deferral.Complete();
}

async void SetVersion()
{
  var appData = ApplicationData.Current;
  if (appData.Version < 2)
  {
    await appData.SetVersionAsync(
      2, new ApplicationDataSetVersionHandler(SetVersionHandler));
  }
}
```

You can add practically any type of file to the RoamingFolder. However, the file name must conform to certain specifications. First, the maximum file name and extension length is 256 characters. Also, the file name can't have leading spaces. There's also a group of Unicode characters that aren't allowed.

There are also several file types that aren't allowed because they act like folders, such as .zip and .cab. If you add a file to the RoamingFolder that doesn't meet these requirements, the file won't be synced, but will still be available for local access.

The API doesn't include a mechanism to force roaming data to synchronize. That process is left to the device itself to manage.

One use for the RoamingFolder is storing complex objects. Here's an example of taking the same Person object, serializing it and then writing it to a file in the RoamingFolder:

```
var roamingFolder = ApplicationData.Current.RoamingFolder;

// Create file and open a stream to write to
StorageFile personFile = await roamingFolder.CreateFileAsync(
  "personInfo.txt", CreationCollisionOption.ReplaceExisting);
var writeStream = await personFile.OpenStreamForWriteAsync();

// JSON serialize object
var serializer = new DataContractJsonSerializer(typeof(Person));
serializer.WriteObject(writeStream, person);

// Flush the stream
await writeStream.FlushAsync();
```

JSON serialization is used over XML for size considerations. With size constraints for roaming data in place, every byte counts.

You can use reverse logic to retrieve an object from the RoamingFolder. The following code demonstrates reading that same file and returning a Person object:

```
// Create file and open a stream to write to
var readStream = await roamingFolder.OpenStreamForReadAsync("personInfo.txt");

// JSON deserialize object
var serializer = new DataContractJsonSerializer(typeof(Person));
var roamingPerson = (Person)serializer.ReadObject(readStream);
```

Now you can read and write roaming data to be synced between devices, but how will you know when it has been updated? This is handled by the DataChanged event in ApplicationData. Any time a device receives new roaming data, the DataChanged event will fire, passing in the updated ApplicationData object. This lets you make any adjustments to your app when data has changed. There's no corresponding event to let you know when data has been pushed from the device. The following code demonstrates how to listen to the DataChanged event:

```
private void HandleEvents()
{
  ApplicationData.Current.DataChanged += Current_DataChanged;
}

void Current_DataChanged(ApplicationData sender, object args)
{
  // Update app with new settings
}
```

Extreme Performance Linear Scalability

For .NET & Java Apps

(Microsoft Azure Supported)

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
- NHibernate & Entity Framework Second Level Cache

FREE EXPRESS EDITION

ASP.NET Optimization in Web Farms

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

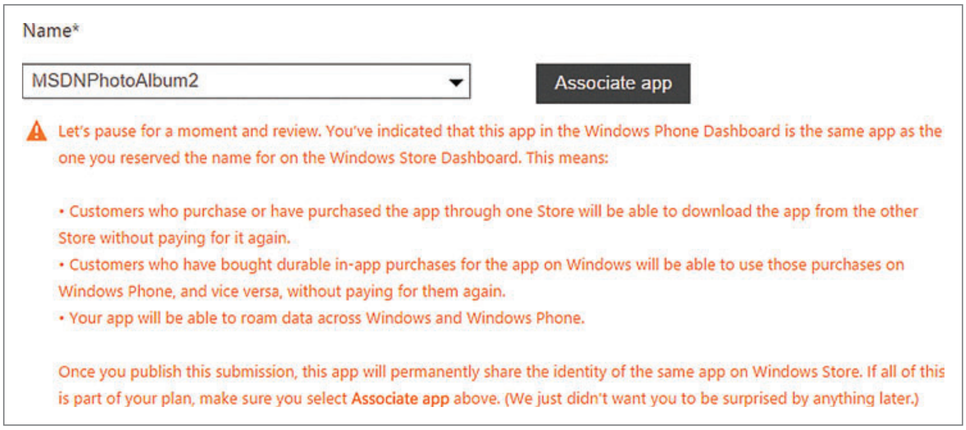
Runtime Data Sharing

- Powerful event notifications for pub/sub data sharing
- Continuous Query events

NCache™

Download a FREE trial!

sales@alachisoft.com US: +1 (925) 236 3830 www.alachisoft.com


Name*

MSDNPhotoAlbum2 ▾

⚠ Let's pause for a moment and review. You've indicated that this app in the Windows Phone Dashboard is the same app as the one you reserved the name for on the Windows Store Dashboard. This means:

- Customers who purchase or have purchased the app through one Store will be able to download the app from the other Store without paying for it again.
- Customers who have bought durable in-app purchases for the app on Windows will be able to use those purchases on Windows Phone, and vice versa, without paying for them again.
- Your app will be able to roam data across Windows and Windows Phone.

Once you publish this submission, this app will permanently share the identity of the same app on Windows Store. If all of this is part of your plan, make sure you select **Associate app** above. (We just didn't want you to be surprised by anything later.)

Figure 4 **Link a Windows Phone Store App to a Windows Store App**

If your app is taking advantage of the DataChanged event, the SignalDataChanged method in ApplicationData is useful. Any time you update any roaming data locally, you can call that method and it will fire the DataChanged event and allow any update handlers you need to run:

```
// Update settings locally and raise DataChanged event
roamingSettings.Values["favoriteColor"] = "Black";
ApplicationData.Current.SignalDataChanged();
```

There's a special key in the roaming settings you can use for data you need to sync immediately.

It's important to keep track of the amount of data you're attempting to roam. Each device has a maximum amount of data that can sync between devices, which is currently 100KB. The potential issue is it's an all-or-nothing approach. If the total amount of data you're attempting to sync exceeds the limit, then nothing will sync between the devices. The ApplicationData class contains a RoamingStorageQuota property that returns the total size of data allowed to be synced in kilobytes. However, ApplicationData doesn't contain any mechanism for determining the current amount of data you're using, so for now it's up to you to keep track.

New versions of your app could mean new or changed settings from previous versions, as well as data from previous versions no longer being needed. To address this issue, the Windows Runtime lets you version your roaming data. You'll find the current roaming data version to which the app is set in the ApplicationData.Current.Version. This version number is independent of the app version number and by default is set to zero. The app will sync data that matches its version number. This lets you create new data structures without fear of breaking older versions.

You can change the app version through the ApplicationData.SetVersionAsync method. This method has two parameters, the new version number and an ApplicationDataSetVersionHandler, to let you write any code necessary to make changes to the app based on the new version.

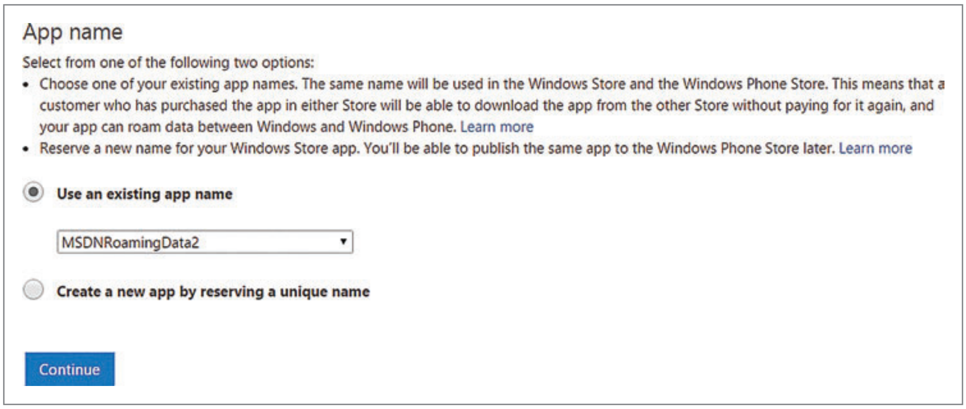
The handler contains a single SetVersionRequest parameter. This provides the current version through the CurrentVersion property, as well as the new version in the DesiredVersion property. Your app can use these two values to handle any migrations on an iterative approach. It also contains a GetDeferralMethod that lets you hold the thread open until you have the opportunity to complete the migration. That way if you have any async calls such as reading or writing files, you can perform those functions before the version change process is complete. **Figure 3** shows how to migrate to a new version.

Share Between Windows Store and Windows Phone Store Apps

Now that you've implemented roaming settings in both your Windows Store and Windows Phone Store apps, the next logical step is for the two companion apps to be able to share their roaming data.

Technically speaking, in order for a Windows Phone Store app and a Windows Store app to share roaming data, they must have the same Package Family Name. This field is generated based on the Package Name. Therefore, the two apps must have the same Package Name. You can find these values by associating the apps with names in their respective stores.

The first step to submitting an app in either store is to assign the app a name. You can either create a new name or associate the app with an existing name. The list of existing names is a combination of


App name

Select from one of the following two options:

- Choose one of your existing app names. The same name will be used in the Windows Store and the Windows Phone Store. This means that a customer who has purchased the app in either Store will be able to download the app from the other Store without paying for it again, and your app can roam data between Windows and Windows Phone. Learn more
- Reserve a new name for your Windows Store app. You'll be able to publish the same app to the Windows Phone Store later. Learn more

● **Use an existing app name**

MSDNRoamingData2 ▾

○ **Create a new app by reserving a unique name**

Figure 5 **Link a Windows Store App to a Windows Phone Store App**

reserved names for the store you're in and a list of apps you have in the other store. Selecting an app in the other store will link your app, giving it the same name and letting the apps share roaming data. You can read more about what it means to link a Windows Store and Windows Phone Store app at bit.ly/10PI2Xi.

Because the stores are still separate, the experience of linking your apps is slightly different. If you have an existing Windows Store app, you can link a Windows Phone Store app by selecting the Windows Store app in the App info section, as shown in **Figure 4**. You can also link a Windows Store app to an existing Windows Phone Store app up in the App name section, as shown in **Figure 5**.

Once you've created and linked the apps in the stores, the final step is to associate each of your apps in Visual Studio to those names. Right-click in your primary Windows Store and Windows Phone Store solution and select Store | Associate App with the Store. Then follow the wizard and select the correct name. This will update your Package.appxmanifest with the information entered in the store.

At this point, your apps will be able to share roaming data. Remember, it's important to track the size of the data you're roaming. If the storage quotas are different across the platforms, you'll need to plan for the smaller of the two as your limit.

Debug Roaming Data

Testing your roaming settings is straightforward. Again, it's an all-or-nothing situation. Either your roaming data is synced between devices or it isn't. When testing the syncing process, locking the device will force the app to attempt to synchronize its data. If this doesn't happen, there are a couple of things to consider:

- The most common cause of data not syncing is the app has exceeded the roaming storage quota. If the total size exceeds the limit, the app won't attempt to sync its roaming data.
- If you're storing your data in files, make sure you've closed all file handlers. Leaving files opened will maintain a lock on them and also prevent the synchronization.

Wrapping Up

Using roaming data in your apps provides your users a consistent and always-connected experience. Sharing settings, preferences, and current app state on one device and carrying it over to another device lets the users feel like it's the same app no matter the device. Adding the ability to share data between Windows Store and Windows Phone Store apps amplifies that experience and opens up a wide range of opportunities. ■

Tony Champion, *a Microsoft MVP, is the president of Champion DS and is active in the community as a speaker, blogger and author. He maintains a blog at tonychampion.net. Reach him at tony@tonychampion.net.*

Thanks to the following Microsoft technical expert for reviewing this article: *Robert Green*

...HOSTED ENVIRONMENT ... ≡BLAH BLAH≡ ... ZERO DEPLOYMENT ... FEATURE...
WOW... HOSTED APPS... IS HE AWARE OF OUR DESKTOP-ONLY ARCHITECTURE?
?
C.E.O.
DEV. LEAD
PROG. MGR.
≡BLAH BLAH≡ LESS ENERGY ... THE ...NEED TO MANAGE HARDWARE ≡BLAH≡ ... RESOURCES...
ERRHH... NOT SURE, IT'S PROBABLY A 3-YEAR PLAN OR SO...
C.E.O.
DEV. LEAD
PROG. MGR.
...WE'RE PLEASED TO ANNOUNCE A CLOUD VERSION IN THE NEXT ...
...FEW WEEKS!
"WE"?
BRAVO!
CLAP! CLAP!
CLAP! APPLAUSE!!

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.

*"I recently spent a week attending a course on Entity Framework but CodeFluent Entities provides so much more and is decidedly easier to understand and implement" **

Peter Stanford - Artefaction - Australia

* *Source : http://visualstudiogallery.msdn.microsoft.com/B6299BBF-1EF1-436D-B618-66E8C16AB410*

To get a license worth $399 for free
Go to www.softfluent.com/forms/msdn-2014

CodeFluent Entities
tools for developers, by developers

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

Developing Your First Game with Unity and C#, Part 4

Adam Tuliper

Welcome to the final article in my series on Unity game development. Remember when the app market was first exploding? Everyone was jumping on the bandwagon, and there was an app for everything. With a gazillion apps, though, came the trouble people had finding *your* app. Having apps get lost in the marketplace is a very real problem for developers. The Windows Store/Windows Phone Store isn't the dominant marketplace on the planet. Does that mean you shouldn't develop for it? Absolutely not. Developers have a real opportunity here to have their applications found. I hear of developers having their apps featured often. I know exactly three people who have had this happen on other platforms, and I talk to a lot of developers from all platforms.

In this article, I'll look at the process of taking a game developed in Unity to Windows. The process is pretty straightforward and allows you to do some pretty cool platform integration that's relatively easy to implement in a game. There's some excellent tooling available as of Unity 4.5, which supports the new Universal Projects (solutions that generate packages for Windows Phone and Windows Store with shared code), as well.

As you work, keep this popular saying in mind, "Test early, test often." You're not likely to hear a better software development mantra, and it holds very true in game dev. I'd like to revise it slightly, though: "Test early on devices, test often on devices." No matter what the platform, you'll find devices might act different than expected.

This article discusses:

- Building for Windows
- Using platform-specific code
- Build options
- Player Settings

Technologies discussed:

Unity, C#, Microsoft .NET, Mono

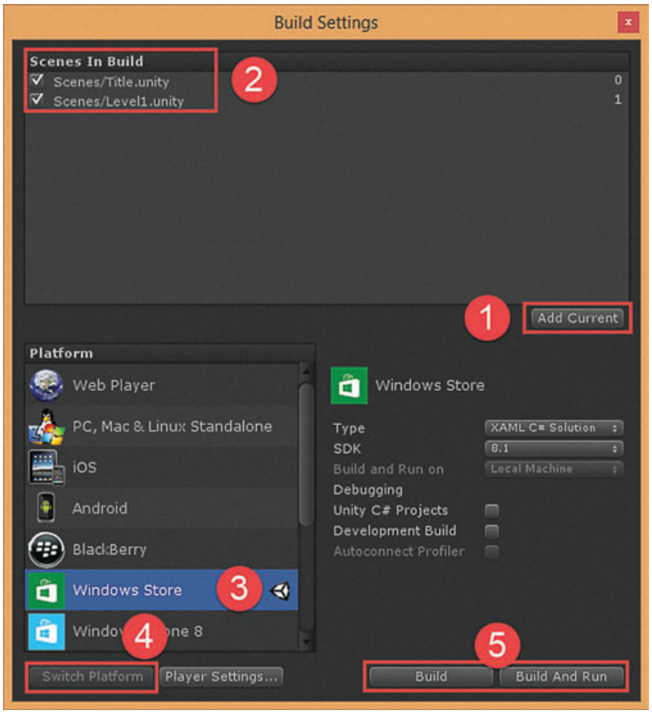
The Platform

If you're reading this magazine, you likely have an idea of the Windows ecosystem. I'll just do a quick review of what Unity supports in that ecosystem. You can target Xbox 360, Xbox ONE, Windows Phone 8/8.1, Windows 8/8.1 (Windows Store apps) and the desktop. You can choose all of these targets in the free version of Unity except Xbox. You'll see Xbox listed in the build options, but you can't build to it unless you're in the ID program for Xbox ONE. For Xbox 360, you must sign up via an approved publisher. If you're working on something super cool, please check out the ID program at xbox.com/Developers/id. Windows Phone and Windows Store have very similar build processes.

Remember, Windows Phone and Windows Store have the following build options: Windows Phone 8 via Silverlight; Windows Phone 8.1 via Silverlight or the Windows Runtime (WinRT); Windows 8; Windows 8.1; universal apps that target Windows Phone 8.1 and Windows 8.1.

Building for Windows

Making a build from Unity is actually quite simple. For local testing, it's just a matter of bringing up the build dialog (File | Build Settings), clicking Build, and choosing an output folder. The Build and Run option launches your game after compilation on either a connected phone or your local system, depending on what you choose. That's it at a basic level. After a successful build, the folder you chose will open with either an executable or a Visual Studio solution. Errors will be shown in the console window, so it's always a good idea to have it open via the window menu. If you don't have the console window open, you have to remember to look at the very bottom



Build Settings

Scenes In Build

☑ Scenes/Title.unity 0

☑ Scenes/Level1.unity 1

Platform

Windows Store

Type: XAML C# Solution

SDK: 8.1

Build and Run on: Local Machine

Debugging

Unity C# Projects □

Development Build □

Autoconnect Profiler □

Figure 1 **Adding Scenes to a Build**

status bar of the Editor window for a single line of red messages. It's a bit hidden, but once you know it's there, you won't forget it.

For All Builds Unity runs your build with what it calls a player and it supports the players of all the different platforms noted previously. When you create a build of your game, you'll need to add every scene you want in the build. To load the various scenes in your game (outside of the first one, which loads by default), you use Application.LoadLevel, which takes either a scene name such as Application.LoadLevel("Level1") or an index such as Application.LoadLevel(2). I'm not a fan of using the index method because, as you'll see, scene ordering can easily vary.

Every level you want to load in code must be added to your build. In the Build settings dialog, you add whatever scenes you want in the build via the "Add Current" button or by dragging and dropping scene files onto the build dialog. Here, you can reorder them, as well (which, again, makes loading scenes by index dangerous because they can easily get reordered). You can enable or disable these scenes for any build by checking or unchecking them. As **Figure 1** shows, the steps are as follows:

1. Add scenes to the build.
2. Ensure the desired scenes are checked. I often check a test scene to include in a local build and uncheck it for the final build.
3. Highlight the platform for which you want to build.
4. Click Switch platform to trigger Unity to prepare your assets for the selected platform and enable platform-specific preprocessor constants, such as UNITY_METRO, UNITY_IPHONE, UNITY_WEBPLAYER, and so forth.
5. Build your game into a platform-specific package.

For Windows Desktop The standalone (desktop) build is the default when you open Unity. With this build, Unity uses Mono to compile your game assemblies and packages them with the runtime engine. You get an executable file and a folder into which everything is packaged. This build is straightforward, as are most desktop builds.

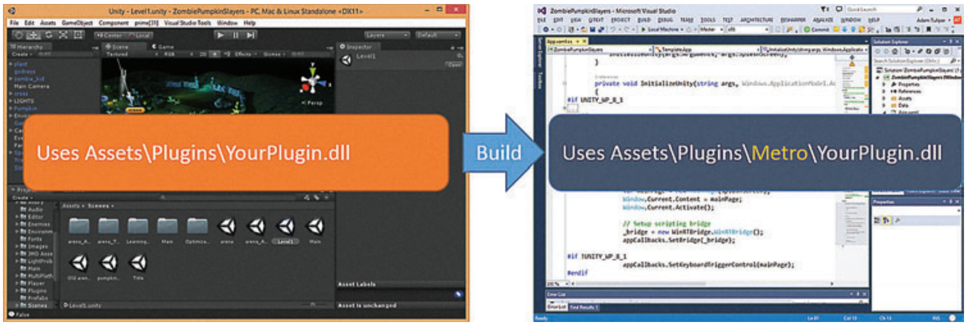
For Windows Store and Windows Phone This is where the fun begins, and I mean it. You can do some really cool platform integration when exporting to Windows Store and Windows Phone. You can use a live tile to entice users to come back to your game for free game credits, send a text message, use geofencing and more. The main thing to note is that it's relatively easy to integrate platform-specific features into your game.

The build process differs a bit from the desktop build, although you can still do a quick Build and Run from the dialog and get a running Windows Store or Windows Phone app pretty quickly for testing.

To understand where platform-specific code can fit into your Unity game, look at how Unity compiles your code. The code that runs in the Unity Editor is compiled by the licensed version of Mono. This means you can't make WinRT (Windows Store/Windows Phone 8.1) API calls like GeoLocation while running your game in the Editor, as there are specific WinRT methods that don't exist in Mono. Likewise, there are features in Mono that don't exist in the Windows Runtime, such as Hashtable and traditional .NET file access via System.IO. There are many overlaps between Mono and the Windows Runtime, but they're different APIs, so expect some differences.

When you build your project for the Windows Store, Unity uses the Microsoft .NET Framework to compile the game assemblies. By default it uses the .NET Core to compile your assemblies, which is essentially what WinRT .NET is. You simply need a way to tell Unity to compile your platform-specific code only when compiling for that particular platform.

There are two main ways of using platform-specific code here. (There's a third technique called a bridge, which wires up actions between Unity and Visual Studio solution code, but it's used far less often and won't be covered here.) The first technique is to use a plug-in. You can write your own, which is fairly trivial, or download one from several good sources. As of this writing, prime31.com plug-ins for Windows Store and Windows Phone are free. Virtually every major publisher using Unity uses plug-ins in its game projects. Get these while you can, it's a pretty amazing deal.



```
        private void InitializeUnity(string args, Windows.ApplicationModel.Ac
        {
#if UNITY_WP_8_1

            Window.Current.Content = mainPage;
            Window.Current.Activate();

            // Setup scripting bridge
            _bridge = new WinRTBridge.WinRTBridge();
            appCallbacks.SetBridge(_bridge);

#if !UNITY_WP_8_1
            appCallbacks.SetKeyboardTriggerControl(mainPage);
#endif
```

Uses Assets\Plugins\YourPlugin.dll
Build
Uses Assets\Plugins\Metro\YourPlugin.dll
ZombiePumpkinSlayers - Microsoft Visual Studio

Figure 2 **Building with Platform-Specific Code via a Plug-In**

Figure 3 **Prime31 Plug-in Code**

```
public class WindowsStoreAppSettings : MonoBehaviour
{
  private void Start()
  {
#if UNITY_METRO // A preprocessor directive to run when Windows Store
                // build is selected in Unity.

  SettingsPane.registerSettingsCommand("Privacy Policy",
    "This is my privacy policy. Consider a url as well to bring you to the site.");

  // Set the live tile. You can set a source as an Internet URL, as well.
  // This is not the Unity assets folder.
  // You'll want to make sure this image appears in the
  // generated Visual Studio project because it won't get
  // pushed from Unity in any way.
  var images = new string[] { "ms-appx:///assets/WideLiveTile.png" };
  Tiles.updateTile(TileTemplateType.TileWideImage, null, images);
#endif
  }
}
```

Using Platform-Specific Code

Via Plug-Ins The plug-in model in Unity uses two versions of a library with the same method signatures. One is essentially a stub version of your library compiled against the .NET Framework 3.5, which is placed in Assets\Plugins\YourPlugin.dll. That's the version Unity uses when your game is running in the Editor. The other one is compiled for the platform you're targeting—say Windows Store or Windows Phone—and is packaged into your game when you create a build from Unity, as shown in **Figure 2**.

The platform-specific library is placed in Assets\Plugins\<Platform>\YourPlugin.dll (the platform can be Windows 8.x, iOS, Windows Phone 8 and so on). If you have a reason to create one yourself as opposed to downloading one of the many already available, check out bit.ly/1EuLV1N for the basic directions. The primary difference in the plug-ins (outside of the platform API, of course) is where you put the platform-specific dll. You can find the plug-in locations at bit.ly/1v2ml1m.

Figure 4 **Using Geolocation**

```
public class FindLandTarget : MonoBehaviour
{

// If you're compiling for the Windows Runtime, use this code
// to GeoLocate on object collision.
#if NETFX_CORE
  void OnCollisionEnter(Collision collision)
  {
    Debug.Log("Collided with " + collision.gameObject.name);
    GetGeoLocation();
  }

  async void GetGeoLocation()
  {
    // Must call geolocation on the UI thread, there's a UI piece to be shown.
    UnityEngine.WSA.Application.InvokeOnUIThread(
      async () =>
        {
        var locator = new Windows.Devices.Geolocation.Geolocator();
        var position = await locator.GetGeopositionAsync();
        Debug.Log(string.Format("{0}, {1}  Accuracy: {2}",
          position.Coordinate.Latitude, position.Coordinate.Longitude,
          position.Coordinate. Accuracy));
        }, false
            );
    /**/
  }
#else // When you're not using the Windows Runtime, use this collision code instead.
  void OnCollisionEnter(Collision collision)
  {
    // Non-Windows Runtime platforms, no geolocation.
    Debug.Log("Collided with " + collision.gameObject.name);

  }
#endif
}
```

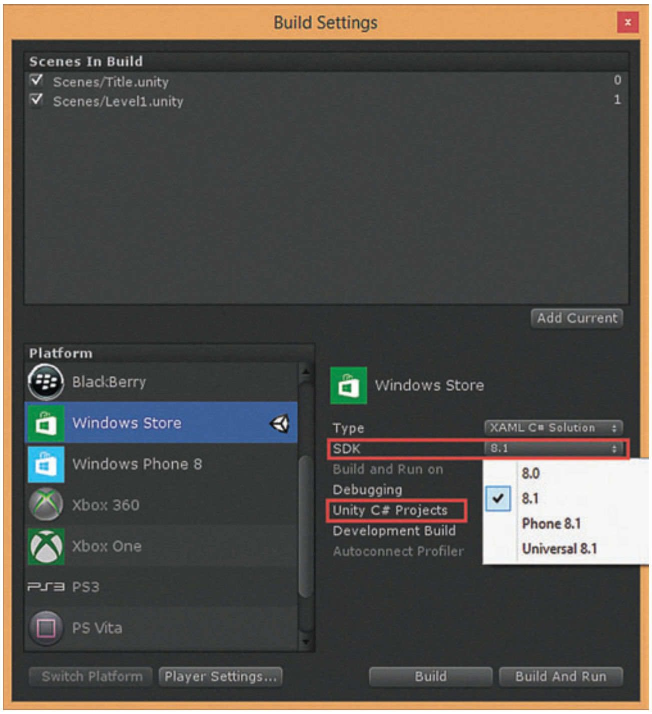
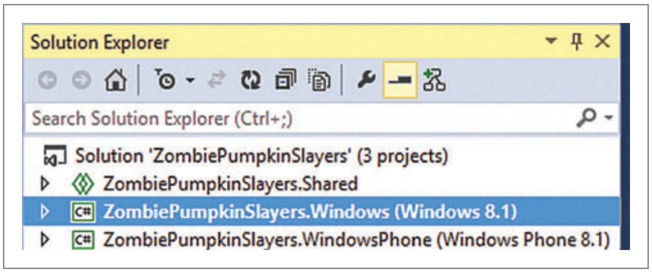


Figure 5 **Building for Windows Store**

Using one of the prime31 plug-ins here to set a live tile is as simple as attaching the code in **Figure 3** to any GameObject in your scene. The SettingsPane and Tiles class are plug-ins that contain functionality to implement platform-specific code.

Via Preprocessor Directives You can also use preprocessor directives to enable or disable compilation of inline code. This is a common technique when sharing code among platforms across a variety of technologies and is also used very often in Unity games. With this method, you simply use a directive in your code class. Preprocessor directives can be used for various purposes. The important rules to remember are: Separate your code by platform with a platform-specific preprocessor directive; some preprocessor directives become active the moment you switch platforms in Unity build settings (like UNITY_METRO); some become active only upon compilation outside of the editor (NETFX_CORE); others may be available all the time (such as the check for Unity version, UNITY_4_5, and other custom directives you define).



Solution 'ZombiePumpkinSlayers' (3 projects)
ZombiePumpkinSlayers.Shared
ZombiePumpkinSlayers.Windows (Windows 8.1)
ZombiePumpkinSlayers.WindowsPhone (Windows Phone 8.1)

Figure 6 **A Universal App with a Shared Project**

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

OCR Barcode PDF Forms Recognition & Processing SVG

DICOM PACS Client & Server HTML5 Medical 3D HL7

Annotation Viewers Cross-platform TWAIN Image Processing

Raster | Vector | Office Formats Virtual Printer Distributed Computing

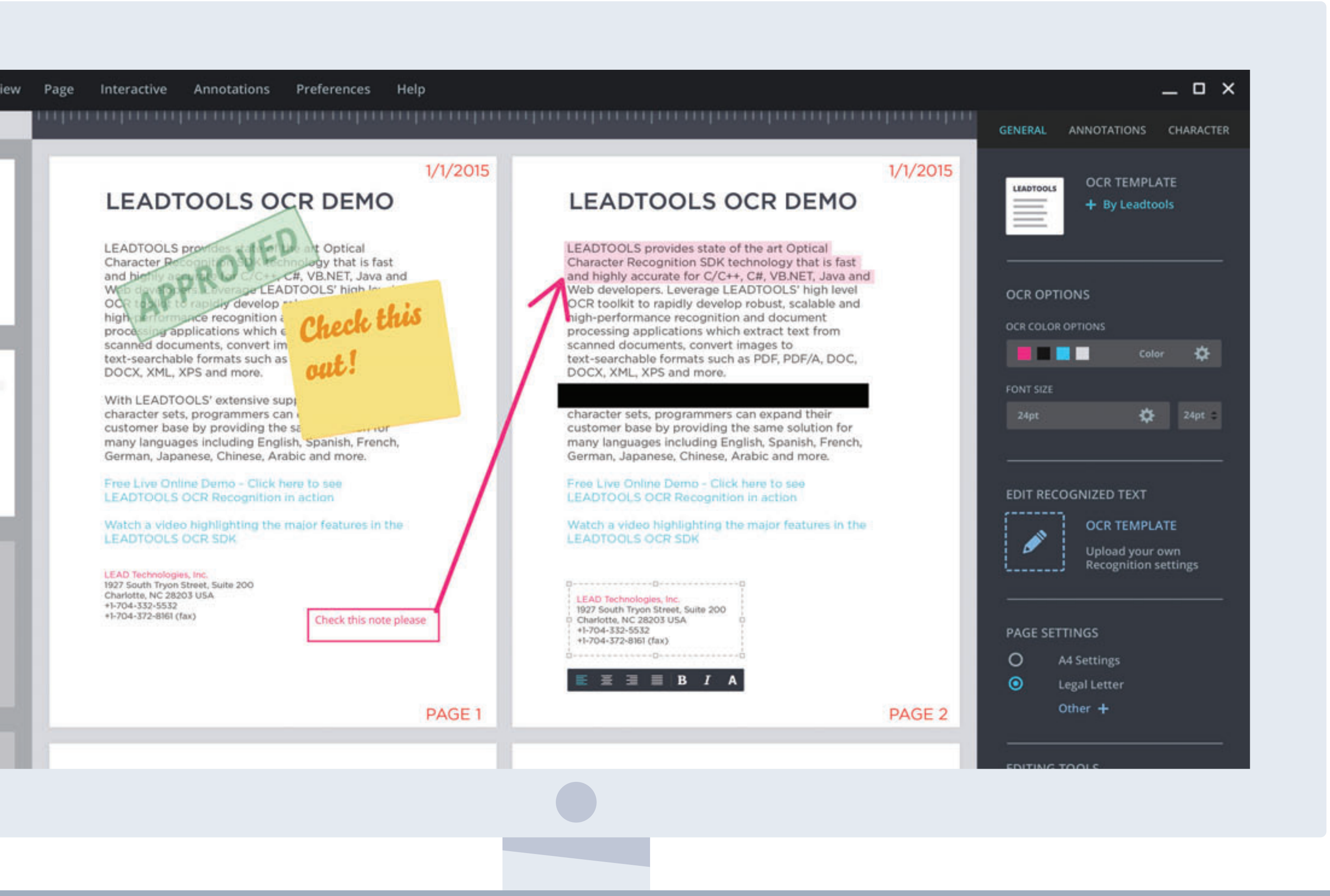
MPEG2-TS DVR RTSP Codecs DVD Streaming

LEADTOOLS
Document Imaging

LEADTOOLS Document Imaging technology has been trusted by application developers for over two decades to deliver powerful document imaging features required by financial institutions, government agencies, corporate offices, and any businesses moving towards automated or paperless environments.

Features include:

- OCR, MICR, OMR and ICR (handwritten)
- Structured and unstructured forms recognition and processing with OEM-ready forms applications (checks, passports, driver licenses, invoices)
- Barcodes auto detection, reading and writing
- PDF and office format viewing, editing and conversion: convert hundreds of different file types to PDF, DOC/DOCX, RTF, HTML, XPS
- PDF SDK powerful features: page manipulation, optimize PDF, bookmarks and metadata, create PDF/A files, PDF annotations, image extraction and more



1/1/2015

LEADTOOLS OCR DEMO

LEADTOOLS provides state of the art Optical Character Recognition SDK technology that is fast and highly accurate for C/C++, C#, VB.NET, Java and Web developers. Leverage LEADTOOLS' high le
OCR toolkit to rapidly develop
high-performance recognition a
processing applications which e
scanned documents, convert im
text-searchable formats such as
DOCX, XML, XPS and more.

With LEADTOOLS' extensive supp
character sets, programmers can
customer base by providing the sa
for
many languages including English, Spanish, French, German, Japanese, Chinese, Arabic and more.

Free Live Online Demo - Click here to see LEADTOOLS OCR Recognition in action

Watch a video highlighting the major features in the LEADTOOLS OCR SDK

LEAD Technologies, Inc.
1927 South Tryon Street, Suite 200
Charlotte, NC 28203 USA
+1-704-332-5532
+1-704-372-8161 (fax)

1/1/2015

LEADTOOLS OCR DEMO

LEADTOOLS provides state of the art Optical Character Recognition SDK technology that is fast and highly accurate for C/C++, C#, VB.NET, Java and Web developers. Leverage LEADTOOLS' high level OCR toolkit to rapidly develop robust, scalable and high-performance recognition and document processing applications which extract text from scanned documents, convert images to text-searchable formats such as PDF, PDF/A, DOC, DOCX, XML, XPS and more.

[redacted]
character sets, programmers can expand their customer base by providing the same solution for many languages including English, Spanish, French, German, Japanese, Chinese, Arabic and more.

Free Live Online Demo - Click here to see LEADTOOLS OCR Recognition in action

Watch a video highlighting the major features in the LEADTOOLS OCR SDK

LEAD Technologies, Inc.
1927 South Tryon Street, Suite 200
Charlotte, NC 28203 USA
+1-704-332-5532
+1-704-372-8161 (fax)

B I A

OCR TEMPLATE
+ By Leadtools

OCR OPTIONS

OCR COLOR OPTIONS
🟥 ■ 🟦 □ Color

FONT SIZE
24pt 24pt

EDIT RECOGNIZED TEXT
OCR TEMPLATE
Upload your own Recognition settings

PAGE SETTINGS
○ A4 Settings
● Legal Letter
Other +

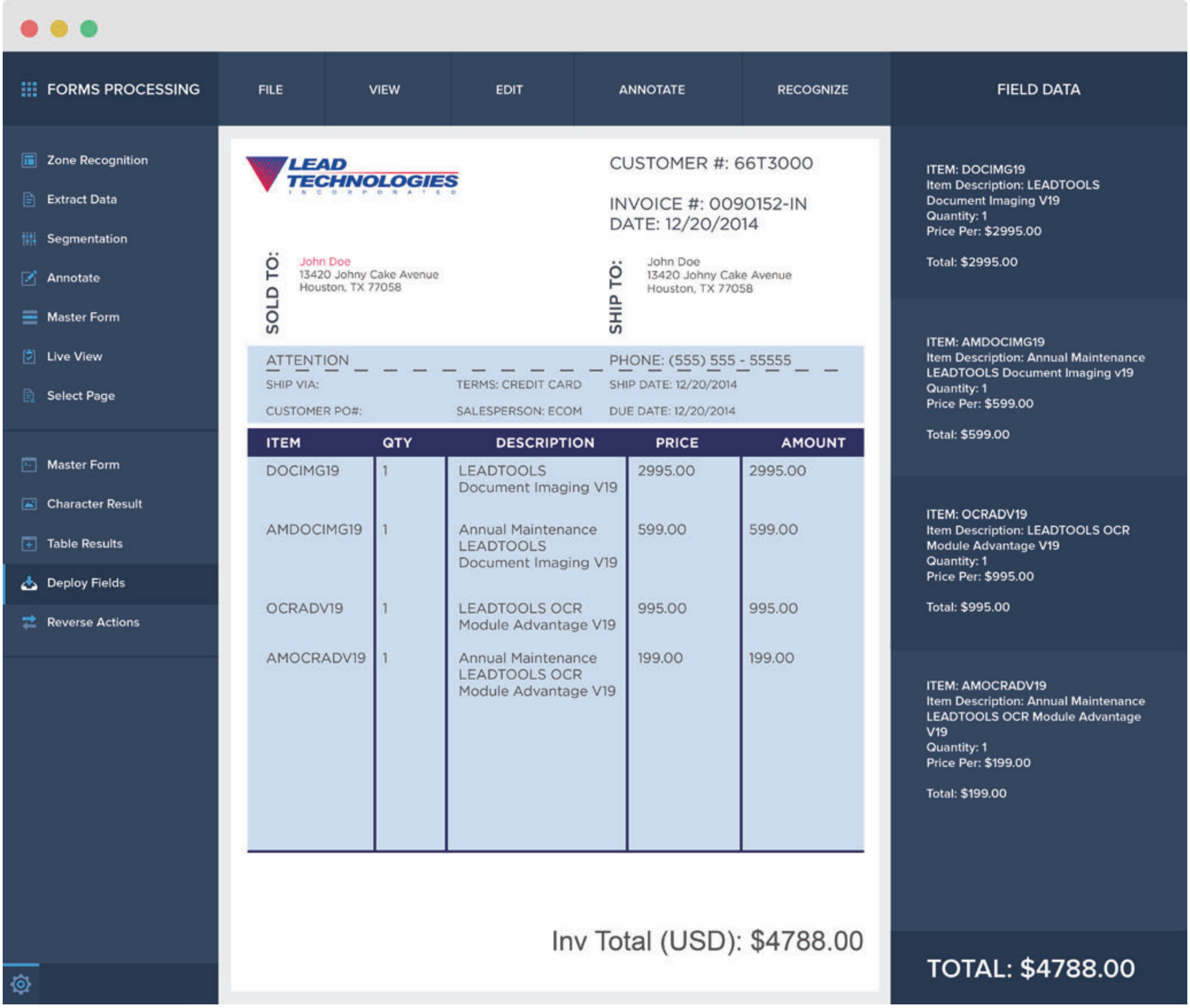
EDITING TOOLS

DOWNLOAD A FREE 60 DAY EVALUATION

LEADTOOLS
Document Imaging

- Document and image viewers for all platforms
- Zero-footprint HTML5/JavaScript UI controls & web services
- Image clean-up and pre-processing
- Virtual Printer and TWAIN/web scanning
- Hundreds of file and image formats for loading, saving and compression
- Thousands of image processing algorithms

Best of all, these amazing features are extremely programmer-friendly and allow the development of enterprise-level document imaging applications with minimal code and time.



CUSTOMER #: 66T3000

INVOICE #: 0090152-IN
DATE: 12/20/2014

SOLD TO:
John Doe
13420 Johny Cake Avenue
Houston, TX 77058

SHIP TO:
John Doe
13420 Johny Cake Avenue
Houston, TX 77058

ATTENTION _ _ _ _ _ _ _ _ _ _ PHONE: (555) 555 - 55555 _ _

SHIP VIA: TERMS: CREDIT CARD SHIP DATE: 12/20/2014

CUSTOMER PO#: SALESPERSON: ECOM DUE DATE: 12/20/2014

ITEM	QTY	DESCRIPTION	PRICE	AMOUNT
DOCIMG19	1	LEADTOOLS Document Imaging V19	2995.00	2995.00
AMDOCIMG19	1	Annual Maintenance LEADTOOLS Document Imaging V19	599.00	599.00
OCRADV19	1	LEADTOOLS OCR Module Advantage V19	995.00	995.00
AMOCRADV19	1	Annual Maintenance LEADTOOLS OCR Module Advantage V19	199.00	199.00

Inv Total (USD): $4788.00

ITEM: DOCIMG19
Item Description: LEADTOOLS Document Imaging V19
Quantity: 1
Price Per: $2995.00

Total: $2995.00

ITEM: AMDOCIMG19
Item Description: Annual Maintenance LEADTOOLS Document Imaging v19
Quantity: 1
Price Per: $599.00

Total: $599.00

ITEM: OCRADV19
Item Description: LEADTOOLS OCR Module Advantage V19
Quantity: 1
Price Per: $995.00

Total: $995.00

ITEM: AMOCRADV19
Item Description: Annual Maintenance LEADTOOLS OCR Module Advantage V19
Quantity: 1
Price Per: $199.00

Total: $199.00

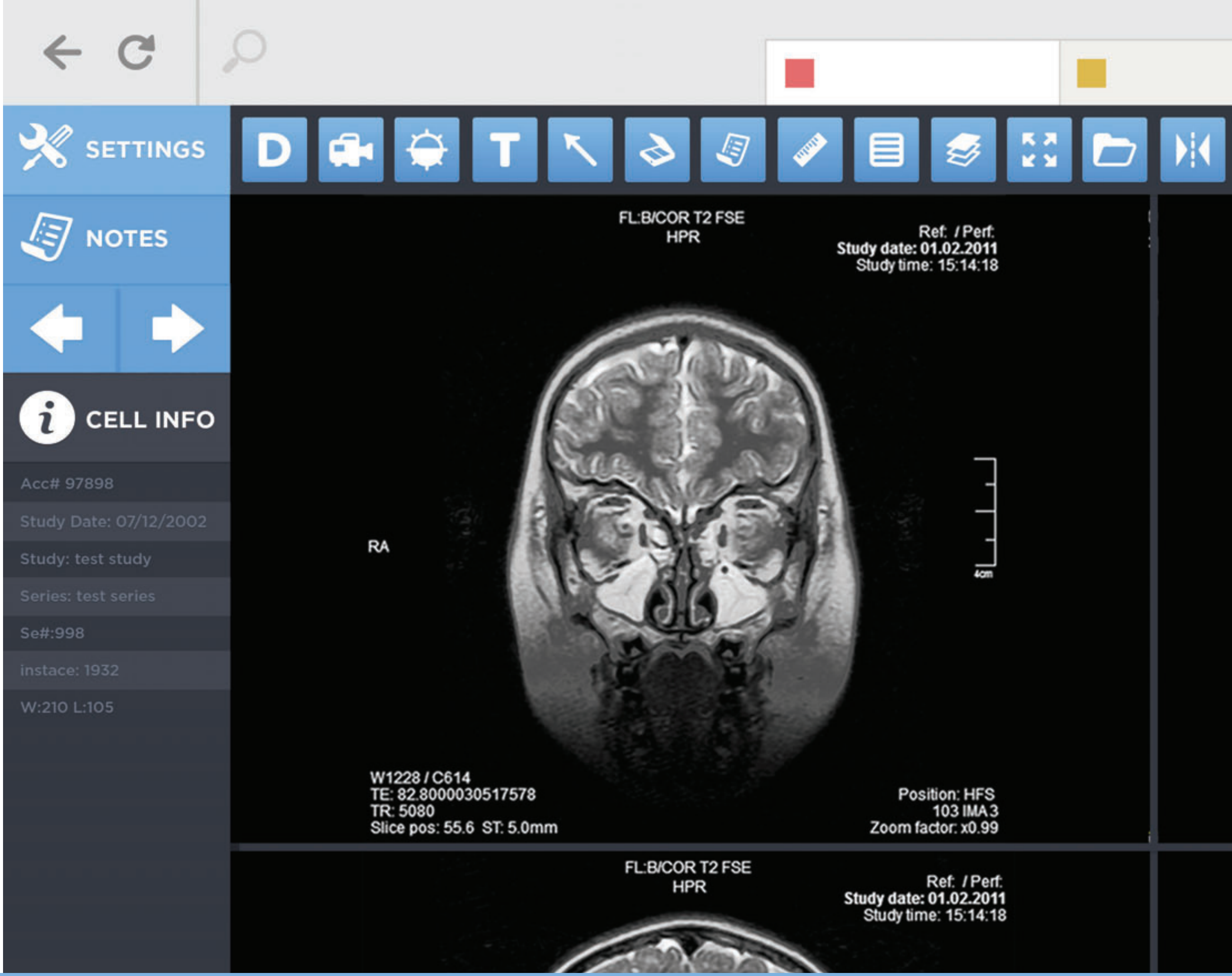
TOTAL: $4788.00

LEADTOOLS
Medical Imaging

LEAD has served developers in the health and life sciences industries with a specific medical-focused version of LEADTOOLS for over 20 years. Our SDKs offer developers all the code necessary to create medical applications that meet the demands of today's rapidly evolving, global healthcare system.

Features include:

- Comprehensive DICOM SDK supporting the latest specifications
- PACS client and server components
- OEM-ready applications with source code
- HTML5 Zero-footprint Medical Viewer
- Rich Client Medical Web Viewer



Acc# 97898
Study Date: 07/12/2002
Study: test study
Series: test series
Se#:998
instace: 1932
W:210 L:105

FL:B/COR T2 FSE
HPR

Ref: / Perf:
Study date: 01.02.2011
Study time: 15:14:18

W1228 / C614
TE: 82.8000030517578
TR: 5080
Slice pos: 55.6 ST: 5.0mm

Position: HFS
103 IMA 3
Zoom factor: x0.99

FL:B/COR T2 FSE
HPR

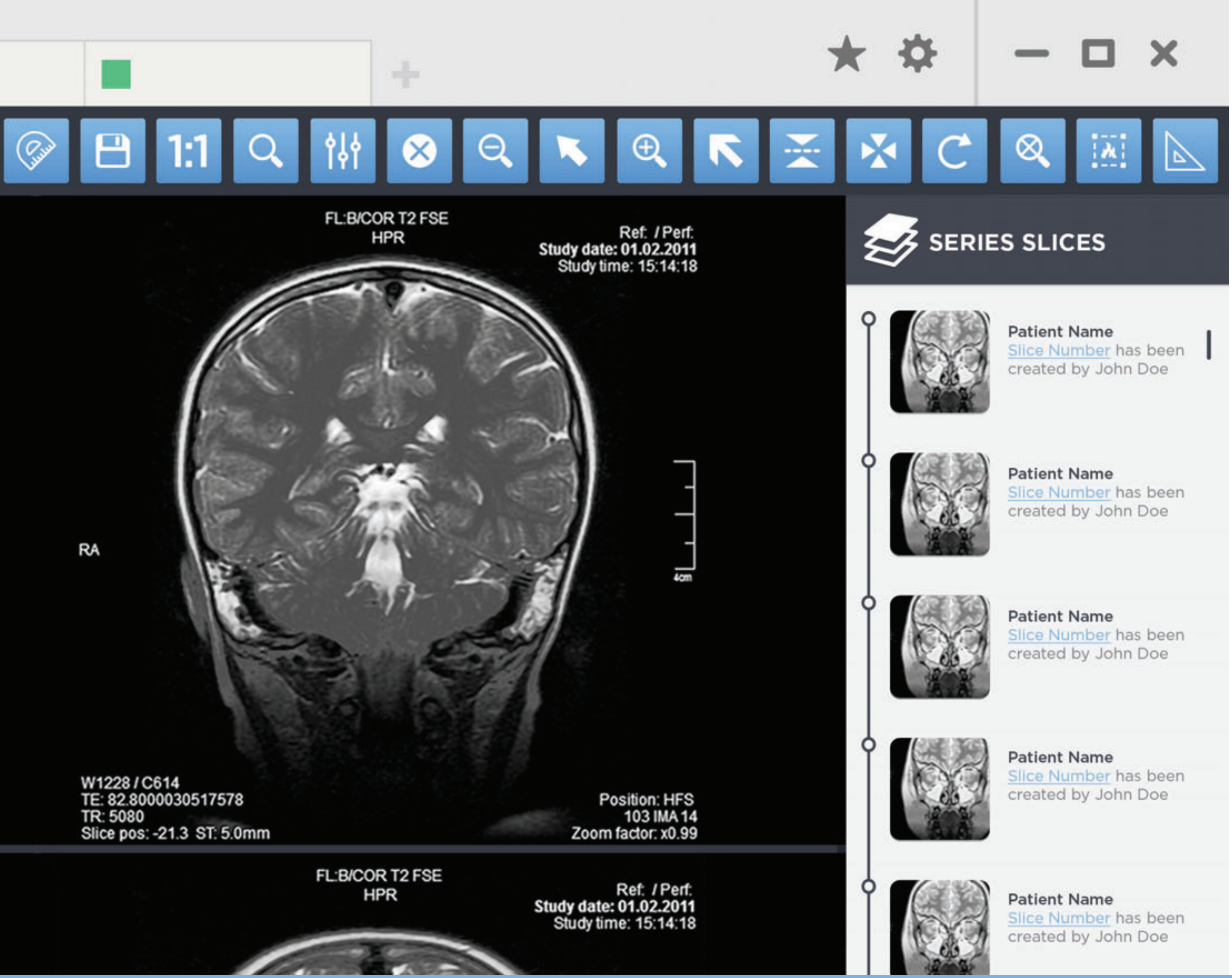
Ref: / Perf:
Study date: 01.02.2011
Study time: 15:14:18

DOWNLOAD A FREE 60 DAY EVALUATION

LEADTOOLS

Medical Imaging

- Specialized 8-16 bit grayscale image display
- Medical image processing
- Medical annotations and collaboration tools
- 3D volume construction and advanced visualization
- Optional third-party cloud storage integration
- Print to PACS
- HL7 messaging and communication
- Clinical Context Object Workgroup (CCOW)



Patient Name
Slice Number has been created by John Doe

Patient Name
Slice Number has been created by John Doe

Patient Name
Slice Number has been created by John Doe

Patient Name
Slice Number has been created by John Doe

Patient Name
Slice Number has been created by John Doe

LEADTOOLS Multimedia

LEADTOOLS Multimedia SDK products are specifically designed for the development of audio and video applications across a wide variety of industries including defense, broadcast and security. The SDK offers a full range of technologies to developers and turns normally complex DirectShow and Media Foundation projects into simple tasks.

Features include:

- Capture, convert, stream and play audio and video
- Hundreds of formats and codecs for DirectShow and Media Foundation
- MPEG-2 Transport Stream, H.265, H.264
- KLV metadata
- UAV and UAS
- Audio and video processing filters and transforms including overlays and motion detection
- DICOM reading and writing
- DVR
- RTSP
- Video conferencing

DOWNLOAD A FREE 60 DAY EVALUATION

LEADTOOLS

Anywhere

The world's most advanced and popular imaging SDK, LEADTOOLS, includes native libraries for every major development platform on the market. Leverage LEADTOOLS' state-of-the-art imaging features to create powerful applications that run natively to get the most out of each device's hardware.

- Native libraries for .NET, Win API, WinRT, Linux, iOS, OS X and Android
- HTML5 viewers, JavaScript libraries and web services for creating zero-footprint applications
- Document viewers and converters
- Native annotation and markup
- OCR and barcode
- DICOM Data Set and PACS components
- Hundreds of formats and image processing algorithms

```
private void recognize() {
    // Create an instance of the engine
    OcrEngine ocrEngine = OcrEngineManag
    // Start the engine using default pa
    ocrEngine.startup(null, null, null,
    // Input TIF file name
    string tifFileName = "/image.tif";
}
```

```
-(void)recognize {
    // Create an insta
    NSString *bundleResource
    LTOcrEngine* ocrEngine
    // Start the engin
    [_ocrEngine startu
    // Input TIF file n
    NSString* tifFileName
    // Output PDF file name
    NSString* pdfFileName =
```

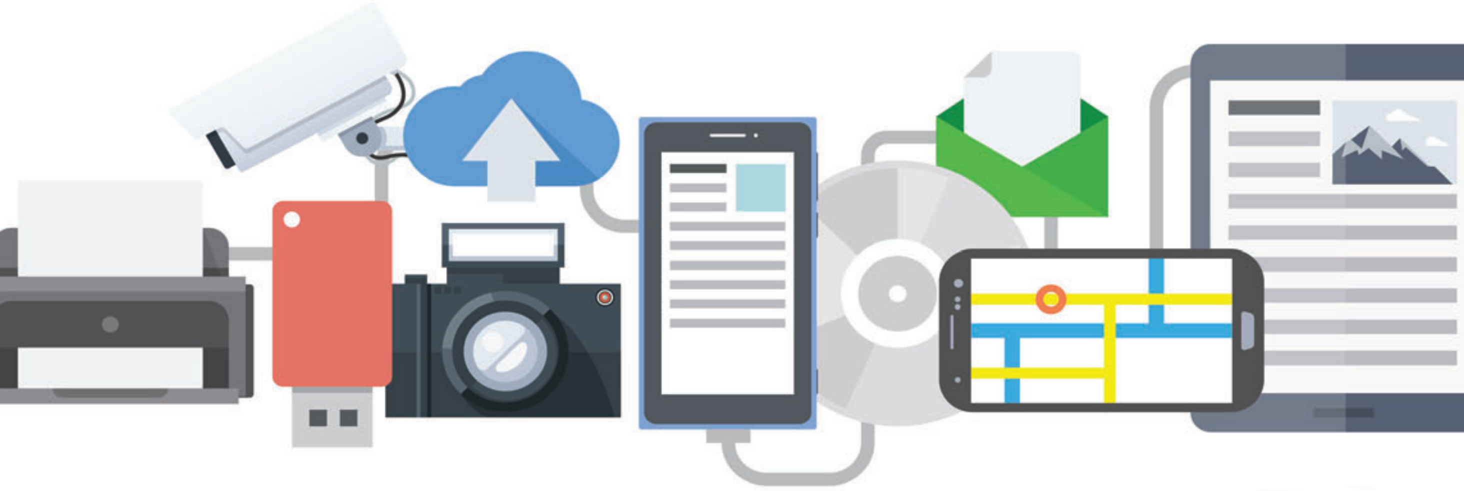
```
public void Recognize()
{
    // Create an instance of the en
    IOcrEngine ocrEngine = OcrEngin
    // Start the engine using defau
    ocrEngine.Startup(null, null, n
    // Input TIF file name
```

Your Application
File Edit View Build Debug Tools Test Window Help
YOUR APP
B I U S
LEADTOOLS medical imaging SDKs offe
DICOM and image annotation options
grayscale, bitonal and color images. LE
used to [redacted] char
iew of LEADTOOLS
look at this asap
YOUR APP
YOUR APP

What's New in V19

LEADTOOLS Version 19 is packed with new features for Document, Medical and Multimedia Imaging. Developers can do more than ever before with V19:

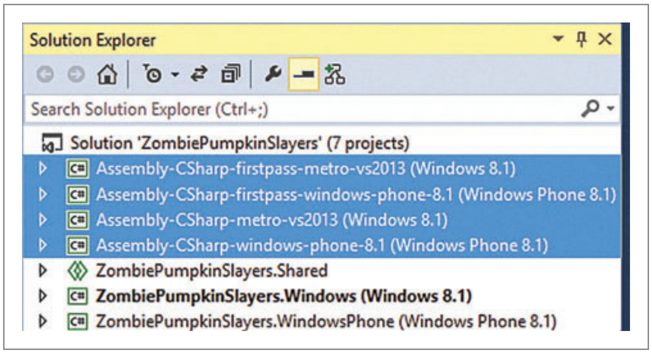
- Improved OCR accuracy across all languages including East Asian
- New SVG engine for smooth scrolling and zooming of documents
- Document viewer supporting any format with text search, annotations, lazy loading, and SVG rendering
- Convert PDF & office formats at 100% accuracy without the need for OCR
- Forms processing with advanced editor to handle invoices, checks, driver licenses, passports, etc.
- Fastest forms recognition SDK available
- Scan with TWAIN from the web
- Redesigned, highly customizable, cloud-enabled HTML5 / JavaScript medical viewer
- HL7 SDK with decoders, listener service and MWL integration
- H.265 with 4K and 8K ultra-high-definition



DOWNLOAD 60 DAY EVALUATION
WWW.LEADTOOLS.COM

LEAD TECHNOLOGIES INCORPORATED

SALES@LEADTOOLS.COM
800.637.1840



Solution Explorer

Search Solution Explorer (Ctrl+;)

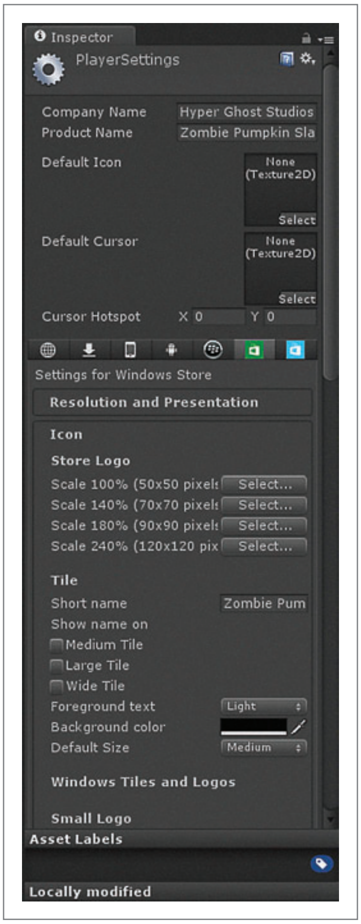
- Solution 'ZombiePumpkinSlayers' (7 projects)
  - Assembly-CSharp-firstpass-metro-vs2013 (Windows 8.1)
  - Assembly-CSharp-firstpass-windows-phone-8.1 (Windows Phone 8.1)
  - Assembly-CSharp-metro-vs2013 (Windows 8.1)
  - Assembly-CSharp-windows-phone-8.1 (Windows Phone 8.1)
  - ZombiePumpkinSlayers.Shared
  - **ZombiePumpkinSlayers.Windows (Windows 8.1)**
  - ZombiePumpkinSlayers.WindowsPhone (Windows Phone 8.1)

Figure 7 **Additional Projects**

I'll show you an example using some Windows platform-specific code. If I want to access GeoLocation in the Windows Runtime, I can include this code directly in my MonoBehavior-derived class (as shown in **Figure 4**), which can be assigned to any GameObject. Because this code is compiled in the Editor with Mono, it will fail without wrapping it in a preprocessor directive. We need to tell Unity this code is only compiled when outside of the Editor. You might wonder what the difference is between NETFX_CORE and UNITY_METRO. The former is a compilation setting and is used only when your game assemblies are compiled for the Windows Runtime. The latter is used when you've selected Windows Store as the platform in the Unity build settings. This can be active in your C#, UnityScript or Boo code. I personally prefer the NETFX_CORE directive to wrap my WinRT code rather than UNITY_METRO, as NETFX_CORE is only available on export/build and UNITY_METRO is active in the Editor as soon as you switch platforms in the Build Settings. This means code can run while in the Editor, which can be a problem if it contains a platform-specific API. You can add other directives on the same line such as && !UNITY_EDITOR, but I prefer just NETFX_CORE.

Builds

Unity will compile your Windows Store and Windows Phone 8.1 code using the .NET Framework, specifically the Windows Runtime. It doesn't use Silverlight for Windows Phone 8.1. However, the Windows Phone 8 code is compiled as HH, a Silverlight package for Windows Phone, as expected.



Inspector

PlayerSettings

Company Name: Hyper Ghost Studios
Product Name: Zombie Pumpkin Sla
Default Icon: None (Texture2D) Select
Default Cursor: None (Texture2D) Select
Cursor Hotspot: X 0 Y 0

Settings for Windows Store

Resolution and Presentation

Icon

Store Logo
Scale 100% (50x50 pixel: Select...
Scale 140% (70x70 pixel: Select...
Scale 180% (90x90 pixel: Select...
Scale 240% (120x120 pix Select...

Tile
Short name: Zombie Pum
Show name on
□ Medium Tile
□ Large Tile
□ Wide Tile
Foreground text: Light
Background color
Default Size: Medium

Windows Tiles and Logos

Small Logo

Asset Labels

Locally modified

Figure 8 **Player Settings for Windows Store**

Unity will compile your game assemblies and create a Visual Studio solution that you in turn will use to create your final build. In other words, apart from when doing a desktop build, you'll need to do your final compilation in Visual Studio before sending your game to the Windows Store or Windows Phone Store. The generated Visual Studio solution contains the Unity binaries packaged with your game assemblies and a basic XAML/C# or C++ host app. Every time you build from Unity, the only thing that's overwritten by Unity is your Data folder. Any custom code or other changes you make to your Visual Studio solution aren't overwritten during future builds from Unity.

Windows Store and Universal App Builds

I'll take a look at the build options for Windows Store, which are shown in **Figure 5**. The Windows Store build types in Unity include Windows 8, Windows 8.1, Windows Phone 8.1, and Universal solutions, which contain both Windows Phone 8.1 and Windows 8.1 projects and a shared project, as shown in **Figure 6**. Also note the Unity C# Projects checkbox in **Figure 5**. This is an incredibly useful feature that I strongly recommend. It will create the Visual Studio solution with additional projects containing your Unity code, as shown in **Figure 7**. You can actually change code in these projects to a certain extent and upon compile Visual Studio will ask Unity to reprocess your game assemblies without having to do a full build again from Unity. This means you can edit your Unity code in your final Visual Studio solution, which is amazing for platform debugging—it means you don't have to rebuild your solution from Unity every time you make a code change.

When Unity generates your Visual Studio solution, there are several things to note. First, your Windows Store project will default (as of this writing) to an ARM build platform, not x86. This can be very confusing when you try to build and run because you'll get an error that your system isn't an ARM architecture. While I do recommend you create builds for both ARM and x86, for local testing you'll want to choose x86 to avoid that ARM error. Also, as of this writing, there are no 64-bit players on Unity for Windows Store or Windows Phone. If you have one of the few ARM Windows 8 devices, such as the Surface 1 or Surface 2 (RT not Pro; Pro is an x86-based system), then by all means you can choose ARM and do a remote deployment and debugging to it. However, most developers use an x86-based PC for local testing.

Next, you'll note that the Build Configuration Type includes Debug, Master and Release options. This is in contrast to typical .NET applications, which include only Debug and Release. The Debug build is the slowest. It contains debugging information and also supports the Unity Profiler—a performance tool in the Pro version used for optimizing games. The Release build is fully optimized but also supports the Unity Profiler. The Master build is the one you submit to each store. This is the optimized, production-ready version and it does not support the Unity Profiler. When I first test a game, I choose a Master build so I can get a true idea of the performance on my device.

For Windows Phone, the two main choices are whether to build for Windows Phone 8 or

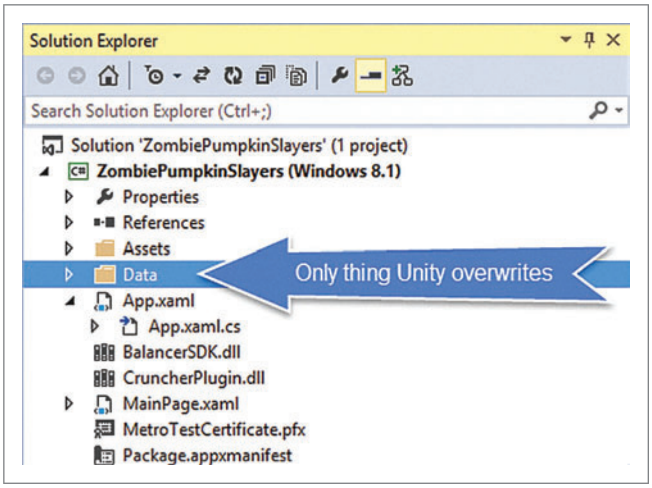


Figure 9 **The Data Folder in a Visual Studio Project**

for Windows Phone 8.1. Windows Phone 8 apps will run just fine on a Windows Phone 8.1 device, although the code-sharing strategy for platform-specific code is much better on Windows Phone 8.1 because you can target universal apps, which share around 90 percent of the API with Windows Store apps. For a typical game, however, this code sharing may not have much meaning unless you're writing platform-specific code.

Once you have your solution in Visual Studio, packaging it up and sending it to the store is the same as for any Windows Phone or Windows Store application. A Windows Store developer account is a requirement; you pay once and have it for life. That submission process has been covered many times before, but for a quick overview see "Publish to the Windows Store" (bit.ly/Za0Jlx), "Publish to the Windows Phone Store" (bit.ly/1rdDCwR) and "Why and How to Port Your Unity Game to Windows and Windows Phone" (bit.ly/1ElzrcL).

Player Settings I discussed the basic process of building from Unity to Windows, but didn't look at how you can set icons, splash screens, the app name and other required items. These can all be found in the Player Settings, which is loaded from the Build Settings screen and allows you to configure all the major settings for Windows Store and Windows Phone applications, as shown in **Figure 8**. You can specify the splash screen, app name, publisher, compilation settings, icons, app capabilities like location services, and a few other settings here. These settings get pushed down to your Visual Studio solution the first time you build your solution from Unity.

App Icons Unity provides some default icons for your application so it will build in Visual Studio. You *must* change these default icons or risk failing certification. Unity makes it pretty easy to see what images you can set on your project, but refer to the certification documents for each platform to determine which images are required. It's imperative you check your Visual Assets in the Package.appxmanifest file in Visual Studio (for Windows Store and universal apps) as Unity may have set some default images that you aren't going to replace with your own and will need to be removed.

Additional Learning

Adam's Channel 9 blog: bit.ly/AdamChannel9

Microsoft Virtual Academy: Developing 2D & 3D Games with Unity for Windows bit.ly/FreeUnityTraining

Unity Resources: unity3d.com/learn

At a minimum, Windows Store apps require a 50x50 Store logo, a 150x150 square logo, a 30x30 square logo, and a 620x300 splash screen, as described on the Windows Dev Center (bit.ly/1vIJsue). Windows Phone 8 apps require the app list image and an image for the small and medium default tiles for the Start screen as described on the Windows Dev Center (bit.ly/1oKo8Ro). Windows Phone 8.1 apps require the app list image and default tile images, as well. The stores (which are said to be unifying with Windows 10) may ask for some other images during submission, such as screenshots.

If you're going to provide just a single image for Start screen tiles, you should provide the image scaled to 240 percent for Windows Phone Store apps and 180 percent for Windows Store apps as per the Windows Dev Center (bit.ly/1nzWber).

By the way, a while back I wrote a no-frills utility to help generate images for Windows Store apps. Check it out at bit.ly/ImageProcessor.

Rebuilds and Overwriting Recall that Unity overwrites just your Data folder in your Visual Studio project (see **Figure 9**). This is a double-edged sword. You can set the splash screen, icons, and so on at any time in Unity (and I recommend doing so), but if you've already generated your Visual Studio solution, the Package.appxmanifest file that contains these settings will not be overwritten the next time you build from Unity. You have to manually set the visual assets, capabilities and so forth in your Visual Studio solution in the Package.appxmanifest file; or build to a new folder and diff the folder structure using a diff tool like BeyondCompare; or delete your Visual Studio solution if you have no custom code in it and let Unity regenerate it. I typically choose the third option because I keep my icon and splash screen images configured inside Unity. I just have to remember to bring in my custom live tile images into my Visual Studio solution. If you're building for Windows Phone 8 (but not 8.1), the icon must be set in Visual Studio anyway; that's one of the few things that isn't pushed from Unity.

Wrapping Up

The build process is extremely simple. Bring up build settings, click build, open solution, deploy. I showed options for customizing your builds via plug-ins and preprocessor directives. Because you can call platform-specific code outside of Mono, you have some power at your fingertips. Don't forget to check out prime31.com for Unity plug-ins that are currently free for the Windows platform. They allow you to easily integrate platform features with just a couple lines of code. Keep an eye on Channel 9 for some new bite-sized Unity training content (think 10-minute videos) and on my Channel 9 blog for many mini-tips for building games for the platform. Until next time! ■

Adam Tuliper *is a senior technical evangelist with Microsoft living in sunny SoCal. He is an indie game dev, co-admin of the Orange County Unity Meetup, and a Pluralsight author. He and his wife are about to welcome their third child, so reach out to him while he still has a spare moment at adamt@microsoft.com or on Twitter at twitter.com/AdamTuliper.*

Thanks to the following technical experts from Unity for reviewing this article: *Tomas Dirvanauskas, Ignas Ziberkas and Tautvydas Žilys*

ALTOVA® MissionKit®

Bring your XML development projects to light with the complete set of tools from Altova®

Experience how Altova MissionKit®, a software development suite of industrial-strength XML, SQL, and data integration tools, can simplify even the most advanced XML development projects.

POWERED BY RAPTORXML®

NEW in Version 2015:

- Sophisticated XML file modifications via XQuery Update Facility in XMLSpy
- New tools for XBRL Table and XBRL Formula editing in XMLSpy
- JSON data mapping in MapForce
- Option to generate HTML fragments in StyleVision
- And much more

Altova MissionKit includes multiple, tightly-integrated XML tools:

XMLSpy® – industry-leading XML editor

- Support for XML Schema 1.1 and XPath/XSLT/XQuery 3.0
- Industry's strongest validation engine with Smart Fix
- Powered by RaptorXML® for lightning-fast validation & processing
- Graphical editing views, powerful debuggers, code generation, & more

MapForce® – any-to-any data mapping & integration tool

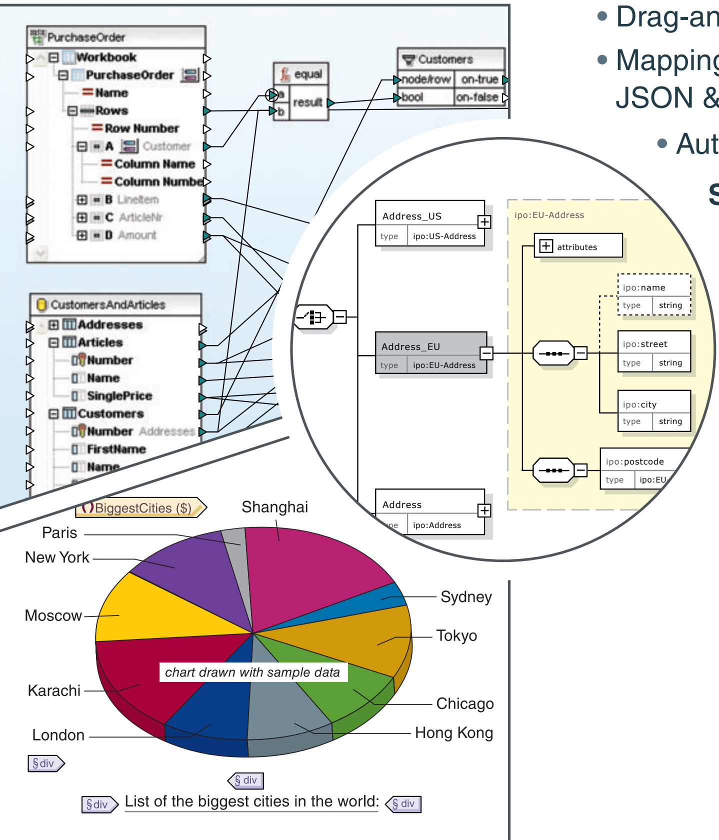
- Drag-and-drop data conversion
- Mapping of XML, DBs, EDI, XBRL, flat files, Excel®, JSON & Web services
- Automation via MapForce Server

StyleVision® – visual XSLT stylesheet & report designer

- Graphical XSLT stylesheet & report design for XML, XBRL, & SQL databases
- Output to HTML, PDF, Word & more
- Automation via StyleVision Server

Download a 30 day free trial!

Try before you buy with a free, fully functional trial from **www.altova.com**



Equip Your Apps with OBEX

Uday Gupta

During the last decade, Bluetooth has become a widely used technology for short-range wireless communication between devices such as mobile phones, personal computers and headsets. The Bluetooth Special Interest Group (BTSIG) is the industry body that defines the standards for wireless services in Bluetooth Profile specifications.

One such profile is the Object Push Profile (OPP), which is used to send files from one device to another. The Object Exchange Protocol (OBEX) is part of the foundation of the OPP. OBEX is also used in profiles other than OPP, as it's a generic protocol for transferring objects between devices.

This article discusses:

- Enabling the Object Exchange Protocol in mobile apps
- Developing universal apps for Windows 8.1 and Windows Phone 8.1
- File sharing across multiple mobile devices

Technologies discussed:

Windows 8.1, Windows Phone 8.1, Windows Phone Runtime API, Visual Studio 2013

Code download available at:

msdn.microsoft.com/magazine/msdnmag1214

API library code available at:

1drv.ms/1yxmc51

For developers looking to use OBEX within their apps, I've developed a set of APIs over the Windows Runtime (WinRT) Bluetooth APIs that provide OBEX from within the app. These APIs come as a library package for universal apps, which means Windows Store apps and Windows Phone Silverlight apps can leverage the same set of APIs.

OBEX and OPP

First, it's important to understand what OBEX and OPP are and how they work. OPP lets a Bluetooth device send a file or object to another OPP-capable Bluetooth device. The intended use for OBEX was file sharing via infrared channels. The BTSIG chose to reuse this protocol for file sharing over Bluetooth. Aside from the underlying transport medium, OBEX-over-infrared and OBEX-over-Bluetooth are similar.

OBEX is based on a client-server model in which the recipient Bluetooth device is running an OBEX server that listens for and accepts client connections. The client Bluetooth device connects to the server Bluetooth device as a stream channel over Bluetooth. The authentication requirements to allow the connection and

Figure 1 **Methods and Associated Events from BluetoothService**

Method (with Parameters)	Associated Events
[static] GetDefault	No associated event
SearchForPairedDevicesAsync	Success - SearchForDevicesSucceeded Failure - SearchForPairedDevicesFailed

Figure 2 **BluetoothService Counting Paired Devices**

```
BluetoothService btService = BluetoothService.GetDefault();
btService.SearchForPairedDevicesFailed += btService_
SearchForPairedDevicesFailed;
btService.SearchForPairedDevicesSucceeded += btService_
SearchForPairedDevicesSucceeded;
await btService.SearchForPairedDevicesAsync();

void btService_SearchForPairedDevicesSucceeded(object sender,
  SearchForPairedDevicesSucceededEventArgs e)
{
  // Get list of paired devices from e.PairedDevices collection
}

void btService_SearchForPairedDevicesFailed(object sender,
  SearchForPairedDevicesFailedEventArgs e)
{
  // Get the failure reason from e.FailureReason enumeration
}
```

object transfer depend on the service or application using OBEX. For example, OPP can allow an unauthenticated connection in order to streamline the process of quickly exchanging business cards. Other services using OBEX may only allow authenticated connections.

Files are shared using three types of packets. These packets are known as first PUT, intermediate PUT and last PUT. The first PUT packet marks the file transfer initialization and the last PUT packet marks its completion. The multiple intermediate PUT packets contain the bulk of the data. After the server receives each PUT packet, it returns an acknowledgement packet to the client.

In a typical scenario, OBEX packets are sent as follows:

1. The OBEX client connects to the recipient device by sending a connect packet. This packet specifies the maximum packet size the client can receive.
2. After receiving a response from the server indicating the connection has been accepted, the OBEX client sends the first PUT packet. This contains the metadata describing the object, including the file name and size. (While the OBEX protocol allows for this first PUT packet to also include object data, the implementation of OBEX in the library I've developed doesn't send any of that data in the first PUT packet.)
3. After receiving acknowledgement the server has the first PUT packet, the OBEX client sends the multiple PUT packets that contain object data. The length of these packets is limited by the maximum packet size the server can receive, set by the server's response to the connect packet sent in step one.
4. The last PUT packet contains the last PUT constant and the final chunk of object data.
5. Once the file sharing is complete, the OBEX client sends a disconnect packet and closes the Bluetooth connection. The OBEX protocol allows repetition of steps two through three to send multiple objects on the same connection.

At any point, the OBEX client can abort the sharing process by sending an ABORT packet. The sharing is immediately canceled. In the library I've written, the OBEX protocol implementation details are hidden and you'll only see high-level APIs.

Figure 3 **Methods and Associated Events from BluetoothSockets**

Method (with Parameters)	Associated Events

Constructor(Bluetooth.Core.Services.BluetoothDevice)
Constructor(Bluetooth.Core.Services.BluetoothDevice, System.UInt32)
Constructor(Bluetooth.Core.Services.BluetoothDevice, System.String)
Constructor(Bluetooth.Core.Services.BluetoothDevice, System.UInt32, System.String)
	No associated event

PrepareSocketAsync	Success – SocketConnected Failure – ErrorOccured

SendDataAsync(System.Byte[])
SendDataAsync(System.String)
	No associated Event

CloseSocket	SocketClosed
No associated method	DataReceived

Figure 4 **Methods and Associated Events from ObexService**

Method (with Parameters)	Associated Events
[static] GetDefaultForBluetoothDevice(Bluetooth.Core.Services.BluetoothDevice)	No associated event
ConnectAsync	Success – DeviceConnected Failure – ConnectionFailed
SendFileAsync(Windows.Storage.IStorageFile)	Success: ServiceConnected DataTransferProgressed DataTransferSucceeded Disconnecting Disconnected Failure: ConnectionFailed DataTransferFailed
AbortAsync	Aborted

The OBEX Library

The Bluetooth OBEX client library for Windows Store apps is designed as a portable library targeting: Windows Store apps and Windows Phone Silverlight 8.1 apps. It contains three DLLs that make the library a runtime for the OBEX client. Each DLL is designed to handle a specific task: Bluetooth.Core.Service, Bluetooth.Core.Sockets and Bluetooth.Services.Obex.

Bluetooth Core Service The file Bluetooth.Core.Service.dll contains the Bluetooth.Core.Service namespace. This library is designed to search for and count nearby Bluetooth devices paired with the client device (see **Figure 1**). Currently, it's restricted to a one-time count of paired devices. Future versions will contain a watcher to keep looking for additional Bluetooth devices.

The core Bluetooth service is represented by a static class called

Figure 5 **Obex Service Usage**

```
protected async override void OnNavigatedTo(Windows.UI.Xaml.Navigation.
NavigationEventArgs e)
{
  base.OnNavigatedTo(e);

  ObexService obexService = ObexService.GetDefaultForBluetoothDevice(null);
  obexService.DeviceConnected += obexService_DeviceConnected;
  obexService.ServiceConnected += obexService_ServiceConnected;
  obexService.ConnectionFailed += obexService_ConnectionFailed;
  obexService.DataTransferProgressed += obexService_DataTransferProgressed;
  obexService.DataTransferSucceeded += obexService_DataTransferSucceeded;
  obexService.DataTransferFailed += obexService_DataTransferFailed;
  obexService.Disconnecting += obexService_Disconnecting;
  obexService.Disconnected += obexService_Disconnected;
  obexService.Aborted += obexService_Aborted;
  await obexService.ConnectAsync();
}

async void obexService_DeviceConnected(object sender, EventArgs e)
{
  // Device is connected, now send file
  await (sender as ObexService).SendFileAsync(fileToSend);
}

void obexService_ServiceConnected(object sender, EventArgs e)
{
  // Device connected to Obex Service on target device
}

void obexService_ConnectionFailed(object sender, ConnectionFailedEventArgs e)
{
  // Connection to service failed
}
```

```
void obexService_DataTransferProgressed(object sender,
DataTransferProgressedEventArgs e)
{
  // Get data transfer progress from DataTransferProgressedEventArgs
}

void obexService_DataTransferSucceeded(object sender, EventArgs e)
{
  // Data transfer succeeded
}

void obexService_DataTransferFailed(object sender,
DataTransferFailedEventArgs e)
{
  // Data transfer failed, get the reason from DataTransferFailedEventArgs
}

void obexService_Disconnecting(object sender, EventArgs e)
{
  // Device is disconnecting from service
  }

void obexService_Disconnected(object sender, EventArgs e)
{
  // Device is now disconnected from targeted device and service
}

void obexService_Aborted(object sender, EventArgs e)
{
  // Data transfer operation is aborted
}
```

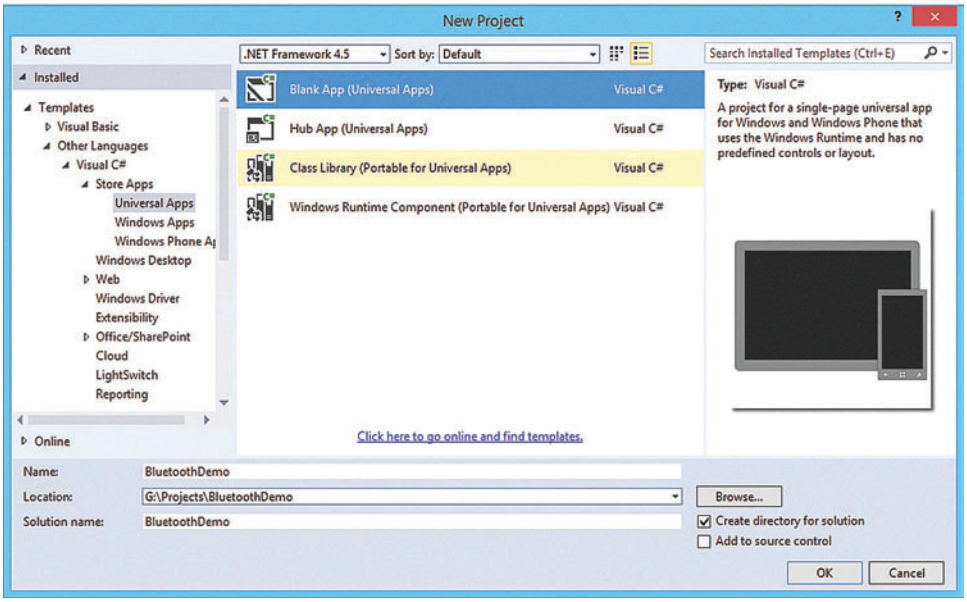
BluetoothService, shown in **Figure 2**. This class has an API to asynchronously count devices.

Bluetooth Core Sockets The file Bluetooth.Core.Sockets.dll contains the Bluetooth.Core.Sockets namespace and is designed to support stream-based socket operations over a Bluetooth connection. The socket functionality is exposed via the BluetoothSockets class (see **Figure 3**). This is neither a TCP nor UDP socket. All communication with the recipient device takes place through BluetoothSockets.

Bluetooth Services Obex The Bluetooth.Services.Obex.dll file contains the Bluetooth.Services.Obex namespace. This is the core implementation of OBEX, exposed through a class named ObexService. This class provides the abstracted view of the Bluetooth OPP specification. It exposes the method that helps connect, send and disconnect from the recipient Bluetooth device. **Figure 4** lists the APIs and associated events this class exposes, and **Figure 5** demonstrates the usage.

A typical usage scenario for using these libraries is something like this:

- Using APIs in Bluetooth.Core.Service, count all the paired OPP-capable Bluetooth devices. OPP-capability check is already implemented.
- Get the instance of BluetoothDevice with which you want to share files.
- Get an instance of ObexService for the recipient Bluetooth device by passing the instance of BluetoothDevice to factory method. The ObexService class will internally create an instance of BluetoothSocket, over which the ObexService will share the file.
- Once file sharing is complete, the ObexService is disconnected automatically.



Type: Visual C#

A project for a single-page universal app for Windows and Windows Phone that uses the Windows Runtime and has no predefined controls or layout.

Name: BluetoothDemo

Location: G:\Projects\BluetoothDemo

Solution name: BluetoothDemo

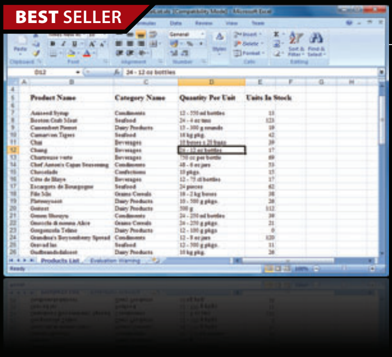
☑ Create directory for solution

□ Add to source control

Figure 6 **Blank Universal App to Create a New Project**

/update/2014/12

ComponentSource®
The Definitive Source of Software Components
www.componentsource.com

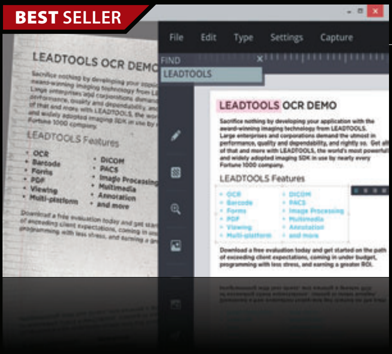


Aspose.Total for .NET | from $2,449.02

ASPOSE
Your File Format APIs

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



LEADTOOLS Document Imaging SDKs V19 | from $2,995.00 SRP

LEAD TECHNOLOGIES

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

ComponentOne Studio Enterprise 2014 v3 | from $1,315.60

ComponentOne
a division of GrapeCity

.NET Tools for the Professional Developer: Windows, HTML5/Web, and XAML.

- Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers
- A line of HTML5 and JavaScript products for enterprise application development
- Built in themes and an array of designers for creating custom themes and styling
- 40+ Windows 8.1 & Windows Phone 8.1 controls and Universal Windows app support
- All Microsoft platforms supported, Visual Studio 2013, ASP.NET, WinForms, WPF & more

Help & Manual Professional | from $583.10

ecsoftware

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

© 1996-2014 ComponentSource. All Rights Reserved. All prices correct at the time of press. Online prices may vary from those shown due to daily fluctuations & online discounts.

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

US Headquarters
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
USA

European Headquarters
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
United Kingdom

Asia / Pacific Headquarters
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japan
102-0083

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

MasterCard VISA DISCOVER

GSA Schedule
Contract GS-35F-0188R

Get Started

Because my library is targeted toward Windows Store apps and Windows Phone 8.1 apps, I'll start with a universal app. These are a great way of developing apps for all Windows devices. To learn more about universal apps, visit bit.ly/1h3AQeu. To start with universal apps, I'll use Visual Studio 2013 and create a new universal apps project under the Store Apps node (see **Figure 6**). I used Visual C#, but you can also use Visual Basic and Visual C++.

Before I start programming for a Bluetooth OBEX client app, I'll update the package.appxmanifest file for both projects (apps for Windows 8.1 and Windows Phone 8.1):

```
<Capabilities>
  <m2:DeviceCapability Name="bluetooth.rfcomm">
    <m2:Device Id="any">
      <m2:Function Type="name:obexObjectPush"/>
    </m2:Device>
  </m2:DeviceCapability>
</Capabilities>
```

To apply this update, I open package.appxmanifest as a code file by choosing View Code in the context menu in Solution Explorer. I'll place this snippet where <Application> tag ends. That code snippet is required to provide device-level capability to use the Bluetooth radio frequency communication (RFCOMM) service with this app. I also specified all services and device types compatible with this device.

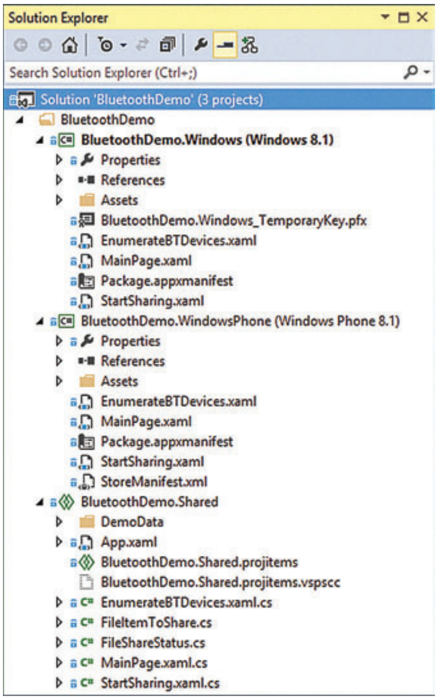
For my scenario, I need obexObjectPush support from the device that's compatible with any OPP-capable device. For more information on Supported profiles on Windows 8.1 and Windows Phone 8.1, visit bit.ly/1pG6rYO. If that capability isn't mentioned, then device enumeration will fail with CapabilityNotDefined enum constant.

Before I start coding, I'll add the reference to the three library files mentioned earlier so I can use the OBEX implementation within those libraries. I need to a add reference to these libraries for both projects separately. If the reference isn't added, the project won't be able to use any features.

I'll follow these coding patterns and practices:

- Implement UX design for Windows Store 8.1 app in Windows project.
- Implement UX design for Windows Phone 8.1 apps in Windows Phone project.
- Implement common feature in Shared project.
- Perform platform-specific implementation in Shared project using platform-specific compiler constants. For Windows 8.1, I'll use WINDOWS_APP. For Windows Phone 8.1, I'll use WINDOWS_PHONE_APP. These compiler constants are already defined as part of the project.

Download the sample code to get hands-on experience with these libraries, along with the coding practices you should follow for developing universal apps. **Figure 7** shows the sample project's Solution Explorer window with the file structure and patterns.



```
Solution 'BluetoothDemo' (3 projects)
  BluetoothDemo
    BluetoothDemo.Windows (Windows 8.1)
      Properties
      References
      Assets
      BluetoothDemo.Windows_TemporaryKey.pfx
      EnumerateBTDevices.xaml
      MainPage.xaml
      Package.appxmanifest
      StartSharing.xaml
    BluetoothDemo.WindowsPhone (Windows Phone 8.1)
      Properties
      References
      Assets
      EnumerateBTDevices.xaml
      MainPage.xaml
      Package.appxmanifest
      StartSharing.xaml
      StoreManifest.xml
    BluetoothDemo.Shared
      DemoData
      App.xaml
      BluetoothDemo.Shared.projitems
      BluetoothDemo.Shared.projitems.vspscc
      EnumerateBTDevices.xaml.cs
      FileItemToShare.cs
      FileShareStatus.cs
      MainPage.xaml.cs
      StartSharing.xaml.cs
```

Figure 7 **Solution Explorer of BluetoothDemo App**

Enumerating Paired Devices

Before I can share files with a paired device, I need to see the list of paired devices and select a target. I'll get the handle to the instance of Bluetooth.Core.Services.BluetoothService, which represents the core Bluetooth service provided by my device. I acquire this instance using the GetDefault static factory method, because there's only one Bluetooth service available per device.

For enumeration, I'll make a call to the SearchForPairedDevicesAsync method. This method will start enumerating devices paired with my device. For Windows Store 8.1 apps, I need to allow the use of a paired device to get the enumerated devices. If I block the use, that paired device won't be enumerated.

If that API succeeds, it will raise the SearchForPairedDevicesSucceeded event and fetch the collection of paired devices from its event argument. Otherwise, the SearchForPairedDevicesFailed event will be raised, with failure enum constant available in its event argument. **Figure 8** shows the code for enumerating devices.

I've also provided the scan button in BottomAppBar for Windows 8.1 and ApplicationBar for Windows Phone 8.1. That way an app user can rescan for devices when a new paired device has arrived.

Figure 8 **Enumerating Paired Devices**

```
protected async override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);
  await EnumerateDevicesAsync();
}

public async Task EnumerateDevicesAsync()
{
  BluetoothService btService = BluetoothService.GetDefault();
  btService.SearchForPairedDevicesFailed +=
    btService_SearchForPairedDevicesFailed;
  btService.SearchForPairedDevicesSucceeded +=
    btService_SearchForPairedDevicesSucceeded;
  await btService.SearchForPairedDevicesAsync();
}

void btService_SearchForPairedDevicesSucceeded(object sender,
  SearchForPairedDevicesSucceededEventArgs e)
{
  (sender as BluetoothService).SearchForPairedDevicesFailed -=
    btService_SearchForPairedDevicesFailed;
  (sender as BluetoothService).SearchForPairedDevicesSucceeded -=
    btService_SearchForPairedDevicesSucceeded;
  this.cvBtDevices.Source = e.PairedDevices;
}

void btService_SearchForPairedDevicesFailed(object sender,
  SearchForPairedDevicesFailedEventArgs e)
{
  (sender as BluetoothService).SearchForPairedDevicesFailed -=
    btService_SearchForPairedDevicesFailed;
  (sender as BluetoothService).SearchForPairedDevicesSucceeded -=
    btService_SearchForPairedDevicesSucceeded;
  txtblkErrorBtDevices.Text = e.FailureReason.ToString();
}
```

SEE THE WORLD AS A DATABASE
ADO.NET ▪ JDBC ▪ ODBC ▪ SQL SSIS ▪ ODATA
MYSQL ▪ EXCEL ▪ POWERSHELL
Visual Studio
Java
ODBC
SQL Server
Excel
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

Copyright © 2013 RSSBus, Inc. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.

When enumerating devices on Windows Phone 8.1, it will enumerate all OBEX-capable devices, regardless of their physical presence and whether the Bluetooth radio is switched on. However, when enumerating devices on Windows 8.1, it will only list those devices physically present near the Windows 8.1 device and with their Bluetooth radio switched on.

Once I enumerate the paired devices, I can select a device to share the files. Each device is represented as an object of the Bluetooth.Core.Services.BluetoothDevice class. The object contains connection details and the display name of the paired device. The Bluetooth.Services.Obex.ObexService will use the connection details internally to create an instance of Bluetooth.Core.Sockets.BluetoothSocket and connect to the paired device.

Using ObexService

Once I get the instance of the Bluetooth.Core.Services.BluetoothDevice object that represents my targeted device for sharing files, I can use Bluetooth.Services.Obex.ObexService for sharing files using OPP. I also need the list of files so I can queue them for sharing. In the code sample, I've only provided a handful of files. Otherwise, I could use Windows.Storage.Pickers.FileOpenPicker (see bit.ly/1qtiLeh) or custom logic from my Windows.Storage.ApplicationData.Current.LocalFolder (see bit.ly/1qtiSGl) to select multiple files.

As of now, I can only share one file per connection. When sharing is complete, the connection to the targeted device is closed. If I need to send multiple files, I need to get the handle to Bluetooth.Services.Obex.ObexService instance multiple times. I can acquire this instance using the static factory method GetDefaultForBluetoothDevice(Bluetooth.Core.Services.BluetoothDevice). This method returns the single instance of Bluetooth.Services.Obex.ObexService that represents the Obex service on the device.

Figure 9 **Hooking ObexService and Its Events**

```
ObexService obexService = null;
BluetoothDevice BtDevice = null;
ObservableCollection<FileItemToShare> filesToShare = null;

async override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);

  if (e.Parameter == null || !(e.Parameter is BluetoothDevice))
  {
    MessageDialog messageBox = new MessageDialog(
      "Invalid navigation detected. Moving to Main Page", "Bluetooth Hub");
    messageBox.Commands.Add(new UICommand("OK", (uiCommand) =>
    {
      this.Frame.Navigate(typeof(MainPage));
    }));
    await messageBox.ShowAsync();
    return;
  }

  BtDevice = e.Parameter as BluetoothDevice;
  filesToShare = GetFilesFromLocalStorage();
  this.cvFileItems.Source = filesToShare;

  obexService = ObexService.GetDefaultForBluetoothDevice(BtDevice);
  obexService.Aborted += obexService_Aborted;
  obexService.ConnectionFailed += obexService_ConnectionFailed;
  obexService.DataTransferFailed += obexService_DataTransferFailed;
  obexService.DataTransferProgressed += obexService_DataTransferProgressed;
  obexService.DataTransferSucceeded += obexService_DataTransferSucceeded;
  obexService.DeviceConnected += obexService_DeviceConnected;
  obexService.Disconnected += obexService_Disconnected;
  obexService.Disconnecting += obexService_Disconnecting;
  obexService.ServiceConnected += obexService_ServiceConnected;
  await obexService.ConnectAsync();
}
```

To represent the file to share, I'll use the FileItemToShare class. This contains the name, path and size of the file, and the instance of Windows.Storage.IStorageFile (see bit.ly/1qMcZlB) representing the file instance on the disk. I'll queue all the files I have to share in terms of data structure objects. When sharing multiple files, the first file in the list is the one currently being shared. It's removed from the list when sharing is complete. **Figure 9** shows how to hook up ObexService and its events for sharing files.

When I call the ConnectAsync method, the ObexService object gets the connection properties from BluetoothDevice object, which was passed in the factory method. It attempts to create a connection with the targeted BluetoothDevice over Bluetooth channel. When that's successful, it raises the DeviceConnected event. **Figure 10** shows the DeviceConnected event handler of ObexService.

As soon as the device is connected to the targeted device, I'll start sharing the file at index 0 from the list of files. The file is shared by calling SendFileAsync(Windows.Storage.IStorageFile) and passing the file object represented by IStorageFile from object of type FileToShare data structure. When this method is called, the ObexService attempts to connect to the OBEX Server running on the targeted device. If the connection is successful, it will raise the ServiceConnected event. Otherwise, it will raise ConnectionFailed event. This code shows the ServiceConnected event handler:

```
async void obexService_ServiceConnected(object sender, EventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
  {
    System.Diagnostics.Debug.WriteLine("service connected");
    filesToShare[0].ShareStatus = FileShareStatus.Sharing;
  });
}
```

Figure 11 shows the ConnectionFailed event handler of ObexService.

When my device connects to the OBEX server of the targeted device, the file sharing process begins. The progress of files shared can be determined from the DataTransferProgressed event. The following code shows the DataTransferProgressed method:

```
async void obexService_DataTransferProgressed(object sender,
  DataTransferProgressedEventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
  {
    System.Diagnostics.Debug.WriteLine("Bytes {0}, Percentage {1}",
      e.TransferInBytes, e.TransferInPercentage);
    filesToShare[0].Progress = e.TransferInBytes;
  });
}
```

Figure 10 **DeviceConnected Event Handler Method**

```
async void obexService_DeviceConnected(object sender, EventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
  {
    System.Diagnostics.Debug.WriteLine("device connected");
    if (filesToShare.Count > 0)
    {
      filesToShare.ShareStatus = FileShareStatus.Connecting;
      await obexService.SendFileAsync(filesToShare[0].FileToShare);
    }
    ...
  });
}
```

Once file sharing is complete, that raises the DataTransferSucceeded event. If file sharing is unsuccessful, it raises the DataTransferFailedEvent. The following code shows DataTransferSucceeded event handler:

```
async void obexService_DataTransferSucceeded(object sender, EventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
  {
    System.Diagnostics.Debug.WriteLine("data transfer succeeded");
    filesToShare.RemoveAt(0);
  });
}
```

In the event of a file sharing error, it will raise a DataTransferFailed event. The event handler is shown in the following code:

```
async void obexService_DataTransferFailed(object sender,
  DataTransferFailedEventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
  {
    System.Diagnostics.Debug.WriteLine("Data transfer failed {0}",
      e.ExceptionObject.ToString());
    filesToShare[0].ShareStatus = FileShareStatus.Error;
    filesToShare[0].Progress = 0;
    FileItemToShare fileToShare = this.filesToShare[0];
    filesToShare.RemoveAt(0);
    this.filesToShare.Add(fileToShare);
  });
}
```

Figure 11 **ConnectionFailed Event Handler Method**

```
async void obexService_ConnectionFailed(object sender, ConnectionFailedEventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
  {
    System.Diagnostics.Debug.WriteLine("connection failed");
    filesToShare[0].ShareStatus = FileShareStatus.Error;
    filesToShare[0].Progress = 0;
    FileItemToShare currentItem = filesToShare[0];
    filesToShare.RemoveAt(0);
    filesToShare.Add(currentItem);
  });
}
```

Figure 12 **Disconnected Event Handler Method**

```
async void obexService_Disconnected(object sender, EventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
  {
    System.Diagnostics.Debug.WriteLine("disconnected");

    obexService.Aborted -= obexService_Aborted;
    obexService.ConnectionFailed -= obexService_ConnectionFailed;
    obexService.DataTransferFailed -= obexService_DataTransferFailed;
    obexService.DataTransferProgressed -= obexService_DataTransferProgressed;
    obexService.DataTransferSucceeded -= obexService_DataTransferSucceeded;
    obexService.DeviceConnected -= obexService_DeviceConnected;
    obexService.Disconnected -= obexService_Disconnected;
    obexService.Disconnecting -= obexService_Disconnecting;
    obexService.ServiceConnected -= obexService_ServiceConnected;
    obexService = null;

    if (filesToShare.Count.Equals(0))
    {
      ...
      MessageDialog messageBox =
        new MessageDialog("All files are shared successfully",
        "Bluetooth DemoApp");
      messageBox.Commands.Add(new UICommand("OK", (uiCommand) =>
      {
        this.Frame.Navigate(typeof(MainPage));
      }));
      await messageBox.ShowAsync();
    }
    else
    {
      obexService = ObexService.GetDefaultForBluetoothDevice(BtDevice);
      obexService.Aborted += obexService_Aborted;
      obexService.ConnectionFailed += obexService_ConnectionFailed;
      obexService.DataTransferFailed += obexService_DataTransferFailed;
      obexService.DataTransferProgressed += obexService_DataTransferProgressed;
      obexService.DataTransferSucceeded += obexService_DataTransferSucceeded;
      obexService.DeviceConnected += obexService_DeviceConnected;
      obexService.Disconnected += obexService_Disconnected;
      obexService.Disconnecting += obexService_Disconnecting;
      obexService.ServiceConnected += obexService_ServiceConnected;
      await obexService.ConnectAsync();
    }
  });
}
```

When the data transfer is finished, the shared file is removed from the list and the ObexService is disconnected. When the ObexService is disconnecting, it raises the Disconnecting event. And, when the connection is disconnected properly, then the Disconnected event is raised. The Disconnecting event handler is shown here:

```
void obexService_Disconnecting(object sender, EventArgs e)
{
  System.Diagnostics.Debug.WriteLine("disconnecting");
}
```

Once the connection is successfully disconnected, the code in **Figure 12** handles raising the Disconnected event.

When the Disconnected event is raised, remove all handlers and clear out the ObexService instance. During data transfer, conditions may arise that require you to abort current transfer. To abort a current transfer, call AbortAsync. The Aborted event is raised with the following code and then the connection to the targeted device is ended:

```
async void obexService_Aborted(object sender, EventArgs e)
{
  await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
  {
    System.Diagnostics.Debug.WriteLine("Aborted");
    if (!filesToShare.Count.Equals(0))
    {
      filesToShare.RemoveAt(0);
    }
  });
}
```

Wrapping Up

The demo app is now complete. The concept of universal apps can really help you in writing a single piece of code for multiple Windows platforms and form-factors, thereby reducing the overall development effort.

I've used these libraries in a number of Windows Store and Windows Phone apps. Search for Code Create (a free Windows Phone App) or OBEX (Universal App) and get a glimpse of how these APIs work in conjunction with the app. These libraries are available to download from the NuGet repository. Simply search for "Bluetooth OBEX for Store Apps" on the NuGet online dialog, straight from the Visual Studio Solution and import these libraries as a reference to the projects. ■

Uday Gupta *is a senior engineer-product development at Symphony Teleca Corp. Pvt Ltd. in India. He has experience in many .NET technologies, especially Windows Presentation Foundation, Silverlight, Windows Phone and Windows 8.x.*

Thanks to the following Microsoft technical experts for reviewing this article: *Jeff Kelley and Guruprasad Subbarayan*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com

2015 Dates Announced

Visual Studio Live! is hittin' the open road on the ultimate code trip! For 22 years, we have helped tens of thousands of developers navigate the .NET Highway, featuring code-filled days, networking nights and the best in independent training:

- Multi-track Events
- Focused, Cutting-edge .NET Education
- Covering the Hottest Topics
- Relevant and Useable Content
- Expert Speakers, Including Many Microsoft Instructors
- Interactive Sessions
- Discuss Your Challenges
- Find Actionable Solutions

MAKE PLANS TO JOIN US IN 2015!

Redmond

Code Home
August 10 – 14
Microsoft HQ
Redmond, WA

San Francisco

Code By The Bay
June 15 – 18

Las Vegas

Code on the Strip
March 16 – 20
Bally's Hotel & Casino
Las Vegas, NV

SUPPORTED BY

Microsoft   Visual Studio   msdn magazine   Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA INC

NAVIGATE THE .NET HIGHWAY

NOW WITH 6 Locations to Choose From!

New York

The Code That Never Sleeps

September 28 – October 1

NY Marriott at Brooklyn
Bridge Brooklyn, NY

Austin

Don't Mess with Code

June 1 - 4

Hyatt Regency
Austin, TX

Orlando

Code in the Sun

PART OF LIVE! 360
Tech Events with Perspective

SharePoint Live!
SQL Server Live!
Modern Apps Live!
TechMentor

November 16 – 20

Loews Royal Pacific Resort
Orlando, FL

vslive.com

Advanced Push Notifications and Mobile Analytics

Kevin Ashley

With the increasing complexity of new mobile apps, developers are interested in the next step in push notification services—push analytics, market segmentation, reporting and reach via push available across all major platforms. Microsoft recently acquired the Capptain platform, which provides all of these services and then some.

While Capptain is not yet a part of Microsoft Azure, it's already available for developers at capptain.com. Azure provides a reliable set of services for push notifications for developers on Windows 8, Android and iOS platforms.This article covers some advanced topics in push notifications for mobile developers.

Capptain provides rich analytics that can tell you most of what you need to know about your app, including on which devices it's been used and usage trends. It can define segments, create marketing campaigns and monetize your apps with push. It's powerful technology, and it can start working for you with just a few initial steps.

Being a mobile developer myself, I quickly realized the benefits of using Capptain in conjunction with Azure Mobile Services. My Active Fitness app (activefitness.co) has a sizeable user base. This study provided me with an opportunity to experiment with Capptain and use it in the production version of my app.

This article discusses:

- Adding push notifications to mobile apps
- Tracking mobile app activity
- Reporting on app activity data

Technologies discussed:

Microsoft Azure, Windows 8.1, Android, iOS

Code download available at:

github.com/kevinash/CapptainAppSample

Capptain Concepts

In this article, I'll be using the following main Capptain concepts: activities, jobs, application information (appinfo) and extra data (extras). **Figure 1** lists most of the Capptain concepts and terms and should be a good starting point to help you understand how Capptain works.

Activity is a fundamental concept to Capptain. For example, an activity could be a page in your app the user is visiting. It could also be any logical activity that has duration. Job is another concept that also has duration, however, jobs are associated with background tasks, not necessarily connected with the UI.

Capptain provides an extensive set of APIs and native SDKs for all major platforms: iOS, Android and Windows 8 (see **Figure 2**).

Besides those APIs, Capptain provides several SDKs for all major platforms, as listed in **Figure 3**.

Get Started with Capptain

First, go to capptain.com and create an account. After creating an account, set up an app of your choice—whether on Android, iOS, Windows Phone, Windows Store (Windows 8.x) or Web. Each app has an associated SDK, which I'll show how to use after explaining some of the Capptain concepts. Capptain also provides a set of demo applications you'll find at the bottom of your account page. You can also check what kind of analytics it provides, with data already in place.

Implement Activity Tracking

To start using Capptain, start with the right SDK for your platform. I placed an example on GitHub that shows how to use the SDK for Windows 8.1. SDKs on other platforms work similarly, and use the same basic concepts.

In the Package.appxmanifest, you need to ensure Internet capability is enabled. Go to the Declarations panel of your Package.appxmanifest

Figure 1 **Capptain Terms and Concepts**

Device	Each device gets a unique identifier. If you have several apps on the same device, the device identifier (deviceid) will be the same. (On Windows Phone, device ID is unique per device and publisher).
User	Capptain implicitly assigns one user to one device, so devices and users are equivalent concepts.
Session	A session is one use of the application performed by a user. Sessions are automatically computed from a sequence of activities performed by a user. There's no need to start/stop a session. Instead, you can start/stop activities. If no activity is reported, no session is reported.
Activity	An activity is one use of a given portion of the application performed by one user (usually a screen, but it can be anything suitable to the application). A user can only perform one activity at a time. An activity has duration—from the moment it's started to the moment it's stopped.
Event	An event is an instant action; unlike an activity, it doesn't have duration.
Job	A job is like an activity. You can start/stop it and it has duration. A job is intended for a background task, it might not have a UI.
Error	An error is an issue correctly reported by the application.
Crash	The Capptain SDK automatically reports a crash and an application failure.
Application Information (AppInfo)	This is used to tag users (similar to cookies). For one given key, Capptain only keeps track of the latest value set (no history). Setting or changing the value of an appinfo forces Capptain to reevaluate audience criteria set on this appinfo (if any), meaning that appinfo can be used to trigger real-time pushes.
Extra Data (Extras)	Extra data (or extras) is some arbitrary data you can attach to an event, error, activity and job. You can use this data to create ways to identify segments of your activities, jobs and so on.

Figure 2 **Capptain Provides an Extensive Set of APIs**

Analytics API	The Analytics API is an HTTP API, which lets you retrieve analytics data (the one displayed on the Analytics tab of the Capptain Web site).
Monitor API	The Monitor API is an XMPP API, which lets you retrieve real-time monitor data (the one displayed on the Monitor tab of the Capptain Web site).
Segments API	The Segments API is an HTTP API, which lets you manage Capptain segments (everything under the Segments tab of the Capptain Web site).
Reach API	The Reach API is an HTTP API, which lets you manage Reach campaigns without having to use the Capptain Web interface manually. The Reach API is a high-level API so you can leverage the Web interface of the Capptain Reach campaign manager.
Device API	The Device API is an HTTP/REST API, which lets you retrieve and enrich the information gathered by the Capptain platform about all devices (and users) using your application.
Push API	The Push API is an HTTP API, which lets you push custom data to devices running an application embedding the Capptain SDK.
SDK API	The SDK API is an HTTP API, which lets you report logs as a native SDK would do, but using a simple HTTP API.
Account API	The Account API is a set of HTTP APIs aimed at retrieving or updating account-related information.

file. In Available Declarations, select and add File Type Associations. In the right screen, in the Name field, type capptain_log_file and in the File type field type .set. Then, in Available Declarations select and add Cached File Updater.

Next, use NuGet to fetch the latest version of Capptain for Windows 8.1. In the application OnLaunched event, add the initialization code for Capptain. Copy your app ID and SDK key from the Capptain portal:

```
/* Capptain configuration. */
CapptainConfiguration capptainConfiguration = new CapptainConfiguration();
capptainConfiguration.Agent.ApplicationId = "YOUR_APPID";
capptainConfiguration.Agent.SDKKey = "YOUR_SDK_KEY";

/* Initialize Capptain angent with above configuration. */
CapptainAgent.Instance.Init(e, capptainConfiguration);
```

For my app, I wanted Capptain to track pages and how much time the user spends on each page. The following snippet, when placed in the page OnNavigatedTo method, will start tracking HubPage as a new activity in Capptain:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);
  CapptainAgent.Instance.StartActivity("HubPage");
}
```

Remember, in Capptain you can only have one activity at a time per user per app. You don't need to call EndActivity. The new page calls StartActivity, and the old activity automatically ends. If you have many pages, you can simplify this approach by deriving from a special class called CapptainPage. This will automatically associate your pages with new activities.

Another way to track your activities with Capptain is by deriving your page from CapptainPage. The advantage to this method is you don't need to StartActivity manually in the OnNavigatedTo handler. The CapptainPage will handle it for you. To do it this way, simply insert the following code into your XAML:

```
<capptain:CapptainPage
  xmlns:capptain="using:Capptain.Agent">
  <!-- layout -->
  ...
</capptain:CapptainPage >
```

With this code, I called the StartActivity method with HubPage. When you use the CapptainPage, you don't need to call StartActivity. It will track the name of your page by default. You can always override GetCapptainPageName to report a different value.

```
// In the .xaml.cs file
protected override string GetCapptainPageName()
{
  /* your code */
  return "new name";
}
```

Use Reach with Push Notifications

Capptain is especially interesting for mobile apps because it lets you push notifications to user devices using Capptain Reach. You can set up campaigns and then monitor how they're delivered. For the announcements you push, you can define campaign name, content (including images), define your audience, timing and so on.

Capptain Reach is a powerful feature, as it lets you drive advertising campaigns, promotions and other types of marketing campaigns directly to your users. The key to successful marketing is relevance. Thanks to the ability of Capptain to know exactly what page your user is on, or what activity your user is doing, each campaign can be more relevant and more effective.

Figure 3 **Capptain Provides SDKs to Support All Major Platforms**

Android SDK	Native Android SDK
iOS SDK	Native iOS SDK
Web	Web SDK
Windows Phone	Windows Phone SDK
Windows 8	Windows Store SDK

For Windows Store apps, Capptain uses the Windows Notifications Service (WNS). You need to update your Package manifest and update the Capptain portal with the WNS key. There are two scenarios in which you can integrate these notifications into your app—via Overlay or WebView integration. After you've started a Capptain account, you can refer to the Capptain documentation for in-depth integration guidance on "How to Integrate Capptain Reach on Windows" at bit.ly/12b3bub and "Initialize the Capptain Reach SDK" at bit.ly/1w90J3M.

Capptain is especially interesting for mobile apps, because it lets you push notifications to user devices.

Overlay Integration

With overlay integration, you derive from the CapptainPageOverlay page. The notification will be delivered automatically, using resources included in the NuGet package in the Resources/Overlay directory. When a notification comes from Reach API, your app page will inject the announcement embedded in CapptainOverlayAnnouncement or CapptainOverlayNotification views. Each of these views effectively contains a WebView that displays the announcement. You can further customize these views if you want to implement your own presentation for the Reach notification.

To get started, include the following declaration in any XAML page in your project (I implemented this method in the ItemPage.xaml of the sample project included with this article):

```
xmlns:capptain="using:Capptain.Overlay"
```

In XAML, instead of <Page>, make your page derive from CapptainPageOverlay, as follows:

```
public sealed partial class ItemPage :
  CapptainPageOverlay
```

Capptain Reach will inject its notification views in the first grid it finds on your page. If you want a specific grid to receive the view, you can use the grid named CapptainGrid:

```
<Grid x:Name="CapptainGrid"></Grid>
```

WebView Integration

For WebView integration, you need to include WebViews named capptain_notification_content or capptain_announcement_content, depending on the type of content you need to receive.

Use the following code to insert the WebViews:

```
<capptain:CapptainPage
  xmlns:capptain="using:Capptain.Page">
  <Grid>
    <WebView x:Name="capptain_notification_content" Visibility="Collapsed"
      ScriptNotify="scriptEvent" Height="64" HorizontalAlignment="Right"
      VerticalAlignment="Top"/>
    <WebView x:Name="capptain_announcement_content" Visibility="Collapsed"
      ScriptNotify="scriptEvent" HorizontalAlignment="Right"
      VerticalAlignment="Top"/>
    <!-- layout -->
  </Grid>
</capptain:CapptainPage>
```

Data Push Notifications

Besides receiving visual content in your app with the Overlay and WebView Integration methods, you can also receive data push notifications directly in your app. To do so, you need to implement two handlers. The best place to put them is the constructor of your App object, in App.cs, as shown in **Figure 4**.

Figure 4 **Implementing Handlers in the Constructor of the App Object**

```
CapptainReach.Instance.DataPushStringReceived += (body) =>
{
  Debug.WriteLine("String data push message received: " + body);
  return true;
};

CapptainReach.Instance.DataPushBase64Received += (decodedBody, encodedBody) =>
{
  Debug.WriteLine("Base64 data push message received: " + encodedBody);
  // Do something useful with decodedBody like updating an image view
  return true;
};

CapptainReach.Instance.PushMessageReceived += (id, replyTo, payload) =>
{
  // Your code
};
```

Analyze Results

Capptain provides a rich view of your data. You can view technical information associated with your devices, such as manufacturer, OS version, firmware, screen resolution and SDK version. Now that your app sends activity information to Capptain, you can also track any technical details you like.

Wrapping Up

Push notifications are powerful mechanisms. Most platforms currently provide notification service mechanisms, such as WNS, ANS and Google Cloud Messaging. Azure also provides a scalable infrastructure called Notification Hubs. As mobile apps evolve, there's a need in analytics, reach, advertising and coordinating marketing campaigns with push. Capptain provides this next-level push notifications analytics tier many mobile developers will need. ■

Kevin Ashley *is an architect evangelist for Microsoft. He's coauthor of "Professional Windows 8 Programming" (Wrox, 2012) and a developer of top apps and games. He often presents on technology at various events, industry shows and Webcasts. In his role, he works with startups and partners, advising on software design, business and technology strategy, architecture, and development. Follow his blog at kevinashley.com and on Twitter at twitter.com/kashleytwit.*

Thanks to the following Microsoft technical experts for reviewing this article: *Greg Oliver and Bruno Terkaly*

```
40, 0); AddEAN13Sup5(page.Elements, 480, 0); AddUPCVersionA(page.Elements, 0, 135); AddUPCVersionASup2(page.Elements, 240, 135); AddUPCVersionASup5(page.Elements, 480, 135); AddEAN8(page.Elements
ements, 480, 270); AddUPCVersionE(page.Elements, 0, 405); AddUPCVersionESup2(page.Elements, 240, 405); AddUPCVersionESup5(page.Elements, 480, 405); // Add the page to the document document.Pages.Add
tionAndRectangle(elements, "EAN/JAN 13 Bar Code", x, y, 204, 99); BarCode barCode = new Ean13("123456789012", x, y + 21); barCode.X += (204 - barCode.GetSymbolWidth()) / 2; elements.Add(barCode); } privat
ctangle(elements, "EAN/JAN 13 Bar Code, 2 digit supplement", x, y, 204, 99); BarCode barCode = new Ean13Sup2("12345678901212", x, y + 21); barCode.X += (204 - barCode.GetSymbolWidth()) / 2; elements.Add(
float x, float y) { AddCaptionAndRectangle(elements, "EAN/JAN 13 Bar Code, 5 digit supplement", x, y, 204, 99); static void Main(string[] args) BarCode barCode = new Ean13Sup5("12345678901212345", x, y + 21)
dd(barCode); } private void AddUPCVersionA(Group elements, float x, float y) {AddCaptionAndRectangle(elements,{ "UPC Version A Bar Code", x, y, 204, 99); BarCode barCode = new UpcVersionA("12345678901", x
dd(barCode); } private void AddUPCVersionASup2(Group elements, float x, float y) { AddCaptionAndRectangle(element //Open PDF and fill in Fields "UPC Version E Bar Code, 2 digit supplement", x, y, 204, 99); BarC
; barCode.X += (204 - barCode.GetSymbolWidth()) / 2; elements.Add(barCode); } private void AddUPCVersionASup5( MergeDocument document = new MergeDocument(@"C:\Input.pdf"); Group elements, float x, fl
t supplement", x, y, 204, 99); BarCode barCode = new UpcVersionASup5("1234567890112345", x, y + 21); barCode.X document.Form.Fields["Name"].Value = "John Smith"; += (204 - barCode.GetSymbolWidth()) / 2
float y) { AddCaptionAndRectangle(elements, "EAN/JAN 8 Bar Code", x, y, 204, 99); BarCode barCode = new Ean8("1 document.Form.Fields["Title"].Value = "President"; 2345670", x, y + 21); OpenFileDialog fileDialo
ter = "Adobe PDF files (*.pdf)|*.pdf|All Files (*.*)|*.*"; if (fileDialog.ShowDialog() == DialogResult.OK) { pdfViewer.Open // Specify Security/Encryption (fileDialog.FileName, ""); } SaveFileDialog saveFileDialog = new
log.Filter = "Adobe PDF files (*.pdf)|*.pdf|All Files (*.*)|*.*"; if (saveFileDialog.ShowDialog() ==DialogResult.OK) { pdfView HighSecurity security = new HighSecurity("OwnerPassword", "UserPassword"); .SaveAs(saveFil
ter = pdfViewer.Printer; printer.PrintWithDialog(); } else { MessageBox.Show("Please open a file to print"); }OpenFile security.AllowCopy = false; Dialog fileDialog = new OpenFileDialog(); fileDialog.Title = "Open Fil
(*.*)|*.*|Adobe PDF files (*.pdf)|*.pdf"; if (fileDialog.ShowDialog() == DialogResult.OK) { DynamicPDFViewerClass test = document.Security = security; new DynamicPDFViewerClass(); PDFPrinter printer = test.OpenFi
e.ReadAllBytes(@"C:\MyDoc.pdf"); //using System.Runtime.InteropServices;GCHandle gch=GCHandle.Aloc(contents, // Save Updated PDF and Send to default Printer GCHandleType.Pinned); IntPtr contentsIntPtr
s.Length, ""); AddCaptionAndRectangle(pageElements, "Bookmark Page Element:", x, y); pageElements.Add(newtest document.Draw(@"C:\Output.pdf"); Bookmark("Bookmarked Text", x + 5, y + 20, parentOutline));
, 10, Font.TimesRoman, 10)); } private void AddCircle(Group pageElements, float x, float y) { //Adds a circlt to the page PrintJob printJob = new PrintJob(Printer.Default, @"C:\Output.pdf"); Elements AddCaptionAndR
d(new Circle(x + 112.5f, y + 50f, 107.5f, 30f, RgbColor.Red, RgbColor.Blue, 2, LineStyle.DashLarge)); } private void Add printJob.Print(); FormattedTextArea(Group pageElements, float x, float y) { // Adds a formatted te
</i><b>PDF</b>&tm; Generator v6.0 for .NET has a formatted text area page " + "element. This provides rich form } atting support for text that appears in the document. You have " + "complete control over 8 paragra
indentation, right indentation, alignment, allowing orphan lines, and white " + "space preservation; 6 font properties: <font face='Times'>font face, </font><font pointSize='6'>font " + "size, </font><font color='FF000
ne properties: leading, and leading type. Text can also be rotated.</p>"; FormattedTextArea formattedTextArea = new FormattedTextArea(formattedHtml, x + 5, y + 20, 215, 60, FontFamily.Helvetica, 9, false); // Sets t
CaptionAndRectangle(pageElements, "FormattedTextArea Page Element:", x, y); AddCaptionAndRectangle(pageElements, "FormattedTextArea Overflow Text:", x + 279, y); pageElements.Add(formattedTextArea); // Cr
ea overflowFormattedTextArea = formattedTextArea.GetOverflowFormattedTextArea(x + 284, y + 20); pageElements.Add(overflowFormattedTextArea); } private void AddImage(Group pageElements, float x, float y) { //
, "Image Page Element:", x, y); Image image = new Image(Server.MapPath("../Images/DPDFLogo.png"), x + 112.5f, y + 50f, 0.24f); // Image is sized and centered in the rectangle image.SetBounds(215, 60); image.VA
} private void AddLabel(Group pageElements, float x, float y) { // Adds a label to the pageElements AddCaptionAndRectangle(pageElements, "Label & PageNumberingLabel Page Elements:", x, y); string labelText = "
page numbering: %%CP%% of %%TP%% pages."; Label label = new Label(labelText, x + 5, y + 12, 220, 80, Font.TimesRoman, 12, TextAlign.Center); label.Angle = 8; PageNumberingLabel pageNumLabel = new P
, TextAlign.Center); pageElements.Add(pageNumLabel); pageElements.Add(label); } private void AddLine(Group pageElements, float x, float y) { // Adds a line to the pageElements AddCaptionAndRectangle(pageElem
 y + 20, x + 220, y + 80, 3, RgbColor.Green)); pageElements.Add(new Line(x + 220, y + 20, x + 5, y + 80, 3, RgbColor.Green)); } private void AddLink(Group pageElements, float x, float y) { // Adds a link to the pageEle
micPDF.com." w Label(text, x + 5, y + 20, 215, 80, font, 12, RgbColor.Blue); label.Underline = true; Link link = new Link(x + 5, y +
"http://www.d Path(Group pageElements, float x, float y) { // Adds a path to the pageElements ceTe.DynamicPDF.PageElements.P
RgbColor.Blue , y + 40)); path.SubPaths.Add(new CurveToSubPath(x + 108, y + 80, x + 160, y + 80)); path.SubPaths.Add(new Cur
ngle(pageEle le(Group pageElements, float x, float y) orderedList = orderedList.GetOverFlowList(x + 5, y + 20); AddCaptionAndR
, 110); page.E orderedList = new UnorderedList(x + 5, y + 20, 400, 90, Font.Helvetica, 10); unorderedList.Items.Add("Fruits"); uno
es(); pieSeries.DataLabel = da; plotArea.Series.Add(pieSeries); pieSeries.Elements.Add(27, "Website A"); pieSeries.Elements.Add(19, "Website B"); pieSeries.Elements.Add(21, "Website C"); pieSeries.Elements[0].Colo
lements[2].Co ubList2.Items.Add("Potato"); unorderedSubList2.Items.Add("Beans"); UnorderedSubList su
ubList(); subU redSubList2 = unorderedSubList.Items[1].SubLists.AddUnorderedSubList(); subUnordered
"); Unordered SubList3.Items.Add("Sweet Potato"); UnorderedSubList subUnorderedSubList4 = unordere
List4.Items.Ac Bean"); x += 279; page.Elements.Add(unorderedList); unorderedList = unorderedList.GetOve
ements, "Uno Field(Group pageElements, float x, float y) { TextField txt = new TextField("txtfname", x + 20,
dTextArea(formattedHtml, x + 5, y + 20, 215, 60, FontFamily.Helvetica, 9, false); // Sets the indent property formattedTextArea.Style.Paragraph.Indent = 18; AddCaptionAndRectangle(pageElements, "FormattedTextAr
dTextArea Ov formatted text area for the overflow text FormattedTextAreaMaxLength = 9; txt1.BorderColor = RgbColor.Black; txt
imum Length 150, 40); txt2.DefaultValue = "This is a TextField which goes to the next line if the text exceeds width"; txt2.MultiLin
bColor.AliceB Elements, "TextField Form Page Element:", x, y, 504, 85); } private void AddComboField(Group pageElements, float
+ 40, 150, 20) t = Font.Helvetica; cb.FontSize = 12; cb.Items.Add("Item 1"); cb.Items.Add("Item 2"); cb.Items.Add("Item 3"); cb.Ite
"].Selected = omboBox cb1 = new ComboBox("cmb1name", x + 303, y + 40, 150, 20); cb1.BorderColor = RgbColor.Black; cb1.
ntSize = 12; cb1.Items.Add("Item 1"); cb1.Items.Add("Item 2"); cb1.Items.Add("Item 3"); cb1.Items.Add("Item 4"); cb1.Items.Add("Non-Editable"); cb1.Items["Non-Editable"].Selected = true; cb1.Editable = false; cb1.T
.Convert("http://www.google.com", "Output.pdf");Converter.Convert(GetDocPath("DocumentA.rtf"), "Output.pdf");System.Diagnostics.Process.Start("Output.pdf"); AsyncConverter aConverter = new AsyncConverter(
ed); aConverter.ConversionError += new ConversionErrorEventHandler(aConverter_ConversionError); aConverter.Convert(@"C:\temp\DocumentA.rtf", @"C:\temp\OutputA.pdf"); aConverter.Convert(@"C:\temp\Docume
cumentC.rtf", @"C:\temp\OutputC.pdf"); aConverter.Convert("http://www.yahoo.com", @"C:\Temp\yahoo.pdf"); ConversionOptions options = new ConversionOptions(720, 720, 72, true); ceTe.DynamicPDF.Conversio
put.pdf", options); ceTe.DynamicPDF.Conversion.Converter.Convert("C:\\temp\\Document2.docx", "C:\\temp\\output.pdf", options); string sampleHtml = "<html><body><p>This is a very simple HTML string including
```

PRECISELY PROGRAMMED FOR SPEED

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

ceTesoftware

Speech Recognition with .NET Desktop Applications

James McCaffrey

With the introduction of Windows Phone Cortana, the speech-activated personal assistant (as well as the similar she-who-must-not-be-named from the Fruit company), speech-enabled applications have taken an increasingly important place in software development. In this article, I'll show you how to get started with speech recognition and speech synthesis in Windows console applications, Windows Forms applications, and Windows Presentation Foundation (WPF) applications.

Note that you can also add speech capabilities to Windows Phone apps, ASP.NET Web apps, Windows Store apps, Windows RT apps and Xbox Kinect, but the techniques to do so are different from those presented in this article.

A good way to see what this article will explain is to take a look at the screenshots of two different demo programs in **Figure 1** and **Figure 2**. After the console application in **Figure 1** was launched, the app immediately spoke the phrase "I am awake." Of course, you can't hear the demo while reading this article, so the demo program displays the text of what the computer is saying. Next, the user spoke the command "Speech on." The demo echoed the text that was recognized, and then, behind the scenes, enabled the application to listen for and respond to requests to add two numbers.

The user asked the application to add one plus two, then two plus three. The application recognized these spoken commands and gave the answers out loud. I'll describe more useful ways to use speech recognition later.

The user then issued the command "Speech off," which deactivated listening for commands to add numbers, but didn't completely deactivate speech recognition. With speech off, the next spoken command to add one plus two was ignored. Finally, the user turned speech back on, and spoke the nonsense command, "Klatu barada nikto," which the application recognized as the command to completely deactivate speech recognition and exit the application.

Figure 2 shows a dummy speech-enabled Windows Forms application. The application recognizes spoken commands, but

This article discusses:

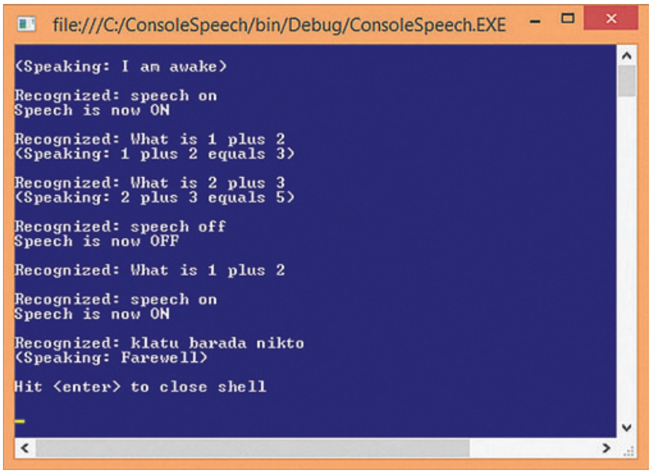
- Adding speech to a console application
- Handling recognized speech
- Installing the speech libraries
- Microsoft.Speech vs. System.Speech
- Adding speech recognition to a Windows Forms application

Technologies discussed:

Visual Studio, C#, .NET Speech Libraries

Code download available at:

msdn.microsoft.com/magazine/msdnmag1214



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

```
(Speaking: I am awake)

Recognized: speech on
Speech is now ON

Recognized: What is 1 plus 2
(Speaking: 1 plus 2 equals 3)

Recognized: What is 2 plus 3
(Speaking: 2 plus 3 equals 5)

Recognized: speech off
Speech is now OFF

Recognized: What is 1 plus 2

Recognized: speech on
Speech is now ON

Recognized: klatu barada nikto
(Speaking: Farewell)

Hit <enter> to close shell
```

Figure 1 **Speech Recognition and Synthesis in a Console Application**

doesn't respond with speech output. When the application was first launched, the Speech On checkbox control wasn't checked, indicating speech recognition wasn't active. The user checked the Speech On control and then spoke, "Hello." The application echoed the recognized spoken text in the ListBox control at the bottom of the application.

The user then said, "Set text box 1 to red." The application recognized, "Set text box 1 red," which is almost—but not quite—exactly what the user spoke. Although not visible in **Figure 2**, the text in the TextBox control at the top of the application was in fact set to "red."

Next, the user spoke, "Please set text box 1 to white." The application recognized "set text box 1 white" and did just that. The user concluded by speaking, "Good-bye," and the application echoed the command, but didn't manipulate the Windows Forms, although it could have, for example, by unchecking the Speech On checkbox control.

Using a synthesizer object is quite simple.

In the sections that follow, I'll walk you through the process of creating both demo programs, including the installation of the required .NET speech libraries. This article assumes you have at least intermediate programming skills, but doesn't assume you know anything about speech recognition or speech synthesis.

Adding Speech to a Console Application

To create the demo shown in **Figure 1**, I launched Visual Studio and created a new C# console application named ConsoleSpeech. I have successfully used speech with Visual Studio 2010 and 2012, but any recent version should work. After the template code loaded into the editor, in the Solution Explorer window I renamed file Program.cs to the more descriptive ConsoleSpeechProgram.cs and then Visual Studio renamed class Program for me.

Next, I added a Reference to file Microsoft.Speech.dll, which was located at C:\ProgramFiles (x86)\Microsoft SDKs\Speech\v11.0\Assembly. This DLL was not on my host machine and had to be downloaded. Installing the files necessary to add speech recognition and synthesis to an application is not entirely trivial. I'll explain the installation process in detail in the next section of this article, but for now, assume that Microsoft.Speech.dll exists on your machine.

After adding the reference to the speech DLL, at the top of the source code I deleted all using statements except for the one that points to the top-level System namespace. Then, I added using statements to namespaces Microsoft.Speech.Recognition, Microsoft.Speech.Synthesis and System.Globalization. The first two namespaces are associated with the speech DLL. Note: Somewhat confusingly, there are also System.Speech.Recognition and System.Speech.Synthesis namespaces. I'll explain the difference shortly. The Globalization namespace was available by default and didn't require adding a new reference to the project.

The entire source code for the console application demo is shown in **Figure 3**, and is also available in the code download that accompanies this article. I removed all normal error checking to keep the main ideas as clear as possible.

After the using statements, the demo code begins like so:

```
namespace ConsoleSpeech
{
  class ConsoleSpeechProgram
  {
    static SpeechSynthesizer ss = new SpeechSynthesizer();
    static SpeechRecognitionEngine sre;
    static bool done = false;
    static bool speechOn = true;

    static void Main(string[] args)
    {
...
```

The class-scope SpeechSynthesizer object gives the application the ability to speak. The SpeechRecognitionEngine object allows the application to listen for and recognize spoken words or phrases. The Boolean variable "done" determines when the entire application is finished. Boolean variable speechOn controls whether the application is listening for any commands other than a command to exit the program.

The idea here is that the console application doesn't accept typed input from the keyboard, so the application is always listening for commands. However, if speechOn is false, only the command to exit the program will be recognized and acted on; other commands will be recognized but ignored.

The Main method begins:

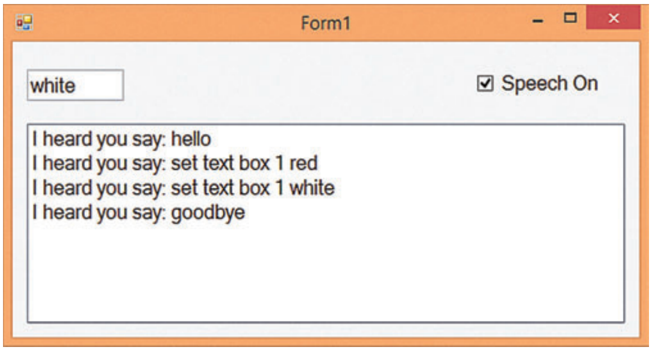
```
try
{
  ss.SetOutputToDefaultAudioDevice();
  Console.WriteLine("\n(Speaking: I am awake)");
  ss.Speak("I am awake");
```

The SpeechSynthesizer object was instantiated when it was declared. Using a synthesizer object is quite simple. The SetOutputToDefaultAudioDevice method sends output to your machine's speakers (output can also be sent to a file). The Speak method accepts a string and then, well, speaks. It's that easy.

Speech recognition is much more difficult than speech synthesis. The Main method continues by creating the recognizer object:

```
CultureInfo ci = new CultureInfo("en-us");
sre = new SpeechRecognitionEngine(ci);
sre.SetInputToDefaultAudioDevice();
sre.SpeechRecognized += sre_SpeechRecognized;
```

First, the language to recognize is specified, United States English in this case, in a CultureInfo object. The CultureInfo object is located in the Globalization namespace that was referenced with a using statement. Next, after calling the SpeechRecognitionEngine constructor, voice input is set to the default audio device, a micro-



white

```
I heard you say: hello
I heard you say: set text box 1 red
I heard you say: set text box 1 white
I heard you say: goodbye
```

Figure 2 **Speech Recognition in a Windows Forms Application**

phone in most situations. Note that most laptops have a built-in microphone, but most desktop machines will need an external microphone (often combined with a headset these days).

The key method for the recognizer object is the SpeechRecognized event handler. When using Visual Studio, if you type "sre.SpeechRecognized +=" and wait just a fraction of a second, the IntelliSense feature will auto-complete your statement with "sre_SpeechRecognized" for the name of the event handler. I recommend hitting the tab key to accept and use that default name.

Next, the demo sets up the ability to recognize commands to add two numbers:

```
Choices ch_Numbers = new Choices();
ch_Numbers.Add("1");
ch_Numbers.Add("2");
ch_Numbers.Add("3");
ch_Numbers.Add("4"); // Technically Add(new string[] { "4" });
GrammarBuilder gb_WhatIsXplusY = new GrammarBuilder();
gb_WhatIsXplusY.Append("What is");
gb_WhatIsXplusY.Append(ch_Numbers);
gb_WhatIsXplusY.Append("plus");
gb_WhatIsXplusY.Append(ch_Numbers);
Grammar g_WhatIsXplusY = new Grammar(gb_WhatIsXplusY);
```

The three key objects here are a Choices collection, a GrammarBuilder template and the controlling Grammar. When I'm designing recognition Grammar, I start by listing some specific examples of what I want to recognize. For example, "What is one plus two?" and, "What is three plus four?"

Then, I determine the corresponding general template, for example, "What is <x> plus <y>?" The template is a GrammarBuilder and the specific values that go into the template are the Choices. The Grammar object encapsulates the template and choices.

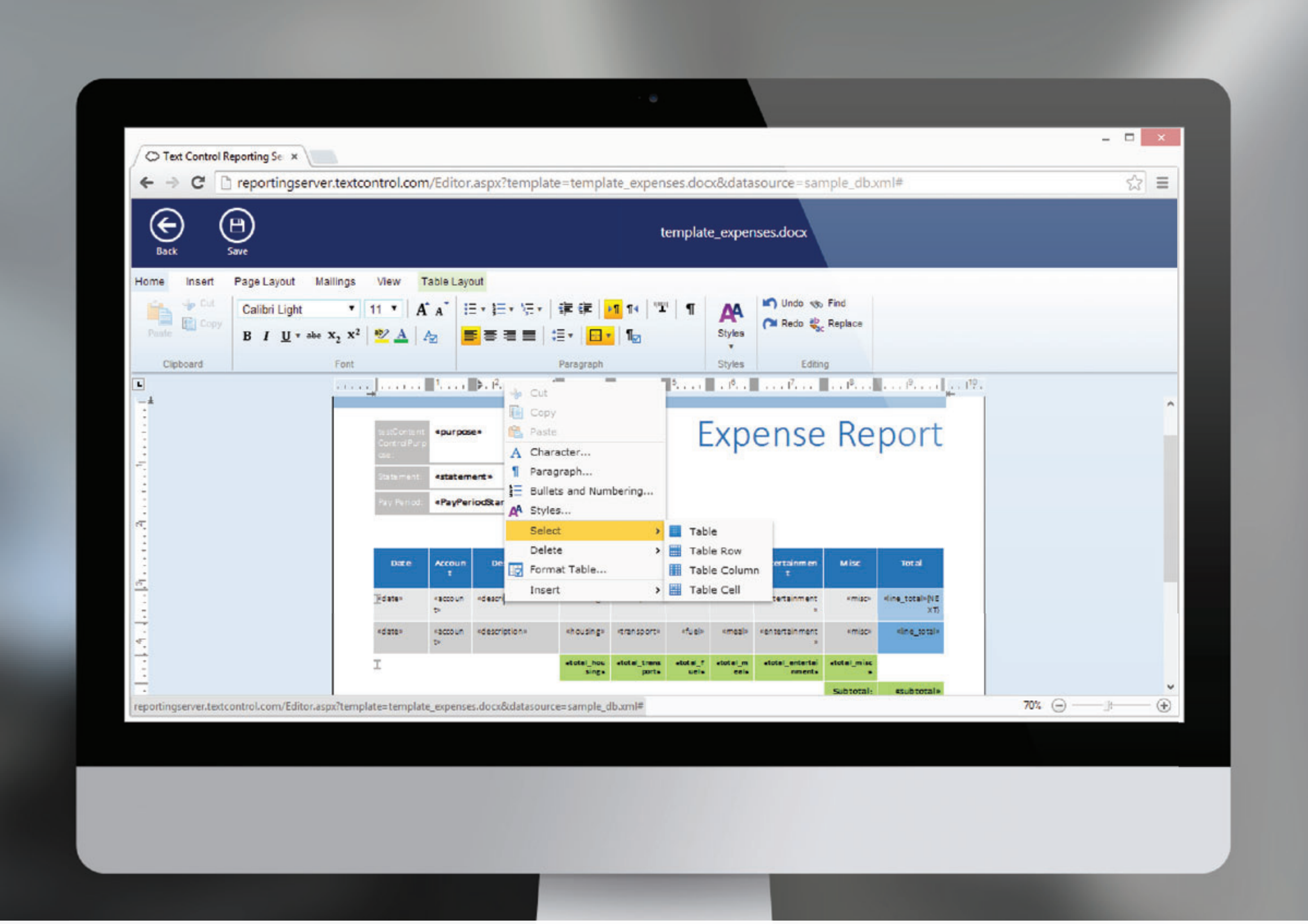
In the demo, I restrict the numbers to add to 1 through 4, and add them as strings to the Choices collection. A better approach is:

```
string[] numbers = new string[] { "1", "2", "3", "4" };
Choices ch_Numbers = new Choices(numbers);
```

I present the weaker approach to create a Choices collection for two reasons. First, adding one string at a time was the only approach I saw in other speech examples. Second, you might think that adding one string at a time shouldn't even work; the real-time Visual Studio IntelliSense shows that one of the Add overloads accepts a parameter of type "params string[] phrases." If you didn't notice the params keyword you might think

Figure 3 **Demo Console Application Source Code**

```
using System;
using Microsoft.Speech.Recognition;
using Microsoft.Speech.Synthesis;
using System.Globalization;
namespace ConsoleSpeech
{
  class ConsoleSpeechProgram
  {
    static SpeechSynthesizer ss = new SpeechSynthesizer();
    static SpeechRecognitionEngine sre;
    static bool done = false;
    static bool speechOn = true;

    static void Main(string[] args)
    {
      try
      {
        ss.SetOutputToDefaultAudioDevice();
        Console.WriteLine("\n(Speaking: I am awake)");
        ss.Speak("I am awake");

        CultureInfo ci = new CultureInfo("en-us");
        sre = new SpeechRecognitionEngine(ci);
        sre.SetInputToDefaultAudioDevice();
        sre.SpeechRecognized += sre_SpeechRecognized;

        Choices ch_StartStopCommands = new Choices();
        ch_StartStopCommands.Add("speech on");
        ch_StartStopCommands.Add("speech off");
        ch_StartStopCommands.Add("klatu barada nikto");
        GrammarBuilder gb_StartStop = new GrammarBuilder();
        gb_StartStop.Append(ch_StartStopCommands);
        Grammar g_StartStop = new Grammar(gb_StartStop);

        Choices ch_Numbers = new Choices();
        ch_Numbers.Add("1");
        ch_Numbers.Add("2");
        ch_Numbers.Add("3");
        ch_Numbers.Add("4");

        GrammarBuilder gb_WhatIsXplusY = new GrammarBuilder();
        gb_WhatIsXplusY.Append("What is");
        gb_WhatIsXplusY.Append(ch_Numbers);
        gb_WhatIsXplusY.Append("plus");
        gb_WhatIsXplusY.Append(ch_Numbers);
        Grammar g_WhatIsXplusY = new Grammar(gb_WhatIsXplusY);

        sre.LoadGrammarAsync(g_StartStop);
        sre.LoadGrammarAsync(g_WhatIsXplusY);
        sre.RecognizeAsync(RecognizeMode.Multiple);

        while (done == false) { ; }

        Console.WriteLine("\nHit <enter> to close shell\n");
        Console.ReadLine();
      }
      catch (Exception ex)
      {
        Console.WriteLine(ex.Message);
        Console.ReadLine();
      }
    } // Main

    static void sre_SpeechRecognized(object sender,
      SpeechRecognizedEventArgs e)
    {
      string txt = e.Result.Text;
      float confidence = e.Result.Confidence;
      Console.WriteLine("\nRecognized: " + txt);

      if (confidence < 0.60) return;
      if (txt.IndexOf("speech on") >= 0)
      {
        Console.WriteLine("Speech is now ON");
        speechOn = true;
      }
      if (txt.IndexOf("speech off") >= 0)
      {
        Console.WriteLine("Speech is now OFF");
        speechOn = false;
      }
      if (speechOn == false) return;
      if (txt.IndexOf("klatu") >= 0 && txt.IndexOf("barada") >= 0)
      {
        ((SpeechRecognitionEngine)sender).RecognizeAsyncCancel();
        done = true;
        Console.WriteLine("(Speaking: Farewell)");
        ss.Speak("Farewell");
      }

      if (txt.IndexOf("What") >= 0 && txt.IndexOf("plus") >= 0)
      {
        string[] words = txt.Split(' ');
        int num1 = int.Parse(words[2]);
        int num2 = int.Parse(words[4]);
        int sum = num1 + num2;
        Console.WriteLine("(Speaking: " + words[2] + " plus " +
          words[4] + " equals " + sum + ")");
        ss.SpeakAsync(words[2] + " plus " + words[4] +
          " equals " + sum);
      }
    } // sre_SpeechRecognized
  } // Program
} // ns
```



Cross-browser, cross-platform document and template editing

HTML5-BASED TX TEXT CONTROL

The first true WYSIWYG, HTML5-based Web editor and reporting template designer for ASP.NET.

Give your users an MS Word compatible editor to create powerful reporting templates anywhere - in any browser on any device.

Download your 30-day trial version today:
www.textcontrol.com/html5

the new TEXT CONTROL

Visual Studio Partner

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

the Add method accepts only an array of strings, rather than either an array of type string or a single string. I recommend passing an array.

Creating a Choices collection of consecutive numbers is somewhat a special case, and allows a programmatic approach like this:

```
string[] numbers = new string[100];
for (int i = 0; i < 100; ++i)
  numbers[i] = i.ToString();
Choices ch_Numbers = new Choices(numbers);
```

After creating the Choices to fill in the slots of the GrammarBuilder, the demo creates the GrammarBuilder and then the controlling Grammar, like so:

```
GrammarBuilder gb_WhatIsXplusY = new GrammarBuilder();
gb_WhatIsXplusY.Append("What is");
gb_WhatIsXplusY.Append(ch_Numbers);
gb_WhatIsXplusY.Append("plus");
gb_WhatIsXplusY.Append(ch_Numbers);
Grammar g_WhatIsXplusY = new Grammar(gb_WhatIsXplusY);
```

The demo uses a similar pattern to create a Grammar for start- and stop-related commands:

```
Choices ch_StartStopCommands = new Choices();
ch_StartStopCommands.Add("speech on");
ch_StartStopCommands.Add("speech off");
ch_StartStopCommands.Add("klatu barada nikto");
GrammarBuilder gb_StartStop = new GrammarBuilder();
gb_StartStop.Append(ch_StartStopCommands);
Grammar g_StartStop = new Grammar(gb_StartStop);
```

You have a lot of flexibility when defining grammars. Here, the commands "speech on," "speech off," and "klatu barada nikto" are all placed in the same grammar, because they're logically related. The three commands could've been defined in three separate grammars, or you can put the "speech on" and "speech off" commands in one grammar and the "klatu barada nikto" command in a second grammar.

After all the Grammar objects have been created, they're passed to the speech recognizer, and speech recognition is activated:

```
sre.LoadGrammarAsync(g_StartStop);
sre.LoadGrammarAsync(g_WhatIsXplusY);
sre.RecognizeAsync(RecognizeMode.Multiple);
```

The RecognizeMode.Multiple argument is required when you have more than one grammar, which will be the case in all but the simplest programs. The Main method finishes like so:

```
...
    while (done == false) { ; }
    Console.WriteLine("\nHit <enter> to close shell\n");
    Console.ReadLine();
  }
  catch (Exception ex)
  {
    Console.WriteLine(ex.Message);
    Console.ReadLine();
  }
} // Main
```

The curious-looking empty while loop allows the console application shell to stay alive. The loop will terminate when Boolean class-scope variable "done" is set to true by the speech recognizer event handler.

Handling Recognized Speech

The code for the speech-recognized event handler begins like this:

```
static void sre_SpeechRecognized(object sender,
  SpeechRecognizedEventArgs e)
{
  string txt = e.Result.Text;
  float confidence = e.Result.Confidence;
  Console.WriteLine("\nRecognized: " + txt);
  if (confidence < 0.60) return;
...
```

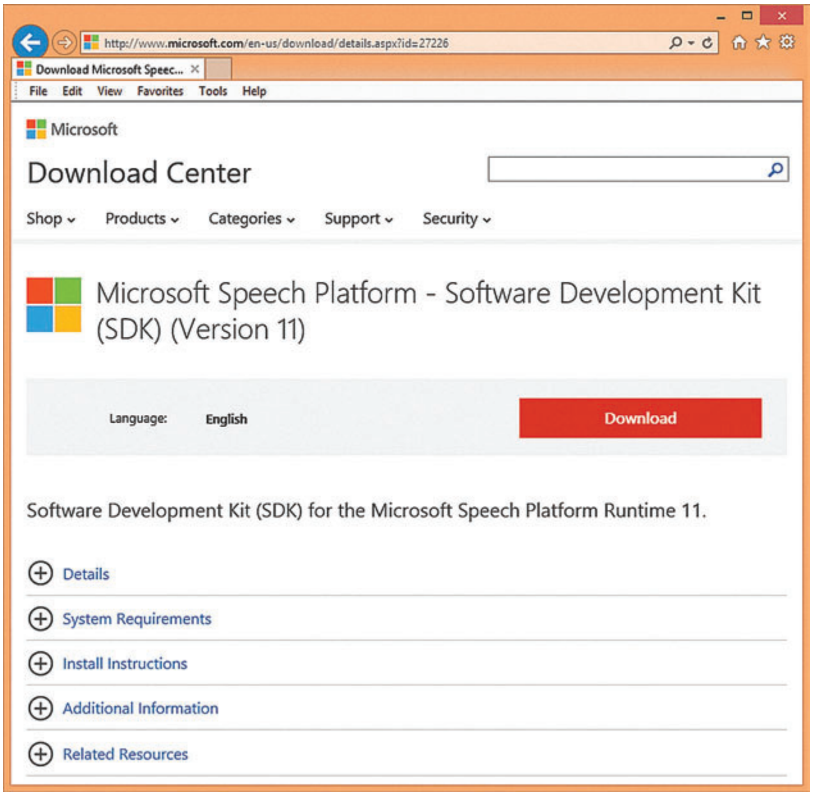
The actual text that's recognized is stored in the SpeechRecognizedEventArgs Result.Text property. You can also use the Result.Words collection. The Result.Confidence property holds a value between 0.0 and 1.0 that's a rough measure of how likely the spoken text matches any of the grammars associated with the recognizer. The demo instructs the event handler to ignore any low-confidence-recognized text.

Confidence values can vary wildly depending on the complexity of your grammars, the quality of your microphone and so on. For example, if the demo program must recognize only 1 through 4, the confidence values on my machine are typically about 0.75. However, if the grammar must recognize 1 through 100, the confidence values drop to about 0.25. In short, you must typically experiment with confidence values to get good speech-recognition results.

Next, the speech-recognizer event handler toggles recognition on and off:

```
if (txt.IndexOf("speech on") >= 0)
{
  Console.WriteLine("Speech is now ON");
  speechOn = true;
}
if (txt.IndexOf("speech off") >= 0)
{
  Console.WriteLine("Speech is now OFF");
  speechOn = false;
}
if (speechOn == false) return;
```

Although perhaps not entirely obvious at first, the logic should make sense if you examine it for a moment. Next, the secret exit command is processed:



http://www.microsoft.com/en-us/download/details.aspx?id=27226

Download Microsoft Speec...

Microsoft

Download Center

Microsoft Speech Platform - Software Development Kit (SDK) (Version 11)

Language: English

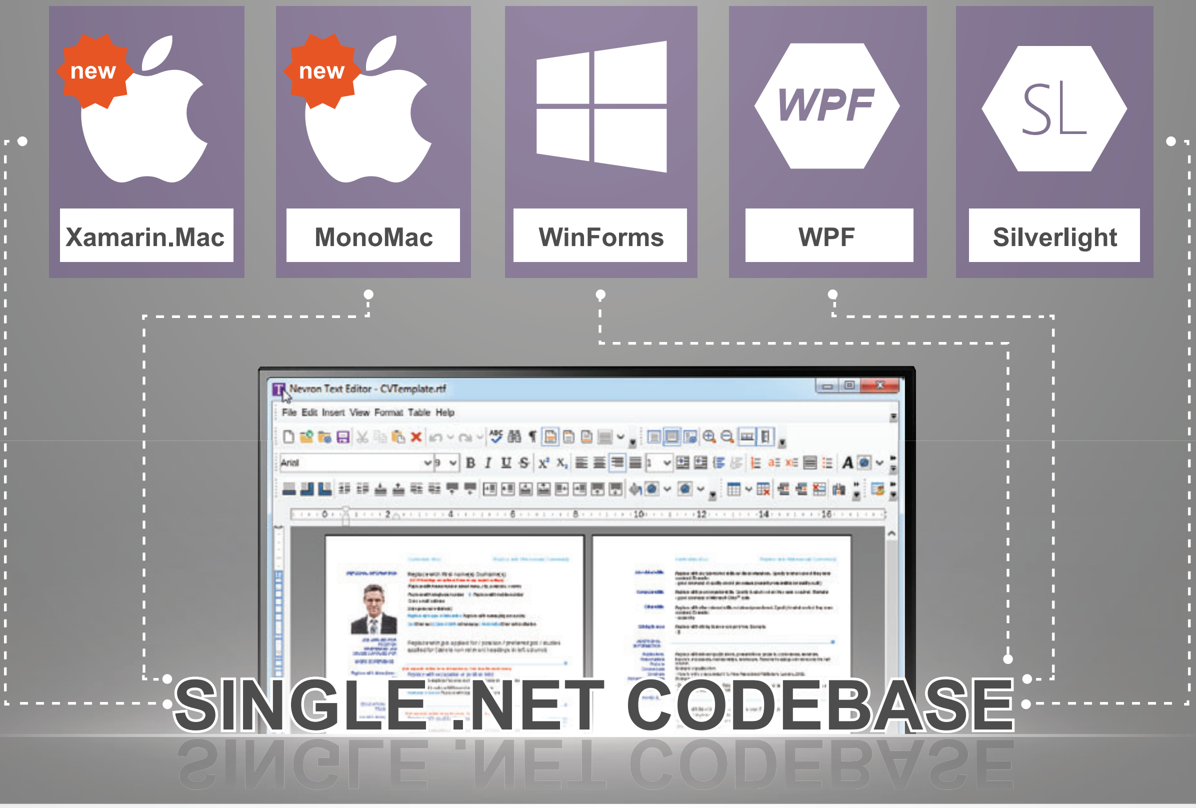
Software Development Kit (SDK) for the Microsoft Speech Platform Runtime 11.

Figure 4 **The SDK Installation Main Page at the Microsoft Download Center**

NEVRON
OPEN VISION for .NET

www.nevron.com

Write Once, Run Everywhere



Nevron Open Vision is the first suite of .NET controls for building cross-platform user interfaces and applications from a single code base.

Nevron Open Vision includes:

NOV **WIDGETS** for .NET

NOV **TEXT EDITOR** for .NET

NOV **GAUGE** for .NET

NOV **BARCODE** for .NET

COMING SOON:

NOV **CHART** for .NET

NOV **DIAGRAM** for .NET

NOV **GRID** for .NET

Learn more at **www.nevron.com** today

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

Visual Studio
Partner

All trademarks are property of their respective owners

```
if (txt.IndexOf("klatu") >= 0 && txt.IndexOf("barada") >= 0)
{
  ((SpeechRecognitionEngine)sender).RecognizeAsyncCancel();
  done = true;
  Console.WriteLine("(Speaking: Farewell)");
  ss.Speak("Farewell");
}
```

Notice that the speech recognition engine can in fact recognize nonsense words. If a Grammar object contains words that aren't in the object's built-in dictionary, the Grammar attempts to identify such words as best it can using semantic heuristics, and is usually quite successful. This is why I used "klatu" rather than the correct "klaatu" (from an old science fiction movie).

Also notice that you don't have to process the entire recognized Grammar text ("klatu barada nikto"), you only need to have enough information to uniquely identify a grammar phrase ("klatu" and "barada").

Next, commands to add two numbers are processed, and the event handler, Program class and namespace finish up:

```
...
      if (txt.IndexOf("What") >= 0 && txt.IndexOf("plus") >= 0)
      {
        string[] words = txt.Split(' ');
        int num1 = int.Parse(words[2]);
        int num2 = int.Parse(words[4]);
        int sum = num1 + num2;
        Console.WriteLine("(Speaking: " + words[2] +
          " plus " + words[4] + " equals " + sum + ")");
        ss.SpeakAsync(words[2] + " plus " + words[4] +
          " equals " + sum);
      }
    } // sre_SpeechRecognized
  } // Program
} // ns
```

Notice that the text in Results.Text is case-sensitive ("What" vs. "what"). Once you've recognized a phrase, you can parse out specific words. In this case, the recognized text has form, "What is x plus y," so the "What" is in words[0], and the two numbers to add (as strings) are in words[2] and words[4].

Installing the Libraries

The explanation of the demo program assumes you have all the necessary speech libraries installed on your machine. To create and run the demo programs, you need to install four packages: an SDK to be able to create the demos in Visual Studio, a runtime to be able to execute the demos after they've been created, a recognition language, and a synthesis (speaking) language.

To install the SDK, do an Internet search for "Speech Platform 11 SDK." This will bring you to the correct page in the Microsoft Download Center, as shown in **Figure 4**. After clicking the Download button, you'll see the options shown in **Figure 5**. The SDK comes in 32-bit and 64-bit versions. I strongly recommend using the 32-bit version regardless of what your host machine is. The 64-bit version doesn't interoperate with some applications.

You don't need anything except the single x86 (32-bit) .msi file. After selecting that file and clicking the Next button, you can run the installation program directly. The speech libraries don't give you much feedback about when the installation has completed, so don't look for some sort of success message.

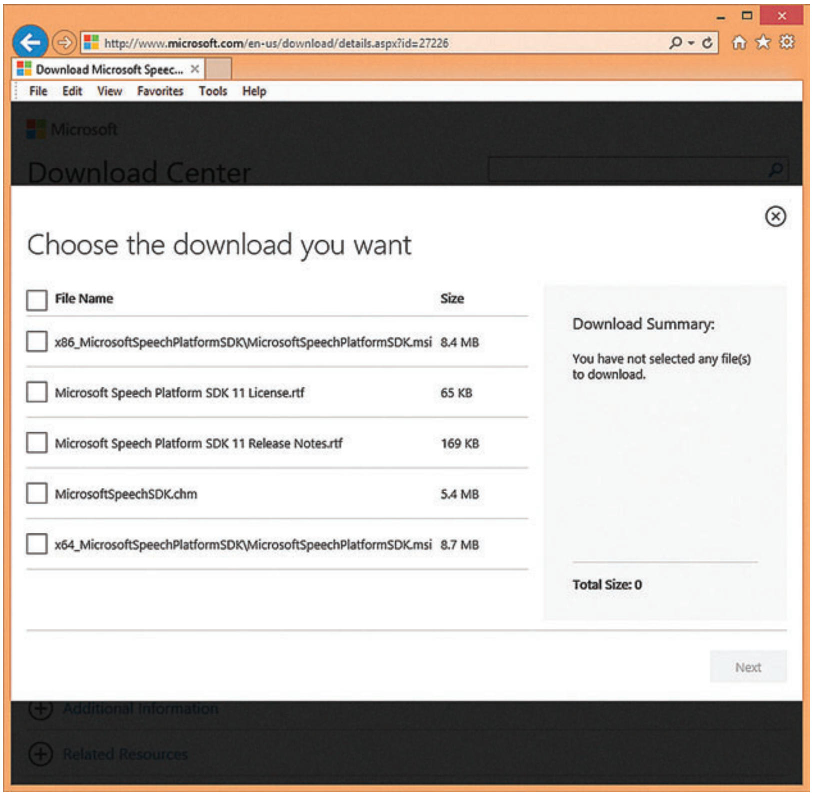
Next, you want to install the speech runtime. After finding the main page and clicking the Next button, you'll see the options shown in **Figure 6**.

It's critical you choose the same platform version (11 in the demo) and bit version (32 [x86] or 64 [x64]) as the SDK. Again, I strongly recommend the 32-bit version even if you're working on a 64-bit machine.

Next, you can install the recognition language. The download page is shown in **Figure 7**. The demo used file MSSpeech_SR_en-us_TELE.msi (English-U.S.). The SR stands for speech recognition and the TELE stands for telephony, which means that the recognition language is designed to work with low-quality audio input, such as that from a telephone or desktop microphone.

Finally, you can install the speech synthesis language and voice. The download page is shown in **Figure 8**. The demo uses file MSSpeech_TTS_en-us_Helen.msi. The TTS stands for text-to-speech, which is essentially a synonym phrase for speech synthesis. Notice there are two English, U.S. voices available. There are other English, non-U.S. voices, too. Creating synthesis files is quite difficult. It's possible to buy and then install other voices from a handful of companies.

Interestingly, even though a speech recognition language and a speech synthesis voice/language are really two entirely different things, both downloads are options from a single download page. The Download Center UI allows you to check both a recognition language and a synthesis language, but trying to install them at the same time was disastrous for me, so I recommend installing them one at a time.



Choose the download you want

☐ File Name	Size
☐ x86_MicrosoftSpeechPlatformSDK\MicrosoftSpeechPlatformSDK.msi	8.4 MB
☐ Microsoft Speech Platform SDK 11 License.rtf	65 KB
☐ Microsoft Speech Platform SDK 11 Release Notes.rtf	169 KB
☐ MicrosoftSpeechSDK.chm	5.4 MB
☐ x64_MicrosoftSpeechPlatformSDK\MicrosoftSpeechPlatformSDK.msi	8.7 MB

Download Summary:

You have not selected any file(s) to download.

Total Size: 0

Figure 5 **Installing the Speech SDK**

Microsoft.Speech vs. System.Speech

If you're new to speech recognition and synthesis for Windows applications, you can easily get confused by the documentation because there are multiple speech platforms. In particular, in addition to the Microsoft.Speech.dll library used by the demos in this article, there's a System.Speech.dll library that's part of the Windows OS. The two libraries are similar in the sense that the APIs are almost, but not quite, the same. So, if you're searching online for speech examples and you see a code snippet rather than a complete program, it's not always obvious if the example is referring to System.Speech or Microsoft.Speech.

The bottom line is, if you're a beginner with speech, for adding speech to a .NET application, use the Microsoft.Speech library, not the System.Speech library.

Although the two libraries share some of the same core base code and have similar APIs, they're definitely different. Some of the key differences are summarized in the table in **Figure 9**.

The System.Speech DLL is part of the OS, so it's installed on every Windows machine. The Microsoft.Speech DLL (and an associated runtime and languages) must be downloaded and installed onto a machine. System.Speech recognition usually requires user training, where the user reads some text and the system learns to understand that particular user's pronunciation. Microsoft.Speech recognition works immediately for any user. System.Speech can recognize virtually any words (called free dictation). Microsoft.Speech will recognize only words and phrases that are in a program-defined Grammar.

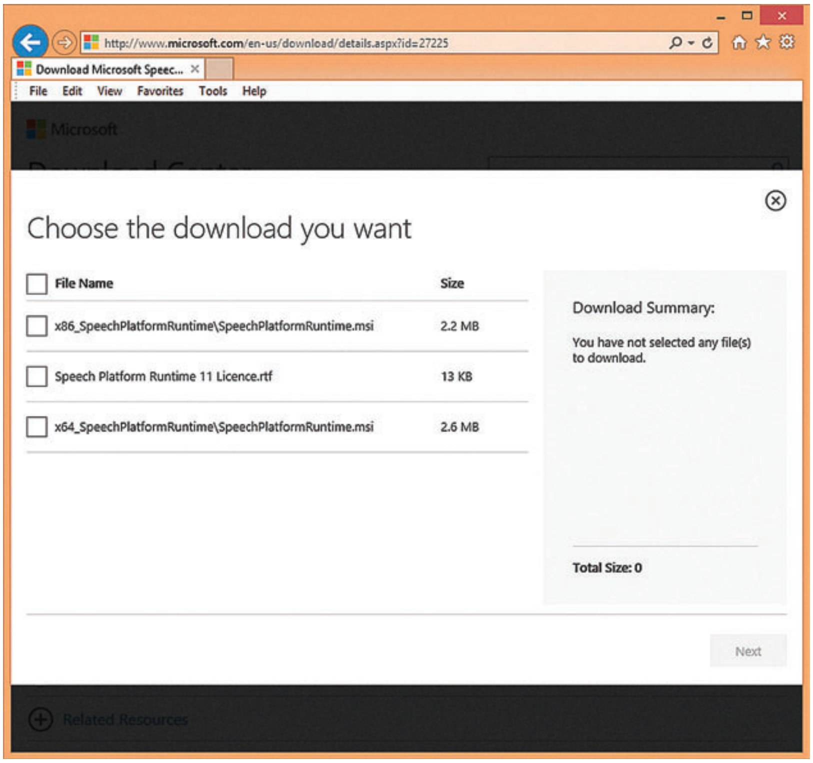
Adding Speech Recognition to a Windows Forms Application

The process of adding speech recognition and synthesis to a Windows Forms or WPF application is similar to that of adding speech to a console application. To create the dummy demo program shown in **Figure 2**, I launched Visual Studio and created a new C# Windows Forms application and named it WinFormSpeech.

After the template code loaded into the Visual Studio editor, in the Solution Explorer window, I added a Reference to file Microsoft.Speech.dll, just as I did with the console application demo. At the top of the source code, I deleted unnecessary using statements, leaving just references to the System, Data, Drawing and Forms namespaces. I added two using statements to bring the Microsoft.Speech.Recognition and System.Globalization namespaces into scope.

The Windows Forms demo doesn't use speech synthesis, so I don't use a reference to the Microsoft.Speech.Synthesis library. Adding speech synthesis to a Windows Forms app is exactly like adding synthesis to a console app.

In the Visual Studio design view, I dragged a TextBox control, a CheckBox control and a ListBox control onto the Form. I double-clicked on the CheckBox control and Visual Studio


http://www.microsoft.com/en-us/download/details.aspx?id=27225
Download Microsoft Speec...
File
Edit
View
Favorites
Tools
Help
Microsoft
Next
Related Resources

Choose the download you want

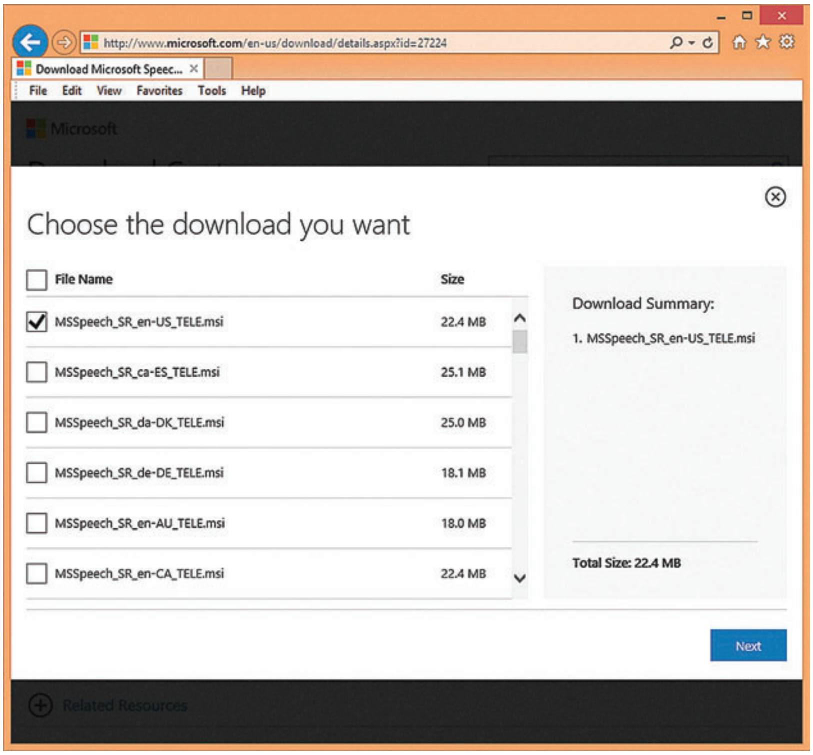
□ File Name	Size
□ x86_SpeechPlatformRuntime\SpeechPlatformRuntime.msi	2.2 MB
□ Speech Platform Runtime 11 Licence.rtf	13 KB
□ x64_SpeechPlatformRuntime\SpeechPlatformRuntime.msi	2.6 MB

Download Summary:

You have not selected any file(s) to download.

Total Size: 0

Figure 6 **Installing the Speech Runtime**


http://www.microsoft.com/en-us/download/details.aspx?id=27224
Download Microsoft Speec...
File
Edit
View
Favorites
Tools
Help
Microsoft
Next
Related Resources

Choose the download you want

□ File Name	Size
☑ MSSpeech_SR_en-US_TELE.msi	22.4 MB
□ MSSpeech_SR_ca-ES_TELE.msi	25.1 MB
□ MSSpeech_SR_da-DK_TELE.msi	25.0 MB
□ MSSpeech_SR_de-DE_TELE.msi	18.1 MB
□ MSSpeech_SR_en-AU_TELE.msi	18.0 MB
□ MSSpeech_SR_en-CA_TELE.msi	22.4 MB

Download Summary:

1. MSSpeech_SR_en-US_TELE.msi

Total Size: 22.4 MB

Figure 7 **Installing the Recognition Language**

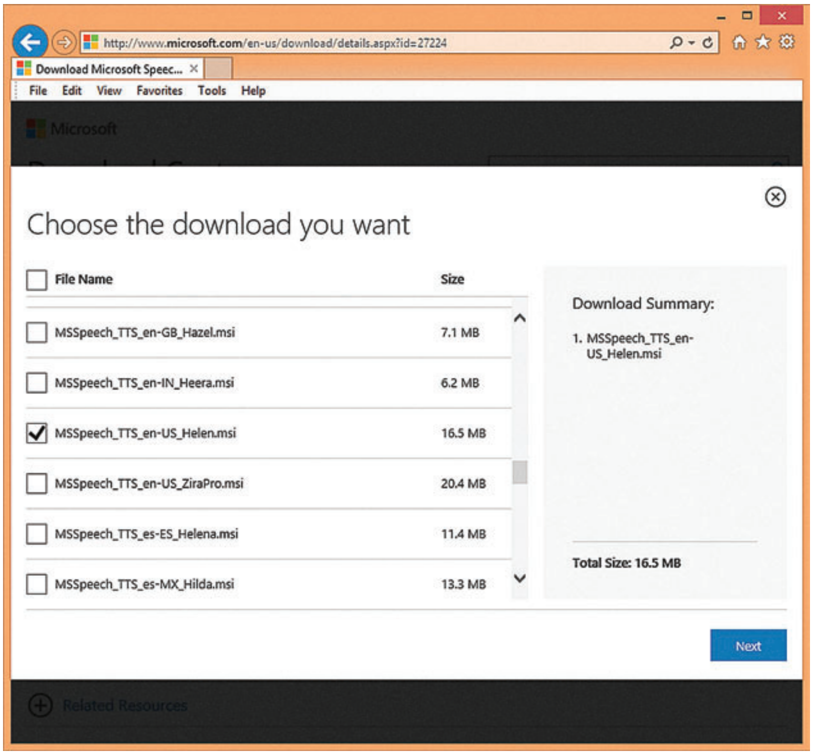
automatically created a skeleton of the CheckChanged event handler method.

Recall that the console app demo started listening for spoken commands immediately, and continuously listened until the app exited. That approach can be used for a Windows Forms app, but instead I decided to allow the user to toggle speech recognition on and off by using the CheckBox control.

The source code for the demo program's Form1.cs file, which defines a partial class, is presented in **Figure 10**. A speech recognition engine object is declared and instantiated as a Form member. Inside the Form constructor, I hook up the SpeechRecognized event handler and create and load two Grammars:

```
public Form1()
{
  InitializeComponent();
  sre.SetInputToDefaultAudioDevice();
  sre.SpeechRecognized += sre_SpeechRecognized;
  Grammar g_HelloGoodbye = GetHelloGoodbyeGrammar();
  Grammar g_SetTextBox = GetTextBox1TextGrammar();
  sre.LoadGrammarAsync(g_HelloGoodbye);
  sre.LoadGrammarAsync(g_SetTextBox);
  // sre.RecognizeAsync() is in CheckBox event
}
```

I could've created the two Grammar objects directly as I did in the console application demo, but instead, to keep things a bit cleaner, I defined two helper methods, GetHelloGoodbyeGrammar and GetTextBox1TextGrammar, to do that work.



http://www.microsoft.com/en-us/download/details.aspx?id=27224

Download Microsoft Speec...

Choose the download you want

	File Name	Size
☐	MSSpeech_TTS_en-GB_Hazel.msi	7.1 MB
☐	MSSpeech_TTS_en-IN_Heera.msi	6.2 MB
☑	MSSpeech_TTS_en-US_Helen.msi	16.5 MB
☐	MSSpeech_TTS_en-US_ZiraPro.msi	20.4 MB
☐	MSSpeech_TTS_es-ES_Helena.msi	11.4 MB
☐	MSSpeech_TTS_es-MX_Hilda.msi	13.3 MB

Download Summary:

1. MSSpeech_TTS_en-US_Helen.msi

Total Size: 16.5 MB

Figure 8 **Installing the Synthesis Language and Voice**

Notice that the Form constructor doesn't call the RecognizeAsync method, which means that speech recognition won't immediately be active when the application is launched.

Helper method GetHelloGoodbyeGrammar follows the same pattern as described earlier in this article:

```
static Grammar GetHelloGoodbyeGrammar()
{
  Choices ch_HelloGoodbye = new Choices();
  ch_HelloGoodbye.Add("hello"); // Should be an array!
  ch_HelloGoodbye.Add("goodbye");
  GrammarBuilder gb_result =
    new GrammarBuilder(ch_HelloGoodbye);
  Grammar g_result = new Grammar(gb_result);
  return g_result;
}
```

Similarly, the helper method that creates a Grammar object to set the text in the Windows Forms TextBox control doesn't present any surprises:

```
static Grammar GetTextBox1TextGrammar()
{
  Choices ch_Colors = new Choices();
  ch_Colors.Add(new string[] { "red", "white", "blue" });
  GrammarBuilder gb_result = new GrammarBuilder();
  gb_result.Append("set text box 1");
  gb_result.Append(ch_Colors);
  Grammar g_result = new Grammar(gb_result);
  return g_result;
}
```

The helper will recognize the phrase, "set text box 1 red." However, the user doesn't have to speak this phrase exactly. For example, a user could say, "Please set the text in text box 1 to red," and the speech recognition engine would still recognize the phrase as "set text box 1 red," although with a lower confidence value than if the user had matched the Grammar pattern exactly. Put another way, when you're creating Grammars, you don't have to take into account every variation of a phrase. This dramatically simplifies using speech recognition.

The CheckBox event handler is defined like so:

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
  if (checkBox1.Checked == true)
    sre.RecognizeAsync(RecognizeMode.Multiple);
  else if (checkBox1.Checked == false) // Turn off
    sre.RecognizeAsyncCancel();
}
```

The speech recognition engine object, sre, always remains in existence during the Windows Forms app's lifetime. The object is activated and deactivated using methods RecognizeAsync and RecognizeAsyncCancel when the user toggles the CheckBox control.

The definition of the speech-recognized event handler begins with:

```
void sre_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
  string txt = e.Result.Text;
  float conf = e.Result.Confidence;
  if (conf < 0.65) return;
...
```

In addition to the more or less always-used Result.Text and Result.Confidence properties, the Result object has several other useful, but more advanced, properties you might want to investigate, such as Homophones and ReplacementWordUnits.

Figure 9 **Microsoft.Speech vs System.Speech**

Microsoft.Speech.dll	System.Speech.dll
Must install separately	Part of the OS (Windows Vista+)
Can package with apps	Cannot redistribute
Must construct Grammars	Uses Grammars or free dictation
No user training	Training for specific user
Managed code API (C#)	Native code API (C++)

Figure 10 **Adding Speech Recognition to a Windows Forms**

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.Speech.Recognition;
using System.Globalization;
namespace WinFormSpeech
{
  public partial class Form1 : Form
  {
    static CultureInfo ci = new CultureInfo("en-us");
    static SpeechRecognitionEngine sre = new SpeechRecognitionEngine(ci);

    public Form1()
    {
      InitializeComponent();

      sre.SetInputToDefaultAudioDevice();
      sre.SpeechRecognized += sre_SpeechRecognized;
      Grammar g_HelloGoodbye = GetHelloGoodbyeGrammar();
      Grammar g_SetTextBox = GetTextBox1TextGrammar();
      sre.LoadGrammarAsync(g_HelloGoodbye);
      sre.LoadGrammarAsync(g_SetTextBox);
      // sre.RecognizeAsync() is in CheckBox event
    }

    static Grammar GetHelloGoodbyeGrammar()
    {
      Choices ch_HelloGoodbye = new Choices();
      ch_HelloGoodbye.Add("hello");
      ch_HelloGoodbye.Add("goodbye");
      GrammarBuilder gb_result = new GrammarBuilder(ch_HelloGoodbye);
      Grammar g_result = new Grammar(gb_result);
      return g_result;
    }

    static Grammar GetTextBox1TextGrammar()
    {
      Choices ch_Colors = new Choices();
      ch_Colors.Add(new string[] { "red", "white", "blue" });
      GrammarBuilder gb_result = new GrammarBuilder();
      gb_result.Append("set text box 1");
      gb_result.Append(ch_Colors);
      Grammar g_result = new Grammar(gb_result);
      return g_result;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
      if (checkBox1.Checked == true)
        sre.RecognizeAsync(RecognizeMode.Multiple);
      else if (checkBox1.Checked == false) // Turn off
        sre.RecognizeAsyncCancel();
    }

    void sre_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
    {
      string txt = e.Result.Text;
      float conf = e.Result.Confidence;

      if (conf < 0.65) return;

      this.Invoke(new MethodInvoker(() =>
      { listBox1.Items.Add("I heard you say: " + txt); })); // WinForm specific

      if (txt.IndexOf("text") >= 0 && txt.IndexOf("box") >=
        0 && txt.IndexOf("1")>= 0)
      {
        string[] words = txt.Split(' ');
        this.Invoke(new MethodInvoker(() =>
        { textBox1.Text = words[4]; })); // WinForm specific
      }
    }
  } // Form
} // ns
```

Additionally, the speech recognition engine has several useful events such SpeechHypothesized.

The event handler code concludes with:

```
...
  this.Invoke(new MethodInvoker(() =>
    { listBox1.Items.Add("I heard you say: " + txt); }));

  if (txt.IndexOf("text") >= 0 &&
    txt.IndexOf("box") >= 0 && txt.IndexOf("1")>= 0)
  {
    string[] words = txt.Split(' ');
    this.Invoke(new MethodInvoker(() =>
    { textBox1.Text = words[4]; }));
  }
}
```

The recognized text is echoed in the ListBox control using the MethodInvoker delegate. Because the speech recognizer is running in a different thread from the Windows Forms UI thread, a direct attempt to access the ListBox control, such as:

```
listBox1.Items.Add("I heard you say: " + txt);
```

will fail and throw an exception. An alternative to MethodInvoker is to use the Action delegate like this:

```
this.Invoke( (Action)( () =>
  listBox1.Items.Add("I heard you say: " + txt)));
```

In theory, in this situation, using the MethodInvoker delegate is slightly more efficient than using the Action delegate because MethodInvoker is part of the Windows.Forms namespace and, therefore, specific to Windows Forms applications. The Action delegate is more general. This example shows you can completely manipulate a Windows Forms application using speech recognition—incredibly powerful and useful.

Wrapping Up

The information presented in this article should get you up and running if you want to explore speech recognition and speech synthesis with .NET applications. Mastering the technology itself isn't too difficult once you get over the initial installation and learning hurdles. The real issue with speech recognition and synthesis is determining when they're useful.

With console applications, you can create interesting back-and-forth dialogs where the user asks a question and the application answers, resulting in a Cortana-like environment. You have to be a bit careful because when your computer speaks, that speech will be picked up by the microphone, and may be recognized. I've found myself in some amusing situations where I ask a question, the application recognizes and answers, but the spoken answer triggers another recognition event, and I end up in an entertaining infinite speech loop.

Another possible use of speech with a console application is to recognize commands such as, "Launch Notepad" and "Launch Word." In other words, a console application can be used to perform actions on your host machine that would normally be performed using multiple mouse and keyboard interactions. ■

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

Thanks to the following Microsoft Research technical experts for reviewing this article: *Rob Gruen, Mark Marron and Curtis von Veh*

NODE.JS

Building Web Apps on the MEAN Stack with OData in Microsoft Azure

Long Le

Microsoft .NET developers typically build great apps using JavaScript on the client side and ASP.NET (C# or Visual Basic .NET) on the server side. But what if you could use one common language to build apps on all layers of the stack, everything from the browser and the services layer to server-side business processing, and even to querying and programming in the database? Now you can, with Node.js. Node.js has been around for a number of years, but its adoption has picked up significantly in recent years. Node.js stacks, such as the MongoDB, Express, AngularJS, Node.js (MEAN) stack, bring many benefits to building apps, including the fact that there's very little disconnect (if any), between front-end, middle-layer and back-end developers. In many cases the same programmer can develop all the layers of an app, because it's all done in JavaScript. Moreover, you can now build Node.js apps directly within Visual Studio 2013 with Node.js Tools for Visual Studio (NTVS), including full debugging capability.

Note that Node Tools for Visual Studio and MongoLab on Microsoft Azure are currently in preview (beta) mode. Information is subject to change.

This article discusses:

- Building Web apps on the MEAN stack
- Configuring OData with JayData in Node.js
- Charts and graphs with Kendo UI
- Continuous integration and deployment to Azure with Git

Technologies discussed:

Microsoft Azure, Node.js Tools for Visual Studio, MongoDB, Express, AngularJS, Node.js

Code download available at:

msdnmeanstack.codeplex.com

Getting Started

In this article I'm going to show you that by using the MEAN stack, building create, read, update and delete (CRUD)-heavy applications can be fast and easy. I'm going to assume you have a basic conceptual understanding of AngularJS (angularjs.org), Node.js (nodejs.org), MongoDB (mongodb.org) and Express (expressjs.com). If you're planning to follow along, please be sure you have the following installed:

- Visual Studio 2013 Update 3 (bit.ly/1o2WTc)
- Node.js Tools for Visual Studio (nodejstools.codeplex.com)
- MongoDB (Download, bit.ly/1rw0BZm; Install, bit.ly/1uJN8e0)

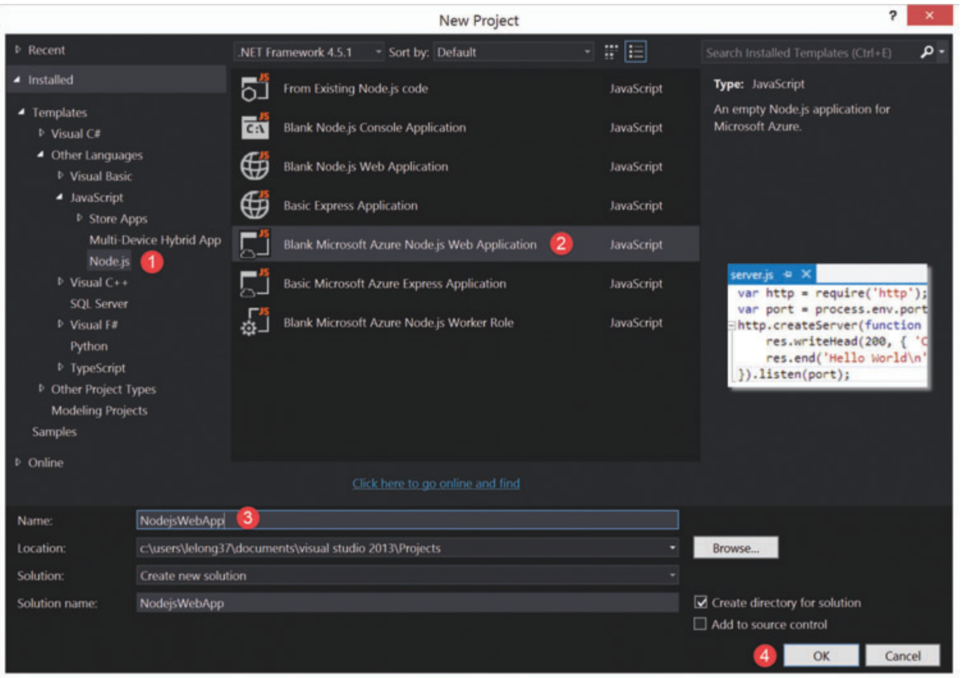
The first step is to open the New Project dialog in Visual Studio and choose the Blank Microsoft Azure Node.js Web Application template, as shown in **Figure 1**. You could shortcut a few items by choosing the Basic Microsoft Azure Express Application template, but a blank template provides more granular control over what to install as middleware for the Node.js application.

What is Node.js middleware? To oversimplify, it's simply modules you can plug in to your Node.js application's Express HTTP request pipeline. Typically, the middleware gets executed for every HTTP request.

Next, install Express using the Node Package Manager (NPM). If you're familiar with NuGet packages, NPM packages are basically the same thing, but for Node.js applications.

As you can see in **Figure 2**, I added @3 in the Other npm arguments text field, in order to install the latest version of Express 3. Although Express 4 has been released, you need to stick with Express 3 because the other modules that will be installed haven't been updated to some of the breaking changes of Express 4.

You'll need to download and install the rest of the required NPM packages: express, odata-server, stringify-object and body-parser, but there's no need to have any "Other npm arguments," as I'll be using the latest version of each of these npm packages.


New Project
Recent
Installed
Templates
Visual C#
Other Languages
Visual Basic
JavaScript
Store Apps
Multi-Device Hybrid App
Node.js
1
Visual C++
SQL Server
Visual F#
Python
TypeScript
Other Project Types
Modeling Projects
Samples
Online
.NET Framework 4.5.1
Sort by: Default
Search Installed Templates (Ctrl+E)
From Existing Node.js code
Blank Node.js Console Application
Blank Node.js Web Application
Basic Express Application
Blank Microsoft Azure Node.js Web Application
2
Basic Microsoft Azure Express Application
Blank Microsoft Azure Node.js Worker Role
JavaScript
Click here to go online and find
Browse...
3
4
OK
Cancel

Type: JavaScript

An empty Node.js application for Microsoft Azure.

server.js

```
var http = require('http');
var port = process.env.port
http.createServer(function
    res.writeHead(200, { 'C
    res.end('Hello World\n'
}).listen(port);
```

Name: NodejsWebApp

Location: c:\users\lelong37\documents\visual studio 2013\Projects

Solution: Create new solution

Solution name: NodejsWebApp

☑ Create directory for solution

□ Add to source control

Figure 1 **Create a Blank Microsoft Azure Node.js Web Application**

Setting up the Server.js File

The server.js (sometimes named app.js) file, shown in **Figure 3**, is basically the starting point of the Node.js app. This is where you configure your application and inject any needed middleware modules.

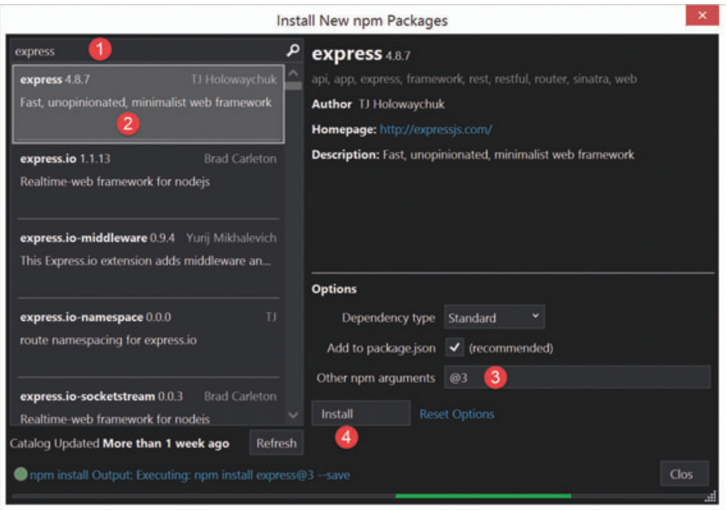
In order to consume the required NPM packages/libraries that you download, you need to use the keyword require("package name") to bring these libraries in scope for a given Node.js class, as shown in lines 1 to 6 in **Figure 3**. I'll quickly review the contents of server.js:

- Line 1-6: Bring all the required packages into the server.js scope so they can be initialized and plugged into the HTTP request pipeline.
- Line 7: Initialize a new Express Web application.
- Line 8: Define the OData configuration for the REST endpoints; more on this in a bit.
- Line 10: Plug in express.static and pass the directory path to expose the directory path that's passed in publicly. This lets anyone reach any content placed in the NodejsWebApp/Public directory. For example, http://localhost:1337/image/myImage.gif would render the image in NodejsWebApp/Public/image/myimage.gif to the browser.
- Line 12: Set up a default landing page using the app.get method. The first parameter takes in the path (the root path of the app). Here, I'm simply rendering a static HTML file by providing the path to it.
- Line 15: Instruct the application to listen and serve HTTP requests on the specified port; for development purposes I'm using port 1337, so my application will listen for requests at http://localhost:1337.
- Line 16: Print the environment variables to the Node.js console window to bring some visibility to the Node.js environment.

Configuring OData

With server.js set up, I'm going to focus now on Line 8, where I configure the OData REST endpoints. First, you'll need to create two modules: NodejsWebApp/server/data/northwind.js (**Figure 4**) and NodejsWebApp/server/data/odata.js (**Figure 5**).

Note that MongoDB is a NoSQL database—that is, a non-relational document database. When migrating a traditional Northwind database to MongoDB to take advantage of the NoSQL model, there can be many ways to structure it. For this article, I'll leave the Northwind schema, for the most part, intact. (I've removed other entity model definitions, registrations, and inserts from **Figure 4** for brevity.)



express 4.8.7 TJ Holowaychuk
Fast, unopinionated, minimalist web framework

express.io 1.1.13 Brad Carleton
Realtime-web framework for nodejs

express.io-middleware 0.9.4 Yurij Mikhalevich
This Express.io extension adds middleware an...

express.io-namespace 0.0.0 TJ
route namespacing for express.io

express.io-socketstream 0.0.3 Brad Carleton
Realtime-web framework for nodejs

Catalog Updated **More than 1 week ago**

express 4.8.7
api, app, express, framework, rest, restful, router, sinatra, web

Author TJ Holowaychuk

Homepage: http://expressjs.com/

Description: Fast, unopinionated, minimalist web framework

Dependency type: Standard

Add to package.json ✓ (recommended)

Other npm arguments: @3

```
npm install Output: Executing: npm install express@3 --save
```

Figure 2 **Finding and Installing NPM Packages Such as Express**

Figure 3 **The Server.js File**

```
var http = require('http');
var express = require( 'express' );
var odata = require( './server/data/odata' );
var stringify = require( 'stringify-object' );
var config = require("./server/config/config");
var bodyParser = require("body-parser");

var app = express( );
odata.config( app );

app.use(bodyParser.json());
app.use( express.static( __dirname + "/public" ) );

var port = process.env.port || 1337;

app.get("/", function(req, res) {
res.sendfile("/public/app/views/index.html", { root: __dirname });
});

http.createServer(app).listen(port);
console.log(stringify( process.env ));
```

Figure 4 **NodejsWebApp/server/data/northwind.js**

```
$data.Entity.extend( 'Northwind.Category', {
  CategoryID: { key: true, type: 'id', nullable: false, computed: true },
  CategoryName: { type: 'string', nullable: false, required: true, maxLength: 15 },
  Description: { type: 'string', maxLength: Number.POSITIVE_INFINITY },
  Picture: { type: 'blob', maxLength: Number.POSITIVE_INFINITY },
  Products: { type: 'Array', elementType: 'Northwind.Product',
inverseProperty: 'Category' }
} );

$data.Entity.extend( 'Northwind.Product', {
  ProductID: { key: true, type: 'id', nullable: false, computed: true },
  ProductName: { type: 'string', nullable: false, required: true, maxLength: 40 },
  EnglishName: { type: 'string', maxLength: 40 },
  QuantityPerUnit: { type: 'string', maxLength: 20 },
  UnitPrice: { type: 'decimal' },
  UnitsInStock: { type: 'int' },
  UnitsOnOrder: { type: 'int' },
  ReorderLevel: { type: 'int' },
  Discontinued: { type: 'bool', nullable: false, required: true },
  Category: { type: 'Northwind.Category', inverseProperty: 'Products' },
  Order_Details: { type: 'Array', elementType: 'Northwind.Order_Detail',
inverseProperty: 'Product' },
  Supplier: { type: 'Northwind.Supplier', inverseProperty: 'Products' }
} );

$data.Class.define( "NorthwindContext", $data.EntityContext, null, {
  Categories: { type: $data.EntitySet, elementType: Northwind.Category },
  Products: { type: $data.EntitySet, elementType: Northwind.Product },
  // Other entity registrations removed for brevity, please see actual source code.
} );

// Other entity definitions removed for brevity, please see actual source code.

NorthwindContext.generateTestData = function( context, callBack ) {

  var category1 = new Northwind.Category( { CategoryName: 'Beverages',
    Description: 'Soft drinks, coffees, teas, beer, and ale' } );
  // Other category instances removed for brevity, please see actual source code.

  context.Categories.add( category1 );
  // Other category inserts removed for brevity, please see actual source code.

  context.Products.add( new Northwind.Product( { ProductName: 'Ipoh
Coffee', EnglishName: 'Malaysian Coffee',
    UnitPrice: 46, UnitsInStock: 670, Discontinued: false, Category: category1 } ) );
  // Other product inserts removed for brevity, please see actual source code.

  context.saveChanges( function ( count ) {
    if ( callBack ) {
      callBack( count );
    }
  } );
};

module.exports = exports = NorthwindContext;
```

In **Figure 4** the models and entities are simply being defined, which can be reused later on the client side when performing CRUD operations, such as creating new Products, for example. Also, the NorthwindContext.generateTestData method will seed the database every time the application is restarted, which will come in handy when you deploy the application to a live demo site. This makes it easy to refresh the data whenever needed simply by recycling the application. A more elegant approach would be to wrap this code into an Azure WebJob and schedule it to refresh at a set frequency, but leave it as is for now. The last line in this module, module.exports = exports = NorthwindContext, wraps everything so that later on you can "require" this module and use the "new" operator to create a new instance of the Northwind object type, which is done in the NodejsWebApp/server/data/odata.js module, shown in **Figure 5**.

You can query MongoDB via the command line or by using one of the many MongoDB GUI tools out there (such as RoboMongo) to confirm that the seed data was indeed inserted. Because the focus of this article is on OData, use LINQPad because it includes a built-in provider to query with LINQ against OData version 3.

To test the endpoints, download and install LINQPad (linqpad.net), and then run your application (F5 in Visual Studio 2013). Then fire up LINQPad and set up a new connection to the OData endpoint. To do so, click Add connection and select OData as your LINQPad data provider. Then configure the OData LINQ connection with the URI http://localhost:1337/northwind.svc; username, Admin; and password, Admin. LINQPad will render the hierarchy based on the OData CSDL endpoint, as you can see in the upper-left corner of **Figure 6**.

Figure 5 **NodejsWebApp/server/data/odata.js Module**

```
( function (odata) {

  var stringify = require( 'stringify-object' );
  var config = require( "../config/config" );
  console.log( stringify( config ) );

  odata.config = function ( app ) {
    var express = require( 'express' );
    require( 'odata-server' );

    var northwindContextType = require( './northwind.js' );

    var northwindContext = new NorthwindContext( {
      address: config.mongoDb.address,
      port: config.mongoDb.port,
      username: config.mongoDb.username,
      password: config.mongoDb.password,
      name: config.mongoDb.name,
      databaseName: config.mongoDb.databaseName,
      dbCreation: $data.storageProviders.DbCreationType.DropAllExistingTables
    } );

    console.log( "northwindContext :" );
    stringify( northwindContext );

    northwindContext.onReady( function ( db ) {
      northwindContextType.generateTestData( db, function ( count ) {
        console.log( 'Test data upload successful. ', count, 'items inserted.' );
        console.log( 'Starting Northwind OData server.' );

        app.use( express.basicAuth( function ( username, password ) {
          if ( username == 'admin' ) {
            return password == 'admin';
          } else return true;
        } ) );
```

Spreadsheets Made Easy.

Fastest Calculations

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

WIN Windows Forms | Silverlight | WPF WPF

Powerful Controls

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

Comprehensive Charting

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

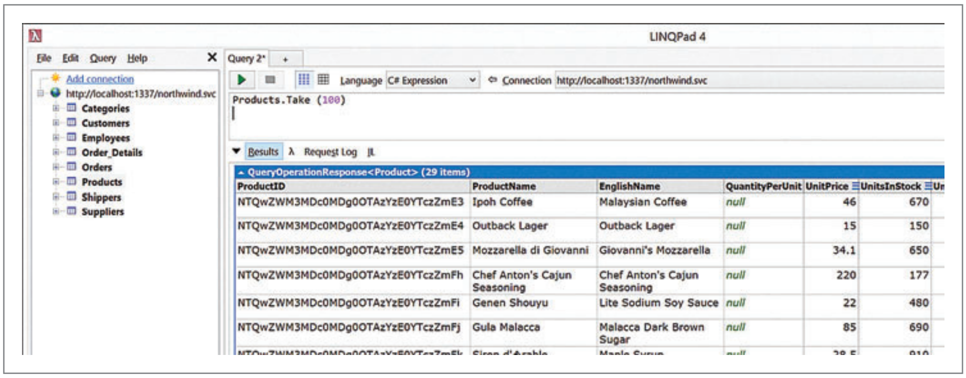
Scalable Reporting

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

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com



```
Products.Take (100)
```

▲ QueryOperationResponse<Product> (29 items)

ProductID	ProductName	EnglishName	QuantityPerUnit	UnitPrice	UnitsInStock
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmE3	Ipoh Coffee	Malaysian Coffee	null	46	670
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmE4	Outback Lager	Outback Lager	null	15	150
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmE5	Mozzarella di Giovanni	Giovanni's Mozzarella	null	34.1	650
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmFh	Chef Anton's Cajun Seasoning	Chef Anton's Cajun Seasoning	null	220	177
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmFi	Genen Shouyu	Lite Sodium Soy Sauce	null	22	480
NTQwZWM3MDc0MDg0OTAzYzE0YTczZmFj	Gula Malacca	Malacca Dark Brown Sugar	null	85	690

Figure 6 **A LINQ Query and Its Results Using the Discovered Data Model**

There should be data for Products based on the seed data used on the server-side (NodejsWebApp/server/northwind.js), so you'll want to do a quick LINQ query on Products using LINQPad:

```
Products.Take(100)
```

Figure 6 also shows the query and its results.

As you can see, the OData server is properly set up and you're able to issue LINQ queries over HTTP and get a list of products back. If you click on the Request Log tab, you can actually see the HTTP GET OData URL LINQPad generates from the LINQ statement: http://localhost:1337/northwind.svc/Products()?$top=100.

Once you've confirmed your OData server is indeed running on the Node.js Express Web app, you'll want to make use of this and start building out some common use cases that can consume that OData goodness. Place everything on the client side in the "public" folder and all code that runs on the server side in a folder named Server. Create all the files needed for your app in advance as stubs or placeholders, and then come back around and fill in the blanks. **Figure 7** shows the structure of the NodejsWebApp project.

The app.js file (NodejsWebApp/public/app/app.js) shown in **Figure 8** is basically the starting point of the (client-side) AngularJS application. I won't go into all the details; the takeaway here is that you want to register your client-side routes for your single-page application (SPA) with the $routeProvider. For each of the routes (defined with the .when method), provide a path to the view (HTML) to render by setting the templateUrl property, and specify a view's controller by setting the controller property for a given route. The AngularJS controller is where all of the code lives to facilitate whatever the view requires—in short, all the JavaScript code for the view. The .otherwise method is used to configure a default route (the home view) for any incoming requests that don't match any of the routes.

Here's a quick recap of how the concerns of the Model-View-ViewModel (MVVM) pattern are represented in the app:

- View = *.html
- ViewModel = *controller.js
- Model = entities that are returned from REST endpoints, usually are domain models and/or entities

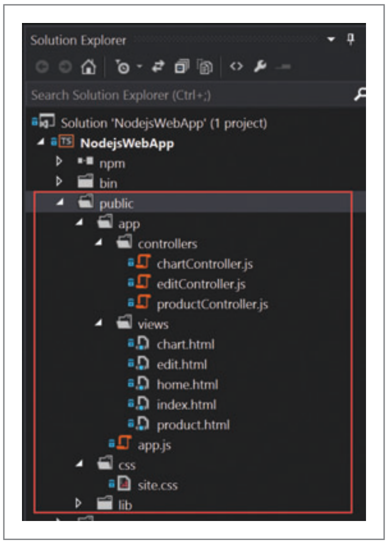


Figure 7 **The NodejsWebApp Project**

Figure 9 shows which files in the application address which concern in the MVVM pattern.

Defining the JayData Client-Side DataContext as an AngularJS Service

Because most of the controllers will use the Northwind context, you'll want to create a service/factory named northwindFactory. And because the initialization of the Northwind context is async, you'll want to set up a JavaScript promise to ensure the Northwind context initialization has completed and is ready to be used by the time any of the controllers are loaded. So, in short, the Northwind context will finish loading before any controller with a dependency on the northwindFactory loads. Notice that all of the configured routes have a "resolve" property, which is how you define what promises need to resolve before the controller is loaded. In this case, the property, "northwind," is set to the northwindFactory. The property name "northwind" will also be the name of the instance that will be injected into the controller. You'll see the constructor function for productController.js in a bit (in **Figure 11**), where the northwindFactory is injected as northwind, the property name that's set for northwindFactory in the resolve property in the routes.

Because most of the controllers will use the Northwind context, you'll want to create a service/factory named northwindFactory.

Index.html, shown in **Figure 10**, will basically be the layout page and AngularJS will know which views to swap into the div with the attribute ng-view. Note that you have to specify the AngularJS app by configuring any HTML element that's a parent element of the div attributed with "ng-view." In this case you want to set "ng-app" to "myApp," which is what the application is named in the app.js.

Note that I'm using a content delivery network (CDN) for all of my client-side JavaScript library includes. You can download the client-side libraries locally using Bower at the command line (as you'd typically do for .NET projects with NuGet using the Package Manager console). In the Microsoft .NET

GdPicture.NET 10

NEW RELEASE
GdPicture.NET 10
Royalty-Free Document Imaging Toolkits for .NET and COM/ActiveX

- Document viewing, processing, printing and scanning (TWAIN & WIA).
- Reading, writing and converting vector and raster images in more than 90 formats, PDF included.
- OMR, OCR, barcode reading and writing (linear & 2D).
- Annotations for image and PDF within Windows and Web applications.
- Color detection engine for image and PDF compression.

And much more ...

All-In-One Document Imaging SDK

Royalty-Free Document Imaging Toolkits
for .NET and COM/ActiveX

GdPicture.NET 10 Plugins

- NEW Color detection
- NEW DICOM image reader
- NEW MICR reader
- Full managed PDF support
- Full annotations support for PDF and images
- OCR
- Forms processing
- JBIG2 encoding
- 1D and 2D barcode reading and writing

Operators in Type 4 Functions
你好
APPROVED By John Smith
1 / 34
NF Z 42-013
Décembre 2001
norme française
Archivage électronique
Spécifications relatives à la conception et à l'exploitation de systèmes informatiques en vue d'assurer la conservation et l'intégrité des documents stockés dans ces systèmes
Norme française homologuée
Correspondance

Try GdPicture.NET 10 FREE for 30 days

www.gdpicture.com

Figure 8 **The App.js File**

```
'use strict';

var myApp = angular.module('myApp',
  [
    'ngRoute',
    'ngAnimate',
    'kendo.directives',
    'jaydata'
  ])
  .factory("northwindFactory",
  [
    '$data',
    '$q',
    function($data, $q) {
      // Here you wrap a jquery promise into an angular promise.
      // Simply returning jquery promise causes bogus things
      var defer = $q.defer();
      $data.initService("/northwind.svc").then(function(ctx) {
        defer.resolve(ctx);
      });
      return defer.promise;
    }
  ])
  .config(function($routeProvider) {
    $routeProvider
      .when('/home',
      {
        templateUrl: 'app/views/home.html'
      })
      .when('/product',
      {
        templateUrl: 'app/views/product.html',
        controller: 'productController',
        resolve: {
          northwind: 'northwindFactory'
        }
      })
      .when('/edit/:id',
      {
        templateUrl: 'app/views/edit.html',
        controller: 'editController',
        resolve: {
          northwind: 'northwindFactory'
        }
      })
      .when('/chart',
      {
        templateUrl: 'app/views/chart.html',
        controller: 'chartController',
        resolve: {
          northwind: 'northwindFactory'
        }
      })
      .otherwise(
      {
        redirectTo: '/home'
      });
  });
```

Framework you use NuGet for both client-side and server-side packages. In the Node.js realm, however, Bower is used to download client-side libraries/packages while NPM is used to download and install server-side libraries/packages.

For the layout UI, I use a vanilla bootstrap theme, the one that the Visual Studio ASP.NET MVC 5 project template generates.

Product View

Just a few lines of HTML are needed for the product view (NodejsWebApp/public/app/views/products.html). The first block is the Kendo directive for AngularJS to render the grid:

```
<!-- Kendo UI's AngularJS directive for the grid -->
<div kendo-grid="grid" k-options="options"></div>

<!-- AngularJS template for our View Detail Button in the Grid Toolbar
-->
<script type="text/x-kendo-template" id="viewDetail">
  <a
    class="k-button "
    ng-click="viewDetail(this)">View Detail</a>
</script>
```

The second block is just an AngularJS template for the custom View Detail button you add to the grid's toolbar.

Figure 11 shows the Product Controller, NodejsWebApp/app/controllers/productController.js.

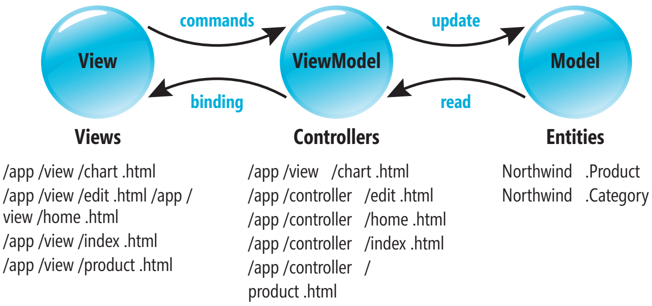
To hydrate the Products Grid, you need to instantiate a Kendo UI DataSource ($scope.options.dataSource). JayData provides a helper method to initialize a Kendo UI DataSource bound to its OData REST endpoints. The JayData asKendoDataSourcehelper method knows how to create the DataSource based on the metadata information published by the OData server (http://localhost:1337/northwindsvc), which is then used to configure the $data instance in northwindFactory in app.js. You'll see more of the Kendo DataSource when I demonstrate visual impressions with the Kendo DataViz charting framework.

Along with the out-of-the-box buttons (create, save and cancel) configured in the grid's toolbar, you add a custom button to navigate to another view that will render the complete details of a selected product row ($scope.viewDetail). When the View Detail button click event occurs, get the selected product DataItem and then, using the AngularJS $location service, navigate to the edit view (MyNodejsWebApp/scripts/app/views/edit.html) for the product.

Figure 12 shows the Edit.html file, NodejsWebApp/public/app/views/edit.html.

Notice how the inputs are decorated with the ng-model attribute, which is the AngularJS way of declaratively indicating that the value for that input will be stored in a property the ng-model value is set to on the controller $scope. For example, in the first input field in this view, whose HTML element id is set to productName (id="productName"), ng-model is set to product.ProductName. This means that whatever the user enters in the input field (textbox), the value for $scope.productName will be set accordingly. Moreover, whatever $scope.product.productName is set to programmatically in editController will be automatically reflected in the value of the input field for productName.

As an example, when the view first loads, you load the product by the ID passed in through URL, then set $scope.product to that product (see **Figure 13**). Once this happens, everything in the view with



/app /view /chart .html
/app /view /edit .html /app / view /home .html
/app /view /index .html
/app /view /product .html

/app /view /chart .html
/app /controller /edit .html
/app /controller /home .html
/app /controller /index .html
/app /controller / product .html

Northwind .Product
Northwind .Category

Figure 9 **The Model-View-ViewModel Pattern**

Figure 10 **The Index.html File**

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8" />
    <title>NodejsWebApp</title>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css"
      rel="stylesheet">
    <link href="//cdn.kendostatic.com/2014.2.716/styles/kendo.common.min.css"
      rel="stylesheet" />
    <link href="//cdn.kendostatic.com/2014.2.716/styles/kendo.bootstrap.min.css"
      rel="stylesheet" />
    <link href="//cdn.kendostatic.com/2014.2.716/styles/kendo.dataviz.min.css"
      rel="stylesheet" />
    <link href="//cdn.kendostatic.com/2014.2.716/styles/
      kendo.dataviz.bootstrap.min.css" rel="stylesheet" />
    <link href="../../css/site.css" rel="stylesheet" />
  </head>
    <body>
    <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle"
            data-toggle="collapse" data-target=".navbar-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="/">NodejsWebApp</a>
        </div>
        <div class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li>
              <a href="#/home">Home</a>
            </li>
            <li>
              <a href="#/about">About</a>
            </li>
            <li>
              <a href="#/contact">Contact</a>
            </li>
            <li>
              <a href="#/product">Product</a>
            </li>
            <li>
              <a href="#/chart">Chart</a>
            </li>
          </ul>
        </div>
       </div>
    </div>
    <!-- Binding the application to our AngularJS app: "myApp" -->
    <div class="container body-content" ng-app="myApp">
      <br />
      <br/>
      <!-- AngularJS will swap our Views inside this div -->
      <div ng-view></div>
      <hr />
      <footer>
        <p>&copy; 2014 - My Node.js Application</p>
      </footer>
    </div>
    <script src="//code.jquery.com/jquery-2.1.1.min.js"></script>
    <script src=
      "//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js">
    </script>
    <script src="//code.angularjs.org/1.3.0-beta.16/angular.min.js"></script>
    <script src="//code.angularjs.org/1.3.0-beta.16/angular-route.min.js"></script>
    <script src="//code.angularjs.org/1.3.0-beta.16/angular-animate.min.js"></script>
    <script src="//cdn.kendostatic.com/2014.2.716/js/kendo.all.min.js"></script>
    <script src="//include.jaydata.org/datajs-1.0.3-patched.js"></script>
    <script src="//include.jaydata.org/jaydata.js"></script>
    <script src="//include.jaydata.org/jaydatamodules/angular.js"></script>
    <script src="/lib/jaydata-kendo.js"></script>
    <!--<script src="//include.jaydata.org/jaydatamodules/kendo.js"></script>-->
    <script src="/app/app.js"></script>
    <script src="/app/controllers/productController.js"></script>
    <script src="/app/controllers/chartController.js"></script>
    <script src="/app/controllers/editController.js"></script>
  </body>

</html>
```

ng-model set to $scope.property.* will reflect all the property values in $scope.product. In the past, developers typically set values in input fields using jQuery or straight JavaScript for any type of manipulation of the DOM. When building an application with the MVVM pattern (regardless of the framework), best practice is to manipulate the DOM only through changes to the ViewModel, never directly (for example, with JavaScript or jQuery). I'm definitely not implying there's anything wrong with JavaScript or jQuery, but if you decide to use a pattern to solve a specific problem (in my case, MVVM to maintain the separation of concerns between the View, ViewModel and Model), it should be consistent throughout your application.

Note that you could implement a POST server-side action on Node.js, which is what is typically done with the ASP.NET Web API. However, the purpose here is to demonstrate how to do this with Node.js and OData:

```
app.post('/api/updateProduct', function(req, res) {
  var product = req.body;
  // Process update here, typically what is done with the ASP.NET Web
API
});
```

For the chart view (NodejsWebApp/public/app/views/chart.html), you need just one line of markup:

```
<kendo-chart k-options="options"></kendo-chart>
```

Figure 11 **The Product Controller**

```
myApp.controller("productController",
  function($scope, northwind, $location) {

    var dataSource =
      northwind
        .Products
        .asKendoDataSource({ pageSize: 10 });

    $scope.options = {
      dataSource: dataSource,
      filterable: true,
      sortable: true,
      pageable: true,
      selectable: true,
      columns: [
        { field: "ProductID" },
        { field: 'ProductName' },
        { field: "EnglishName" },
        { field: "QuantityPerUnit" },
        { field: "UnitPrice" },
        { field: 'UnitsInStock' },
        { command: ["edit", "destroy"] }
      ],
      toolbar: [
        "create",
        "save",
        "cancel",
        {
          text: "View Detail",
          name: "detail",
          template: $("#viewDetail").html()
        }
      ],
      editable: "inline"
    };

    $scope.viewDetail = function(e) {
      var selectedRow = $scope.grid.select();

      if (selectedRow.length == 0)
        alert("Please select a row");

      var dataItem = $scope.grid.dataItem(selectedRow);;

      $location.url("/edit/" + dataItem.ProductID);
    };
  });
```

Figure 12 **The Edit.html File**

```
<div class="demo-section">
  <div class="k-block" style="padding: 20px">
    <div class="k-block k-info-colored">
      <strong>Note: </strong>Please fill out all of the fields in this form.
    </div>
    <div>
      <dl>
        <dt>
          <label for="productName">Name:</label>
        </dt>
        <dd>
          <input id="productName" type="text"
            ng-model="product.ProductName" class="k-textbox" />
        </dd>
        <dt>
          <label for="englishName">English Name:</label>
        </dt>
        <dd>
          <input id="englishName" type="text"
            ng-model="product.Englishname" class="k-textbox" />
        </dd>
        <dt>
          <label for="quantityPerUnit">Quantity Per Unit:</label>
        </dt>
        <dd>
          <input id="quantityPerUnit" type="text"
            ng-model="product.QuantityPerUnit" class="k-textbox" />
        </dd>
        <dt>
          <label for="unitPrice">Unit Price:</label>
        </dt>
        <dd>
          <input id="unitPrice" type="text"
            ng-model="product.UnitPrice" class="k-textbox" />
        </dd>
        <dt>
          <label for="unitsInStock">Units in Stock:</label>
        </dt>
        <dd>
          <input id="unitsInStock" type="text"
            ng-model="product.UnitsInStock" class="k-textbox" />
        </dd>
        <dt>
          <label for="reorderLevel">Reorder Level</label>
        </dt>
        <dd>
          <input id="reorderLevel" type="text"
            ng-model="product.ReorderLevel" class="k-textbox" />
        </dd>
        <dt>
          <label for="discontinued">Discontinued:</label>
        </dt>
        <dd>
          <input id="discontinued" type="text"
            ng-model="product.Discontinued" class="k-textbox" />
        </dd>
        <dt>
          <label for="category">Category:</label>
        </dt>
        <dd>
           <select
             kendo-drop-down-list="dropDown"
             k-data-text-field="'CategoryName'"
             k-data-value-field="'CategoryID'"
             k-data-source="categoryDataSource"
             style="width: 200px"></select>
        </dd>
      </dl>

      <button kendo-button ng-click="save()"
        data-sprite-css-class="k-icon k-i-tick">Save</button>
      <button kendo-button ng-click="cancel()">Cancel</button>

      <style scoped>
        dd {
          margin: 0px 0px 20px 0px;
          width: 100%;
        }

        label {
          font-size: small;
          font-weight: normal;
        }

        .k-textbox { width: 100%; }

        .k-info-colored {
          margin: 10px;
          padding: 10px;
        }
      </style>
    </div>
  </div>
</div>
```

All that's happening here is declaring the Kendo UI Bar Chart directive, setting these options to bind to a property in the controller named options. **Figure 14** shows the product chart view and **Figure 15** shows the product chart controller.

As with productController.js, here you also inject the northwindFactory as northwind in the controller construction function, again creating a Kendo dataSource with the JayData helper asKendoDataSource method. Here are more details about what happens in the chart controller:

$scope.options.series

- type: This configures the type of chart.
- field: The field from the model/entity that will be used for the series (x-axis) value.

Figure 13 **The editController.js File**

```
myApp.controller("editController",
  function($scope, northwind, $routeParams, $location) {

    var productId = $routeParams.id;

    $scope.categoryDataSource = northwind.Categories.asKendoDataSource();

    northwind
      .Products
      .include("Category")
      .single(
        function(product) {
          return product.ProductID == productId;
        },
        { productId: productId },
        function(product) {
          $scope.product = product;
          northwind.Products.attach($scope.product);
          $scope.dropDown.value($scope.product.Category.CategoryID);
          $scope.$apply();
        });

    $scope.save = function() {
      var selectedCategory = $scope
                            .categoryDataSource
                            .get($scope.product.Category.CategoryID);

      console.log("selecctedCategory: ", selectedCategory.
innerInstance());
      $scope.product.Category = selectedCategory.innerInstance();
      // Unwrap kendo dataItem to pure JayData object
      northwind.saveChanges();
    };

    $scope.cancel = function() {
      $location.url("/product");
    };

  });
```

$scope.options.valueAxis

- majorUnit: The interval between major divisions. If the valueAxis.type is set to log, the majorUnit value will be used for the base of the logarithm.
- plotBands: The plot bands for the graph, which are used to visually indicate the product quantity. If the quantity falls below a specified level, the user should invoke the product restocking process.
- max: The maximum value for the y-axis.

$scope.options.categoryAxis

- field: The field labels for the x-axis.
- labels.rotation: The degrees to rotate the labels. Here you configure the labels on the x-axis to be perpendicular to the running x-axis by setting the value to -90 (degrees), that is, rotate the labels counterclockwise by 90 degrees.
- majorGridLines.visible: Turns the major gridlines on or off. You may want to turn them off for cosmetic reasons, to give the chart a cleaner and more polished look.
- tooltip.visible: This enables tooltips when a user hovers over a vertical bar.

Please see the Kendo UI Chart API for details at bit.ly/1owgWrS.

Azure Web Site Deployment

Because the source code is conveniently hosted in a CodePlex Git repository, using Azure Web Sites to set up continuous deployment (continuous delivery) is as simple as it gets:

1. Navigate to your Azure Web site dashboard and select Set up deployment from source control.
2. Select your repository; for this example, select CodePlex.
3. Click Next.
4. Select your CodePlex project.
5. Select the branch.
6. Click Check.
7. For every sync to your Git repository, a build and deployment will occur.

That's it. With a few simple clicks, your application is deployed with continuous integration and delivery. For more information on deploying with Git, visit bit.ly/1ycBo9S.

As a .NET developer, I've come to really enjoy how quick and easy it is to build CRUD-heavy applications with ASP.NET MVC, ASP.NET Web API, OData, Entity Framework, AngularJS and Kendo UI. Now, by developing on the MEAN stack, I can still leverage much of this domain knowledge and experience, with the help of the JayData libraries. The only difference between the two stacks is the server-side layer. If you've been developing with ASP.NET MVC and the ASP.NET Web API, Node.js shouldn't present many problems because you already have some basic JavaScript experience. You'll find complete source code for the example in this article at msdnmeanstack.codeplex.com, and a live demo at meanjaydatakendo.azurewebsites.net. ■

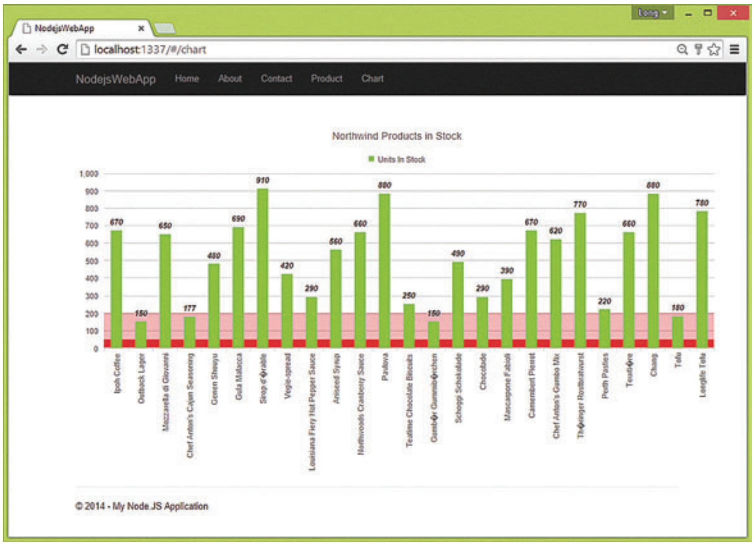


Figure 14 **Product Chart View**

Figure 15 **The Product Chart Controller**

```
myApp.controller("chartController",
  function($scope, northwind) {

    var dataSource = northwind.Products.asKendoDataSource();

    $scope.options = {
      theme: "metro",
      dataSource: dataSource,
      chartArea: {
        width: 1000,
        height: 550
      },
      title: {
        text: "Northwind Products in Stock"
      },
      legend: {
        position: "top"
      },

      series: [
        {
          labels: {
            font: "bold italic 12px Arial,Helvetica,sans-serif;",
            template: '#= value #'
          },
          field: "UnitsInStock",
          name: "Units In Stock"
        }
      ],
      valueAxis: {
        labels: {
          format: "N0"
        },
        majorUnit: 100,
        plotBands: [
          {
            from: 0,
            to: 50,
            color: "#c00",
            opacity: 0.8
          }, {
            from: 50,
            to: 200,
            color: "#c00",
            opacity: 0.3
          }
        ],
        max: 1000
      },
      categoryAxis: {
        field: "ProductName",
        labels: {
          rotation: -90
        },
        majorGridLines: {
          visible: false
        }
      },
      tooltip: {
        visible: true
      }
    };
  });
```

Long Le *is the principal app/dev architect at CBRE Inc. and a Telerik/Kendo UI MVP. He spends most of his time developing frameworks and application blocks, providing guidance for best practices and patterns, and standardizing the enterprise technology stack. In his spare time, he enjoys blogging (blog.longle.net), playing Call of Duty, or mentoring (codementor.io/lelong37). You can reach and follow him on Twitter at twitter.com/LeLong37.*

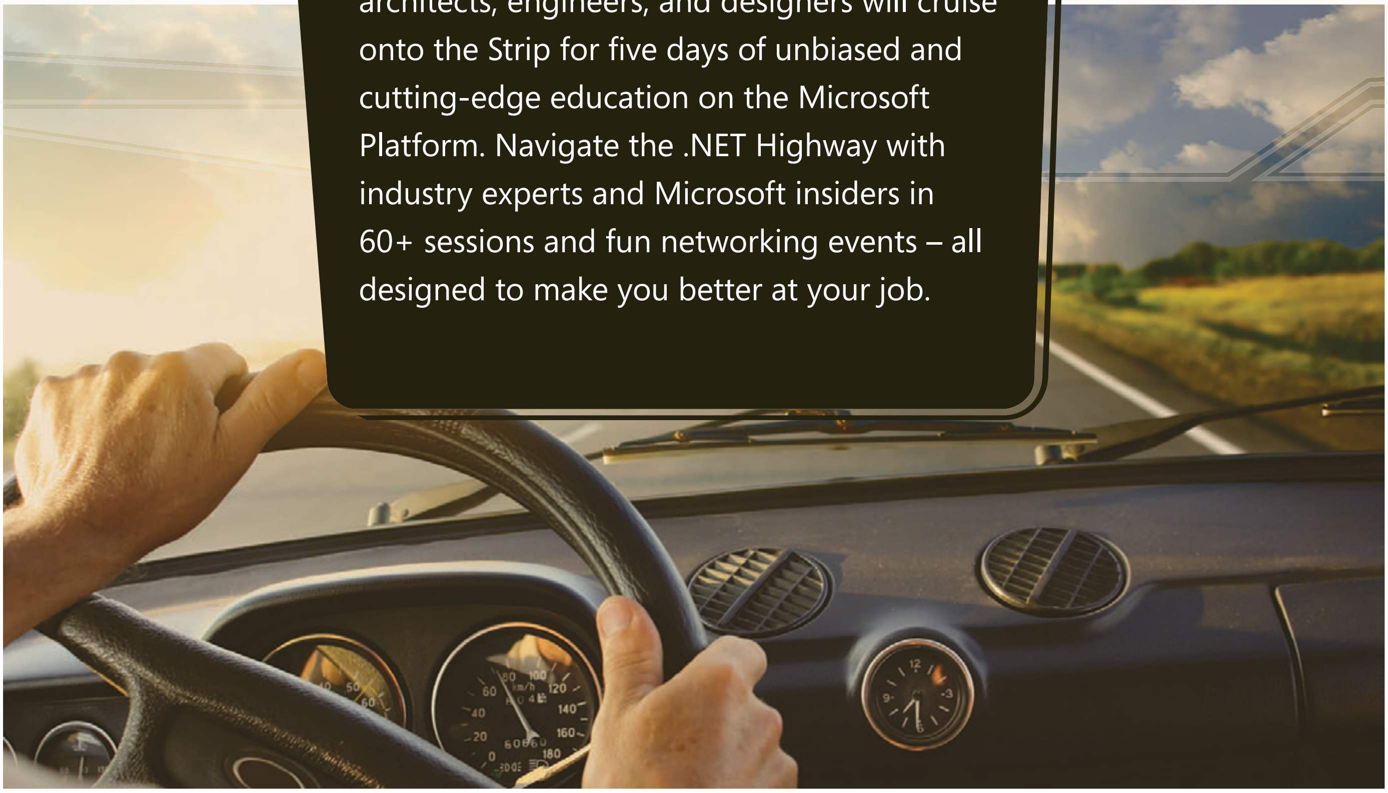
Thanks to the following technical experts for reviewing of this article:
Robert Bany (JayData), Burk Holland (Telerik) and Peter Zentai (JayData)

Visual Studio® LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/lasvegas
Las Vegas MARCH 16 – 20
BALLY'S HOTEL & CASINO LAS VEGAS, NV

Code on the Strip

Visual Studio Live!'s first stop on its 2015 Code Trip is Las Vegas, situated fittingly near Historic Route 66. Developers, software architects, engineers, and designers will cruise onto the Strip for five days of unbiased and cutting-edge education on the Microsoft Platform. Navigate the .NET Highway with industry experts and Microsoft insiders in 60+ sessions and fun networking events – all designed to make you better at your job.



SUPPORTED BY
Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY
1105 MEDIA INC

NAVIGATE THE .NET HIGHWAY

TRACKS INCLUDE:

- Visual Studio / .NET
- JavaScript/HTML5
- ASP.NET
- Cross-Platform Mobile Development
- Windows 8.1/WinRT
- Database and Analytics
- Cloud Computing
- Windows Phone

Register NOW and Save $500!

Use promo code VSLDEC2

CONNECT WITH VISUAL STUDIO LIVE!

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

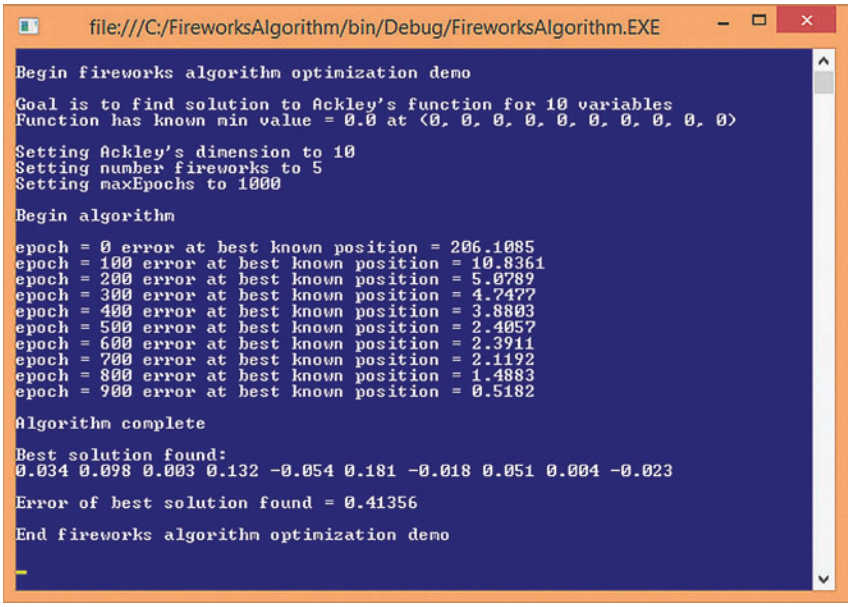
Scan the QR code to register or for more event details.

vslive.com/lasvegas

Test Run

JAMES MCCAFFREY

Fireworks Algorithm Optimization



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

```
Begin fireworks algorithm optimization demo

Goal is to find solution to Ackley's function for 10 variables
Function has known min value = 0.0 at (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Setting Ackley's dimension to 10
Setting number fireworks to 5
Setting maxEpochs to 1000

Begin algorithm

epoch = 0 error at best known position = 206.1085
epoch = 100 error at best known position = 10.8361
epoch = 200 error at best known position = 5.0789
epoch = 300 error at best known position = 4.7477
epoch = 400 error at best known position = 3.8803
epoch = 500 error at best known position = 2.4057
epoch = 600 error at best known position = 2.3911
epoch = 700 error at best known position = 2.1192
epoch = 800 error at best known position = 1.4883
epoch = 900 error at best known position = 0.5182

Algorithm complete

Best solution found:
0.034 0.098 0.003 0.132 -0.054 0.181 -0.018 0.051 0.004 -0.023

Error of best solution found = 0.41356

End fireworks algorithm optimization demo
```

Figure 1 **Fireworks Algorithm Optimization in Action**

Many machine learning software systems use numerical optimization. For example, in logistic regression classification, training the classifier is the process of finding a set of values for the weights associated with the input variables so that for a collection of training data, the computed output values closely match the known output variable values. In other words, the goal is to optimize (minimize) the error between computed and desired output values.

There are many different numerical optimization algorithms. Back-propagation is based on classical calculus techniques and is often used with neural networks. Particle swarm optimization mimics group behavior such as the flocking of birds. Evolutionary algorithm optimization, a form of genetic algorithm optimization, models the biological processes of chromosome replication.

This article describes a relatively new (first published in 2010) optimization technique called fireworks algorithm optimization (FAO). The technique doesn't explicitly model or mimic the behavior of fireworks, but when the algorithm is visually displayed, the resulting image resembles the geometry of exploding fireworks.

Many machine learning software systems use numerical optimization.

The best way to see where this article is headed and to get an idea of what FAO is, is to take a look at the demo program in **Figure 1**. The goal of the demo program is to use FAO to find the minimum value of Ackley's function with 10 variables. Ackley's function is a standard benchmark for evaluating the effectiveness of numerical optimization algorithms. The demo version has a minimum value of 0.0 located at (0, 0, 0, 0, 0, 0, 0, 0, 0, 0). Ackley's function is tricky because it has many local minimum solutions that can trap search algorithms before finding the one global minimum. A graph of Ackley's function of two variables is shown in **Figure 2**.

Code download available at msdn.microsoft.com/magazine/msdnmag1214.

The demo program sets the number of fireworks to 5. More fireworks increase the chance of finding a true optimal solution, at the expense of performance. FAO typically works well with 3 to 10 fireworks. FAO is an iterative process and requires a maximum loop counter value. A loop counter variable in machine learning optimization is often named "epoch" and the demo sets the maximum value to 1,000 iterations. This is a bit small, intended for demo purposes, and in realistic scenarios, values from 10,000 to 100,000 are typical.

In the demo run in **Figure 1**, the best (smallest) error associated with the best position found so far is displayed every 100 epochs. Notice that FAO starts converging very quickly. After 1,000 epochs, the best position found is (0.034 0.098 0.003 0.132 -0.054 0.181 -0.018 0.051 0.004 -0.023) and the associated error is 0.41. If the maximum number of epochs is increased to 10,000, then FAO will in fact find the optimal solution. FAO, like most numerical optimization algorithms, isn't guaranteed to find an optimal solution in realistic scenarios where the optimal solution isn't known.

This article assumes you have at least intermediate programming skills, but doesn't assume you know anything about numerical

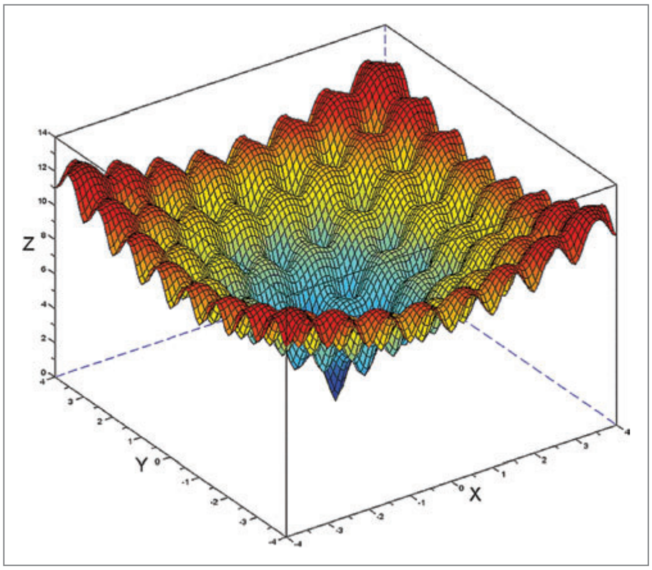


Figure 2 **Ackley's Function of Two Variables**

optimization or FAO. The demo program is coded using C#, but you shouldn't have too much difficulty refactoring the code to another language, such as JavaScript or Python.

The demo code is a bit too long to present in its entirety, but the complete source code is available in the code download that accompanies this article. The demo code has most normal error checking removed to keep the main ideas as clear as possible and the size of the code small.

Overall Program Structure

The overall program structure, with a few minor edits to save space, is presented in **Figure 3**. To create the demo, I launched Visual Studio and created a new C# console application named FireworksAlgorithm. The demo has no significant .NET dependencies, so any recent version of Visual Studio will work.

After the template code loaded into the Visual Studio editor, in the Solution Explorer window I renamed file Program.cs to the more descriptive FireworksProgram.cs and Visual Studio automatically renamed class Program for me. At the top of the source code, I deleted all using statements that pointed to unneeded namespaces, leaving just the references to System and Collections.Generic.

I coded the demo using a static-method approach rather than an object-oriented programming (OOP) approach. The demo has all the control logic in the Main method, which calls two public methods, Solve and Error. The essential calling statements are:

```
int dim = 10;
int n = 5;
int maxEpochs = 1000;
double[] bestPosition = Solve(dim, n, maxEpochs)
double error = Error(bestPosition);
```

The variable dim holds the number of problem dimensions. In the case of machine learning training, this would normally be the number of model weights to determine. Variable n is the number of fireworks. I use as much as possible the rather terse variable names, such as n, that are used in the research papers on FAO so you can use those papers as a resource more easily. The FAO technique is contained in method Solve. Method Error accepts a position (an array of 10 values), computes the value of Ackley's function for those values, and returns the average of the sum of squared differences between the computed outputs and the known minimum of 0.0.

The demo program has a utility class named Info that encapsulates a position and its associated error. The class has no associated methods, such as an explicit constructor, in order to make it easier for you to refactor the demo code to languages with limited OOP support.

Understanding the Algorithm

The fireworks algorithm process is illustrated in the graph in **Figure 4**. The image represents a simplified dummy minimization problem, not the Ackley's function of the demo program. The goal of

Figure 3 **Overall Program Structure**

```
using System;
using System.Collections.Generic;
namespace FireworksAlgorithm
{
  class FireworksProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("\nBegin fireworks algorithm demo\n");
      Console.WriteLine("Goal is to solve Ackley's function");
      Console.WriteLine("Function has min value = 0.0 at (0, .. ))");

      int dim = 10;
      int n = 5;    // Number of fireworks
      int maxEpochs = 10000;
      Console.WriteLine("\nSetting Ackley's dimension to " + dim);
      Console.WriteLine("Setting number fireworks to " + n);
      Console.WriteLine("Setting maxEpochs to " + maxEpochs);

      Console.WriteLine("\nBegin algorithm\n");
      double[] bestPosition = Solve(dim, n, maxEpochs);
      Console.WriteLine("\nAlgorithm complete");

      Console.WriteLine("\nBest solution found: ");
      for (int i = 0; i < dim; ++i)
        Console.Write(bestPosition[i].ToString("F3") + " ");
      Console.WriteLine();

      double error = Error(bestPosition);
      Console.WriteLine("\nError of best solution found = " +
        error.ToString("F5"));

      Console.WriteLine("\nEnd fireworks algorithm demo\n");
      Console.ReadLine();
    } // Main

    public static double Error(double[] position) { . . }
    public static double[] Solve(int dim, int n, int maxEpochs) { . . }

    private static int[] PickDimensions(int dim, int z, int seed) { . . }
    private static double YMax(Info[] fireworks) { . . }
    private static double YMin(Info[] fireworks) { . . }
    private static int[] NumberSparks(Info[] fireworks, int m,
      double a, double b) { . . }
    private static double[] Amplitudes(Info[] fireworks, int A,
      int epoch, int maxEpochs, double minX, double maxX) { . . }
    private static double MinAmplitude(int epoch, int maxEpochs,
      double minX, double maxX) { . . }
    private static void AddGaussianSparks(Info[] fireworks,
      List<Info>[] sparksList, int dim, int mHat, int epoch, double minX,
      double maxX, double[] bestPosition, ref double bestError, Random rnd)
  }

  public class Info
  {
    public double[] position;
    public double error;
  }

} // ns
```

the dummy problem is to minimize some arbitrary function that has two input variables, X and Y, where the function has a minimum value of 0.0 located at X = 0 and Y = 0.

The image in **Figure 4** uses three fireworks. Each firework has so-called sparks. There are two kinds of sparks, regular and Gaussian. **Figure 5** shows the fireworks algorithm in very high-level pseudo-code.

For the image in **Figure 4**, the positions of the three fireworks are indicated by the colored dot markers at (-6, 2), (3, 4), and (3, -3). Notice that the first firework is the worst because it's farthest away from the target position of (0, 0) and that the third firework is the best because it's closest. The amplitudes are indicated by the dashed circles around each firework. Good fireworks have smaller amplitudes and bad fireworks have larger amplitudes.

> The process of generating fireworks and then sparks is iterated until some sort of stopping condition is met.

Each firework will generate a different number of regular (non-Gaussian) sparks. Bad fireworks will generate fewer sparks than good fireworks. In **Figure 4**, firework[0] generates three sparks, firework[1] generates four sparks, and firework[2] generates five sparks. Each regular spark will be located within its parent firework's amplitude. Because good fireworks have small amplitude and relatively many sparks, the algorithm search will focus near good fireworks. Bad fireworks shouldn't be entirely neglected because they could lead to an optimal solution that's located out of range of the current fireworks. The Gaussian sparks are generated in such a way that they tend to be located between a firework and the best-known position of any spark encountered during the search.

After all regular and Gaussian sparks have been generated, each is evaluated. New fireworks for the next round of explosions are selected from the current sparks. The position of the best spark is retained as one of the new fireworks to intensify search in good locations. The position of the worst spark is retained to maintain search diversity and prevent the algorithm from quickly collapsing onto a solution that may not be optimal. The positions of the remaining fireworks, just one firework in the simple example in **Figure 4**, are randomly selected from the current sparks.

The process of generating fireworks and then sparks is iterated until some stopping condition is met. The stopping condition is just a maximum loop counter value in the case of the demo program. When using FAO for machine learning training, the stopping condition might also include an error threshold so that when error drops below some small specified value that depends on the particular problem being solved, the processing loop terminates.

Implementing the Fireworks Algorithm

The definition of method Solve begins as:

```
public static double[] Solve(int dim, int n, int maxEpochs)
{
  int m = n * 10;
  int mHat = 5;
  double a = 0.04;
  double b = 0.8;
  int A = 40;
  double minX = -10.0;
  double maxX = 10.0;
...
```

Variable m is the total number of regular sparks, which will be allocated to the n fireworks. Variable mHat (so named because research papers use a lowercase m with a carat over it) is the number of special Gaussian sparks to generate. Variables a and b control the minimum and maximum number of sparks for any particular firework. Variable A is the maximum amplitude for any firework. Variables minX and maxX hold the smallest and largest values for any single value in an Info object's position array.

Method Solve creates n initial fireworks, like so:

```
Random rnd = new Random(3);
Info[] fireworks = new Info[n];
for (int i = 0; i < n; ++i)
{
  fireworks[i] = new Info();
  fireworks[i].position = new double[dim];
  for (int j = 0; j < dim; ++j)
    fireworks[i].position[j] = (maxX - minX) * rnd.NextDouble() + minX;
  fireworks[i].error = Error(fireworks[i].position);
}
```

Random object rnd is initially using a seed value of 3 only because that value gave a representative demo run. Each of the n fireworks are

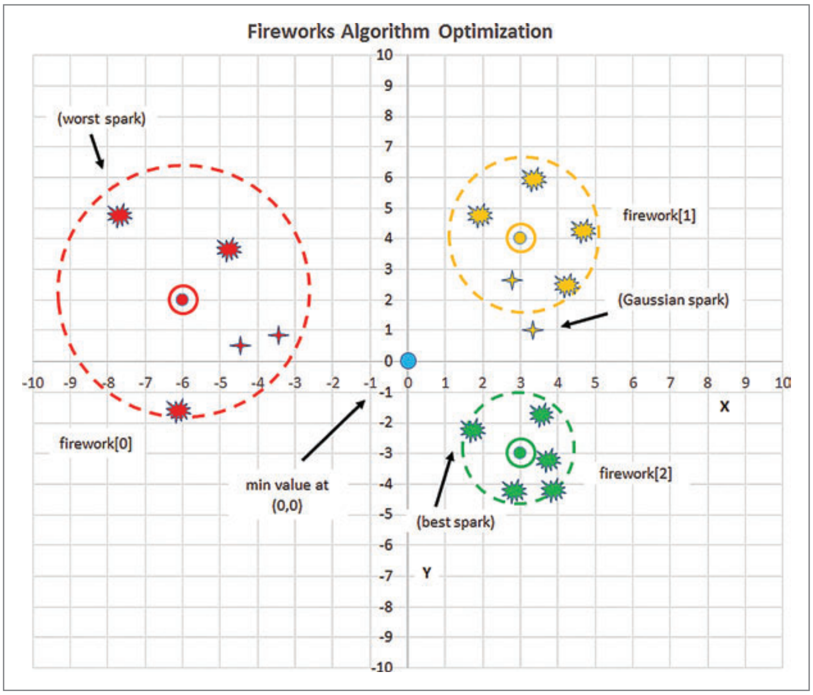


Figure 4 **The Fireworks Algorithm**

stored as Info objects in an array. The initialization code picks random values between minX and maxX for each cell of the position array. For some specific machine learning training scenarios, it's preferable to initialize the initial fireworks so they're far apart from each other.

Method Solve continues by establishing variables to hold the best position found by any spark, and that position's associated error. Unlike fireworks, which have a fixed number, the number of sparks per firework will vary in each iteration of the main processing loop, so sparks are stored in a List collection rather than an array:

```
double[] bestPosition = new double[dim];
for (int k = 0; k < dim; ++k)
  bestPosition[k] = fireworks[0].position[k];
double bestError = fireworks[0].error; // arbitrary

List<Info>[] sparksList = new List<Info>[n];
for (int i = 0; i < n; ++i)
  sparksList[i] = new List<Info>();
```

The main processing loop begins:

```
int epoch = 0;
while (epoch < maxEpochs)
{
  if (epoch % 100 == 0) // Show progress every 100 iterations
  {
    Console.Write("epoch = " + epoch);
    Console.WriteLine(" error at best known position = " +
      bestError.ToString("F4"));
  }
...
```

Here, progress is displayed every 100 epochs. You might want to consider passing a Boolean flag variable named "progress" to control whether progress messages are displayed:

```
int[] numberSparks = NumberSparks(fireworks, m, a, b);
double[] amplitudes = Amplitudes(fireworks, A, epoch, maxEpochs, minX, maxX);
for (int i = 0; i < n; ++i)
  sparksList[i].Clear(); // Number of sparks changed
```

Next, the number of sparks for each firework, and the maximum amplitude for each firework, are calculated using helper methods NumberSparks and Amplitudes. The number of sparks for a firework is proportional to the error of the firework so that good fireworks receive a larger proportion of the m total regular sparks. Similarly, each amplitude is proportional to the error, so that good fireworks have smaller amplitudes.

Next, each spark is instantiated:

```
for (int i = 0; i < n; ++i)
{
  double amp = amplitudes[i]; // Amplitude for curr firework
  int ns = numberSparks[i];   // Number of sparks for curr firework

  for (int j = 0; j < ns; ++j) // Each spark for curr firework
  {
    Info spark = new Info(); // A spark has a position and error
    spark.position = new double[dim]; // Allocate space (ctor doesn't)
    for (int k = 0; k < dim; ++k) // Spark position based on its parent firework
      spark.position[k] = fireworks[i].position[k];
```

Figure 5 **The Fireworks Algorithm in High-Level Pseudo-Code**

```
initialize 3 fireworks to random positions
loop until done
  for each firework
    calculate the amplitude of the firework
    calculate the number of regular sparks
    generate the regular sparks
  end for
  generate special Gaussian sparks
  evaluate each spark
  from the list of sparks,
    select 3 to act as positions of new fireworks
  create 3 new fireworks
end loop
return the position of the best spark found
```

A spark's position is based on its parent firework's position. Next, a few (z) of the dimensions of the position array are randomly selected, and a random displacement is added to the position array:

```
int z = (int)Math.Round(dim * rnd.NextDouble());
int[] dimensions = PickDimensions(dim, z, epoch);
for (int ii = 0; ii < dimensions.Length; ++ii)
{
  double h = amp * 2 * rnd.NextDouble() - 1;
  int k = dimensions[ii]; // convenience
    spark.position[k] += h; // displace from parent
  if (spark.position[k] < minX || spark.position[k] > maxX)
    spark.position[k] = (maxX - minX) * rnd.NextDouble() + minX;
}
spark.error = Error(spar.position);
sparksList[i].Add(spark);
```

After generating a spark, method Solve checks to see if the new spark has the best position found so far:

```
    if (spark.error < bestError)
    {
      bestError = spark.error;
      for (int k = 0; k < dim; ++k)
        bestPosition[k] = spark.position[k];
    }
  } // Each new regular spark
} // Each firework
```

Next, special Gaussian sparks are generated:

```
AddGaussianSparks(fireworks, sparksList, dim, mHat,
  epoch, minX, maxX, bestPosition, ref bestError, rnd);
```

Helper method AddGaussianSparks generates special sparks so that their positions tend to be between a randomly selected firework and the best-known position. Method Solve concludes by finding the best and worst spark generated and using their positions for two of the new fireworks for the next iteration. The remaining n-2 fireworks are created using randomly selected sparks:

```
...
    // Find best, worst spark
    // Create 2 new fireworks
    // Create remaining n-2 fireworks

    ++epoch;
  } // main loop
  return bestPosition;
} // Solve
```

See the code download for implementation details.

A Few Comments

The original paper that presented the fireworks algorithm is "Fireworks Algorithm for Optimization," by Y. Tan and Y. Zhu (2010). That paper contained several errors and factors that made the algorithm impractical for real-life optimization. These flaws were quickly noted by other researchers. My article is based primarily on the 2013 research paper, "Enhanced Fireworks Algorithm," by S. Zheng, A. Janecek and Y. Tan. You can find both papers in several locations on the Internet. I have made quite a few simplifications and minor modifications to the algorithms presented in both research papers. In my opinion, there isn't enough research evidence yet to answer the question of whether fireworks optimization is better than or worse than other optimization algorithms. But it sure is interesting. ■

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

Thanks to the following Microsoft technical experts for reviewing this article: *Todd Bello, Kenneth Griffin, Yong Liu, Brian Mellinger and Esin Saka*

Build a Better UX with Live Tiles

Live tiles are an effective way to introduce your app. You can display information about your app without requiring a user to enter the app itself. As part of the UX feature of the Windows platform, live tiles should be alluring and inviting, filled with relevant and click-worthy content.

Live tiles are your chance to make a great first impression. The user doesn't care about the code or any technical details. All he cares about is the experience that he'll have using your app. Is it easy to use? Is it intuitive? These things are more important to the user than what's under the hood.

Live tiles are the entryway to your app, that first impression that tells the user about your app's usability. Everyone who uses your app will see and use your live tile before any other part of the app. It doesn't matter whether it's to launch the app or just snack on tidbits of data you display in them. The live tile is how you can make that all-important first impression.

This means you had better ensure your tiles and the splash screen are enticing and attractive. That's not to say you can ignore the rest of the app, but you'll never get users to that point without leading off with some good-looking tiles. As you probably imagine, you can update live tiles with fresh information periodically, via push notifications. You can even extend notifications to the lock screen by adding badges to show app-related information.

Figure 1 illustrates the four styles of live tiles available on Windows tablets and desktops: large, wide, medium and small. The images in **Figure 1** show how The Weather Channel has capitalized on a clean design when the tiles aren't displaying info. When they are, you see rich graphics and the tiles display only the most important information. In this case, it's the current weather and any severe storm warnings.

A better UX will lead to better ratings in the store and better user satisfaction. Live tiles with your branding are the best way to differentiate your app from others. **Figure 1** demonstrates how tiles can be part of a great experience. Users don't need to open the app to see the current weather. On the large primary tile, they can set up to three cities to track weather, and then track one city at a time on the others. That's squeezing some serious data into just 310x310 pixels, or an even smaller playing field.

How Live Tile Graphics Work

Now that I've explained why live tiles are important, it's time to understand how graphics work with them. Fortunately, Microsoft did a great job creating an easy-to-follow developer-friendly UX paradigm with a set of design principles that works well on tablets, phones and smaller, touch-first devices. You can publish apps that look good without hiring a designer. That's not to say you should eschew designers altogether. Some apps, especially games, have complicated graphics that require a designer's input.

This article will cover some live tile designs that non-designing developers can easily accomplish. As part of that set of principles, live tiles enable the user to consume data from your app without actually interacting with your app. This is the height of convenience, as the user can quickly scan the home screen of his device to get relevant information from multiple sources in a flash.

There are two kinds of graphics in a modern app: user-generated and app graphics. Family vacation or professional photos are considered user-generated. The images within live tiles or on the app bar are app graphics. When constructing tiles, you can use either kind to convey information. Visual Studio includes templates that follow the Microsoft modern design that will help you achieve a professional presentation. This means everything you need is ready in the package.appxmanifest file. All you need to do is provide the graphics themselves in the correct size.

It's important to have proper branding for tiles that aren't live yet. As you can see in **Figure 2**, the tiles have a simple design with a quote balloon and the initials for Quote of the Day (QotD). It's easy to find and determine what app this is with its clean design

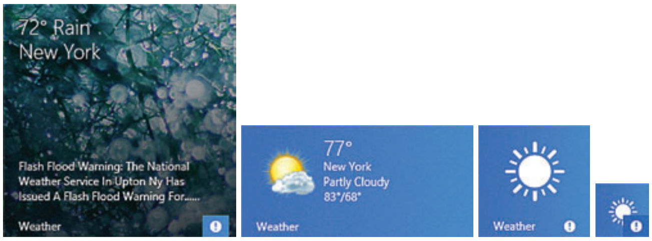

72° Rain
New York
Flash Flood Warning: The National Weather Service In Upton Ny Has Issued A Flash Flood Warning For......
Weather
77°
New York
Partly Cloudy
83°/68°
Weather
Weather

Figure 1 **The Four Sizes of Live Tiles—Large, Wide, Medium and Small**

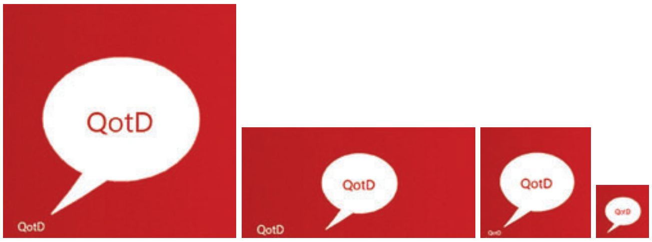


Figure 2 **The Live Tiles for the Quote of the Day App**

JetBRAINS

ReSharper
jetbrains.com/msdn

The legendary extension
to Visual Studio.*

* WARNING! PROLONGED USE MAY CAUSE ADDICTION.

and bright red color. For now, they're standard, static tiles. That will change when you start sending updates.

Using graphics in a Windows Store app or Windows Phone Store app requires several images for the following assets:

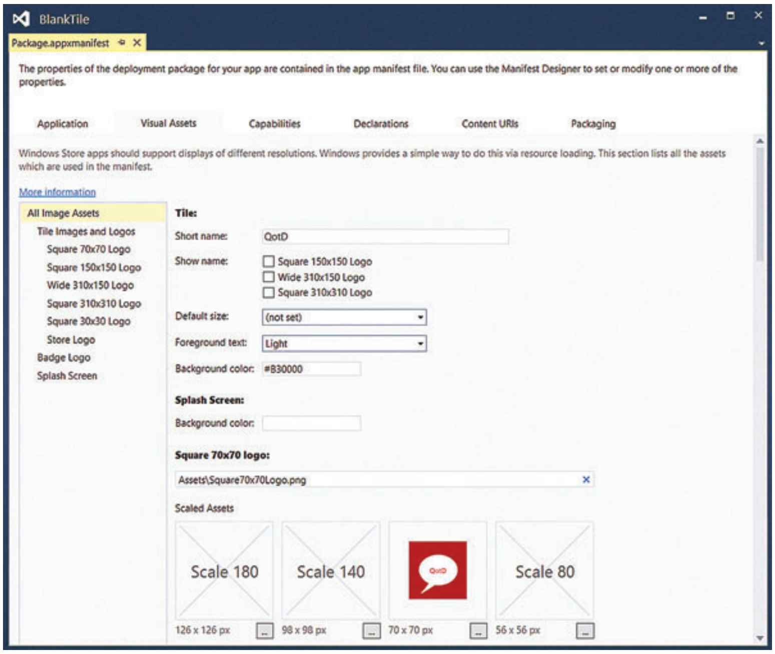
- Live tiles: The large or small squares or rectangles the user taps or clicks to launch an app. These often contain images or text.
- Badges: Small icons or glyphs on the live tiles that denote app status. **Figure 1** shows a badge in both the large and small tiles.
- Splash screen: The introductory screen that displays just before the app loads. Frequently, this screen contains a logo or graphic.
- In-app tiles: These tiles display app data and live in a GridView or ListView. They look and feel like live tiles on the Windows Start page.

These images work best using the .png and .jpg formats. Each Windows Store app and Windows Phone Store app can contain many different sizes of these image assets. You can configure which images belong to which tile by editing the project's package manifest. The package manifest is a file where you can set up all app-specific settings, such as the name, version number, graphic assets and capabilities declarations. **Figure 3** shows the logos you define in the package.appxmanifest in a Windows Store app.

You should supply at least one graphic for each of the following asset types, or the app will show their default icons instead:

- Square 70x70 Logo: A 70x70 pixel image
- Square 150x150 Logo: A 150x150 pixel image
- Wide 350x150 Logo: A 350x150 pixel image
- Square 310x310 Logo: A 310x310 pixel image
- Square 30x30 Logo: A 30x30 pixel image

You can—and should—supply scaled graphics for these tile classes. The scaled assets for the items in **Figure 3** are at 80 percent, 100 percent, 140 percent and 180 percent. The bottom of **Figure 3** lists the precise dimensions required for the scaled assets. Note that these are just for the Windows Store apps. Windows Phone Store app projects have slightly different sets of assets and scales that work for the smaller device types. Both Windows Store and Windows Phone Store apps have the following logo and Splash screen configurations:



The properties of the deployment package for your app are contained in the app manifest file. You can use the Manifest Designer to set or modify one or more of the properties.

Windows Store apps should support displays of different resolutions. Windows provides a simple way to do this via resource loading. This section lists all the assets which are used in the manifest.

Tile:

Short name: QotD

Show name:
☐ Square 150x150 Logo
☐ Wide 310x150 Logo
☐ Square 310x310 Logo

Default size: (not set)

Foreground text: Light

Background color: #B30000

Splash Screen:

Background color:

Square 70x70 logo:

Assets\Square70x70Logo.png

Scaled Assets

126 x 126 px | 98 x 98 px | 70 x 70 px | 56 x 56 px

Figure 3 **Configure the Tile Graphics in the App's Package.appxmanifest File**

- Store logo: The logo displayed in the store.
- Badge logo: A small icon that can display on the lock screen or inside existing tiles.
- Splash screen: A full screen page usually containing a logo or introductory text shown at app launch.

There are many versions of these images because of the need to scale them to size correctly on the various screen configurations, from desktop to mobile to tablet and others. A 150x150 pixel image doesn't have to stretch too far before it starts to look pixelated and blurry.

You can create graphics for tiles using any photo or graphics editing program. With these tools, it's not difficult to create a logo like **Figure 2** that's simple yet noticeable. There are many stock graphics also available for use in apps.

Code Tiles from Templates

Tiles would be little more than plain icons if not for their ability to display different images and text at different times. It's these regular updates that make them live. You can update tiles with code or integrate them with push notifications. If you use push, you can update tiles using background tasks that fit into the process lifecycle. This way your app doesn't need to be running, per se, yet some code can run at predetermined intervals to cycle through five different items. For background information on process lifecycle management, see "The Windows Store App Lifecycle" at msdn.microsoft.com/magazine/jj660301.

The code in **Figure 4** performs a simple tile update to generate the tiles in **Figure 5**, without push notifications. It first obtains a TileUpdater object from calling the static CreateTileUpdaterForApplication of the TileUpdateManager class. Once that's done, the code moves onto fetching a predefined tile template by passing in one of the TileTemplateType enum options to the GetTemplateContent method of the TileManager class.

Accessing the innerText attribute or calling SetAttribute lets the code work with the tile. That's because an HTML template is what describes how a tile looks and feels. Fortunately, XAML developers don't need to learn a lot of HTML just to make a tile. It does look like HTML, after all. However, you should know about elements like the HTML here that defines the TileSquare150x150Text04 template.

```
<tile>
  <visual version="2">
    <binding template="TileSquare150x150Text04" >
      <image id="1" src=""/>
      <text id="1"></text>
    </binding>
  </visual>
</tile>
```

You can't just whip up any old tile out of any old dimensions, and you don't write the template HTML yourself. Instead, there are several templates from which you can choose that let you edit the basic tile shapes, icons and messages. There's a listing of these templates for both Windows 8.1 and Windows Phone 8.1, along with code samples at bit.ly/1oqwQd6.

Pick one of the templates from the list and use it in your code. In **Figure 4**, the call to GetTemplateContent is what fetches this pre-defined template. You can choose from any of the numerous templates available.

Up to this point, the code samples have dealt with simply updating a tile once, on the spot, by executing some code. If you want to add some extra pizazz, update tiles on a periodic basis with fresh information. You can do this by adding background tasks and push notes to your app. Because this article's focus is more UX-related, you might want to view the video at bit.ly/1vH6J2p to learn how to update a tile from a background task.

Use Secondary Tiles for a First-Class UX

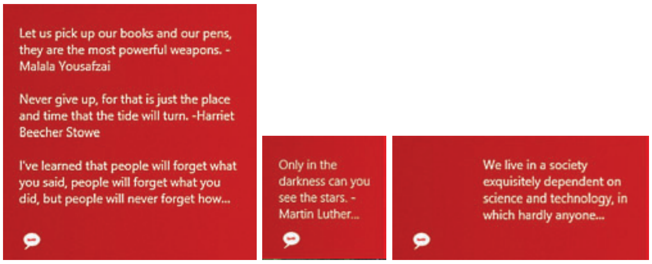
Users occasionally want to pin some piece of data to the Start screen. Airline boarding passes, an eBook, or specific stock ticker symbols are all good examples of things users want to pin to their screen. The way to do that in Windows 8.1 or Windows Phone 8.1 is with secondary tiles. Secondary tiles make it easy for the user to customize her app, which translates to higher ratings in the store.

There's a concept of deep linking in the app world. This means a user navigates a few times within an app, then pins it to link to it later. Here are several uses for secondary tiles:

- Weather for a specific city
- Travel itineraries for a specific trip
- Saved games
- Specific movies to start or resume
- A friend's contact information
- Your favorite pictures
- Custom alarms

Figure 4 **Simple Tile Update Without Push Notifications**

```
var updater = TileUpdateManager.CreateTileUpdaterForApplication();
var square150x150 =
  TileUpdateManager.GetTemplateContent(
    TileTemplateType.TileSquare150x150Text04);
square150x150.GetElementsByTagName("text")[0].InnerText =
  "Only in the darkness can you see the stars. -Dr. Martin Luther King, Jr.";
updater.Update(new TileNotification(square150x150));
var wide310x150 =
  TileUpdateManager.GetTemplateContent(
    TileTemplateType.TileWide310x150SmallImageAndText03);
wide310x150.GetElementsByTagName("text")[0].InnerText =
  "We live in a society exquisitely dependent on science and
  technology, in which hardly anyone knows anything about
  science and technology. - Carl Sagan";
((XmlElement)wide310x150.GetElementsByTagName("image")[0]).SetAttribute("src",
  "ms-appx:///Assets/qotd310x150-sm.png");
updater.Update(new TileNotification(wide310x150));
var large310x310 =
  TileUpdateManager.GetTemplateContent(TileTemplateType.
TileSquare310x310TextList02);
large310x310.GetElementsByTagName("text")[0].InnerText =
  "Let us pick up our books and our pens, they are the most powerful
  weapons. -Malala Yousafzai";
large310x310.GetElementsByTagName("text")[1].InnerText =
  "Never give up, for that is just the place and time that the tide will turn.
  -Harriet Beecher Stowe";
large310x310.GetElementsByTagName("text")[2].InnerText =
  "I've learned that people will forget what you said, people
  will forget what you did, but people will never forget
  how you made them feel. -Maya Angelou";
updater.Update(new TileNotification(large310x310));
```



Let us pick up our books and our pens, they are the most powerful weapons. - Malala Yousafzai

Never give up, for that is just the place and time that the tide will turn. -Harriet Beecher Stowe

I've learned that people will forget what you said, people will forget what you did, but people will never forget how...

Only in the darkness can you see the stars. - Martin Luther...

We live in a society exquisitely dependent on science and technology, in which hardly anyone...

Figure 5 **Several of the Tile Sizes You Can Use**

Keep in mind these tiles are secondary because you already have a primary live tile. App-wide updates should go in the primary tile, though there aren't any hard-and-fast rules there. The types of particular data listed for secondary tiles wouldn't usually make sense in the main tile. Users typically launch an app, and then go to its start page. Secondary tiles let users launch an app and go directly to a specific part of the app. The term for this is "deep linking." As a rule of thumb, use primary tiles to launch to the app's Start screen and secondary tiles for specific data details.

The following code creates and displays a secondary tile:

```
private void AppBarButton_Click(object sender, RoutedEventArgs e)
{
  string displayName = "Secondary tile";
  string tileActivationArguments = "Look both ways before crossing the street";
  System.Uri square150x150Logo = new System.Uri("ms-appx:///Assets/Logo.
scale-100.png");
  SecondaryTile secondaryTile = new SecondaryTile("Pinned",
    displayName,
    tileActivationArguments,
    square150x150Logo,
    TileSize.Square150x150);
}
```

When calling the SecondaryTile constructor, pass in all the information you need to populate the tile, such as the name, arguments, size and which image to use. Everything you need is right there in the call, and the constructor is overloaded, so you have many choices.

Secondary tiles are similar to standard tiles, but they allow that deep linking to content or a specific part of the app. The more user-friendly you make both sets of tiles, the better your reviews in the store. So consider using secondary tiles if your app contains specific data that's useful to view from the Windows Start page.

A Square Conclusion

When a user doesn't have to do anything to interact with your app, and yet receives useful information, that's a win. The ratings in the app store will reflect that. Even developers who aren't big into graphic design can still create good-looking and useful apps.

Remember to create graphics for as many scaled sizes as possible, and Windows will deliver the best graphic depending on the device. The better the UX, the better the rating in the app store. And this translates directly to more downloads. ■

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 rachelappel.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *Frank La Vigne*

DON'T GET ME STARTED

DAVID S. PLATT

My Biggest Misteaks

I know I must seem omniscient to you. Because I get to pick my topics, my predictions in this column always come true—for example, last November I predicted that I wouldn't be named the new president of Microsoft.

I know you'll be astounded to hear that I've actually made one or two mistakes in my life, blown calls on the future of technology that I didn't initially see the use of, but which later evolved to become ubiquitous. Perhaps not on the scale of Bill Gates saying, "No one would ever need a computer with more than 640k of memory," but still: Here are some of the biggest ones that I've blown.

A Microsoft evangelist was talking to a small group of us digerati in 2001, explaining their upcoming Web platform, code-named "HailStorm" (bit.ly/X009Z3). She said, "Your PC will be able call your kids' school's computer and automatically download their schedules into your Outlook so you'll know when vacations are." I exploded at her: "What planet are you on?" I thought HailStorm was a classic example of Microsoft hyper-geeks building something that they themselves would like, and thinking the world would like it because users are just like them. Ubiquitous downloadable schedules didn't happen with the single-vendor, pay-only plans that Microsoft then had for HailStorm. But with the vendor-independent RFC 5545 calendar data standard, and with smartphones in everyone's pockets, online schedules such as Google Calendar now pervade our lives. Oops.

I was a judge at the first Imagine Cup finals in Barcelona in 2003. The team from Singapore created a supermarket cart with a built-in scanner so customers could tally up their groceries as they shopped. Their handling of discount coupons almost sent me through the ceiling. The team expected users to carry a Pocket PC, with which they would scan a bar code on a poster. The Pocket PC would then contact the vendor's Web site, display more information about the product and provide a discount coupon. I whacked them for it in the judging, and also in my book, "Why Software Sucks" (Addison-Wesley Professional, 2006). I thought, "Who, but a pitiful geek like you, is going to carry a bar code scanner in his pocket?" But now that everyone carries a smartphone with a camera, QR codes appear on almost everything, even tombstones (bit.ly/1vij0Ls). Double oops.

My biggest public blunder ever has to be my prediction that the 2007 launch of the Apple iPhone would crash in flames. I thought the cellular bandwidth was too low, the price too high, the carrier (AT&T) too lame and the apps too stupid (see my December 2010 column at msdn.microsoft.com/magazine/gg490348). And I thought that users would reject a touchscreen in place of physical buttons. I used deliberately inflammatory language to get attention, saying that the iPhone would be "the biggest flop since 'Ishtar' and 'Waterworld' combined." With typical restraint, I wrote that there was no way the iPhone could meet its expectations, because "God Himself could not build a phone that would live up to all the hype that the iPhone has gotten." (That got me some flak from religiously minded readers.)

> Hey, as my book agent always says, "There's no such thing as bad publicity. Just make sure they spell your name right."

We all know how that prognostication turned out. The iPhone soared (total sales topped 500 million last spring), smartphones became ubiquitous (with fascinating changes to society, see my February 2012 column at msdn.microsoft.com/magazine/hh781031), and I got ripped up one side of the Internet and down the other. Hey, as my book agent always says, "There's no such thing as bad publicity. Just make sure they spell your name right."

On the five-year anniversary of the iPhone's launch, a reporter called me up for his story on pundits who foolishly predicted its failure (see bit.ly/1q1g3vu). "What do you have to say for yourself?" he asked me. Having had four-and-a-half years to consider that question, I replied with the words, which if they ever name a university after me (prediction: don't hold your breath), will be inscribed in stone above the entrance: "Saepe fallitur, numquam in dubium."

"Often mistaken, never in doubt." ■

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.*

All New Releases!

.NET Controls for the Professional Developer

ComponentOne delivers Data Visualization, UI, Spreadsheet and Reporting Controls for Microsoft Visual Studio Development

Reporting

ActiveReports

Easily create a variety of form-based, analytical and transaction reports. Visualize data with graphical components, design reports in the IDE & extend the reach of your reports with versatile viewers.

Data Visualization

Studio Enterprise

Hundreds of data and UI controls for all .NET platforms including grids, charts, reports and schedulers. Make building next-gen UIs in today's apps easier with an array of samples with source code.

Spreadsheet

Spread Studio

A Microsoft Excel-like spreadsheet control with the functionally of an advanced data grid, for adding power to any .NET application. Includes a formula engine and flexible tabular layout.

Download your free trials at
ComponentOne.com

Visual Studio Partner

ComponentOne
a division of GrapeCity®

© 2014 GrapeCity, inc. All rights reserved. All other product and brand names are trademarks and/or registered trademarks of their respective holders.

Deploy predictive analytics solutions within .NET applications.

- Model using R®, SAS®, and SPSS® and deploy using .NET.
- Deploy using rich client, web, services, and Hadoop.
- No runtime dependency on the modeling environment.
- Absolutely no deployment fees.

syncfusion.com/predictiveanalytics

Contact us to learn more:
1-888-9-DOTNET
sales@syncfusion.com

Syncfusion®