THE MICROSOFT JOURNAL FOR DEVELOPERS

MARCH 2015 VOL 30 NO 3

# msdn magazine

Working with the .NET Framework 2015....18, 26, 36

When Only the Best Will Do

Our high-performance and feature-complete UI Controls and Libraries will help you build your best, without limits or compromise

DevExpress®

Free 30-day Trial
devexpress.com/try

Unleash the UI Superhero in You

With DevExpress tools, you'll deliver amazing user-experiences for Windows®, the Web and Your Mobile World

Experience  the DevExpress difference today.
Download your free 30-day trial.
**devexpress.com/try**

DevExpress®   WIN   ASP   MVC   WPF   HTML

THE MICROSOFT JOURNAL FOR DEVELOPERS

MARCH 2015 VOL 30 NO 3

# msdn magazine

Working with the .NET Framework 2015....18, 26, 36

Microsoft

The preferred choice for Microsoft developers and consultants.

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

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

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

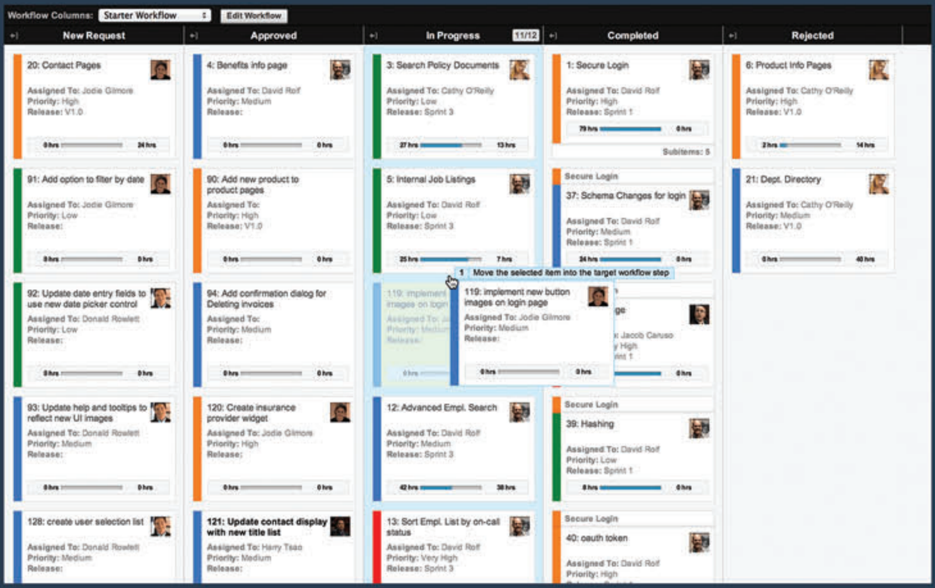
**try us today for free** at **axosoft.com/MSDN**

popular integrations

salesforce slack Visual Studio zendesk **and more**

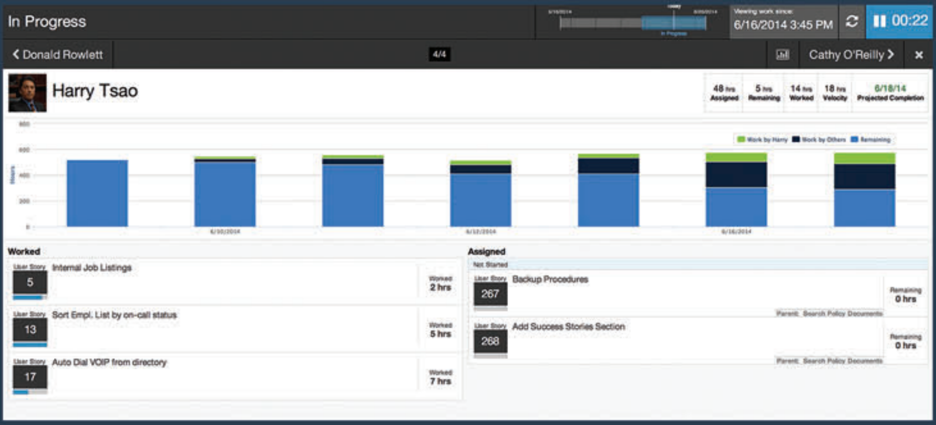
axosoft
800.653.0024

**take back your development time**


Workflow Columns: Starter Workflow | Edit Workflow

| New Request | Approved | In Progress 11/12 | Completed | Rejected |
|---|---|---|---|---|
| 20: Contact Pages. Assigned To: Jodie Gilmore. Priority: High. Release: V1.0. 0 hrs / 24 hrs | 4: Benefits info page. Assigned To: David Rolf. Priority: Medium. Release:. 0 hrs / 0 hrs | 3: Search Policy Documents. Assigned To: Cathy O'Reilly. Priority: Low. Release: Sprint 3. 27 hrs / 13 hrs | 1: Secure Login. Assigned To: David Rolf. Priority: High. Release: Sprint 1. 79 hrs / 0 hrs. Subitems: 5 | 6: Product Info Pages. Assigned To: Cathy O'Reilly. Priority: High. Release: V1.0. 2 hrs / 14 hrs |
| 91: Add option to filter by date. Assigned To: Jodie Gilmore. Priority: Low. Release:. 0 hrs / 0 hrs | 90: Add new product to product pages. Assigned To:. Priority: High. Release: V1.0. 0 hrs / 0 hrs | 5: Internal Job Listings. Assigned To: David Rolf. Priority: Low. Release: Sprint 3. 25 hrs / 7 hrs | Secure Login. 37: Schema Changes for login. Assigned To: David Rolf. Priority: Medium. Release: Sprint 1. 34 hrs / 0 hrs | 21: Dept. Directory. Assigned To: Cathy O'Reilly. Priority: Medium. Release: V1.0. 0 hrs / 40 hrs |
| 92: Update date entry fields to use new date picker control. Assigned To: Donald Rowlett. Priority: Low. Release:. 0 hrs / 0 hrs | 94: Add confirmation dialog for Deleting invoices. Assigned To:. Priority: Medium. Release:. 0 hrs / 0 hrs | 1 Move the selected item into the target workflow step. 119: implement new button images on login page. Assigned To: Jodie Gilmore. Priority: Medium. Release:. 0 hrs / 0 hrs | Assigned To: Jacob Caruso. Priority: High. Release: Sprint 1. 0 hrs | |
| 93: Update help and tooltips to reflect new UI images. Assigned To: Donald Rowlett. Priority: Medium. Release:. 0 hrs / 0 hrs | 120: Create insurance provider widget. Assigned To: Jodie Gilmore. Priority: High. Release:. 0 hrs / 0 hrs | 12: Advanced Empl. Search. Assigned To: David Rolf. Priority: Medium. Release: Sprint 3. 42 hrs / 38 hrs | Secure Login. 39: Hashing. Assigned To: David Rolf. Priority: Low. Release: Sprint 1. 8 hrs / 0 hrs | |
| 128: create user selection list. Assigned To: Donald Rowlett. Priority: Medium. Release: | 121: Update contact display with new title list. Assigned To: Harry Tsao. Priority: Medium. Release: | 13: Sort Empl. List by on-call status. Assigned To: David Rolf. Priority: Very High. Release: Sprint 3 | Secure Login. 40: oauth token. Assigned To: David Rolf. Priority: High | |

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


In Progress — Viewing work since: 6/16/2014 3:45 PM — 00:22

‹ Donald Rowlett — 4/4 — Cathy O'Reilly ›

Harry Tsao — 48 hrs Assigned · 5 hrs Remaining · 14 hrs Worked · 18 hrs Velocity · 6/18/14 Projected Completion

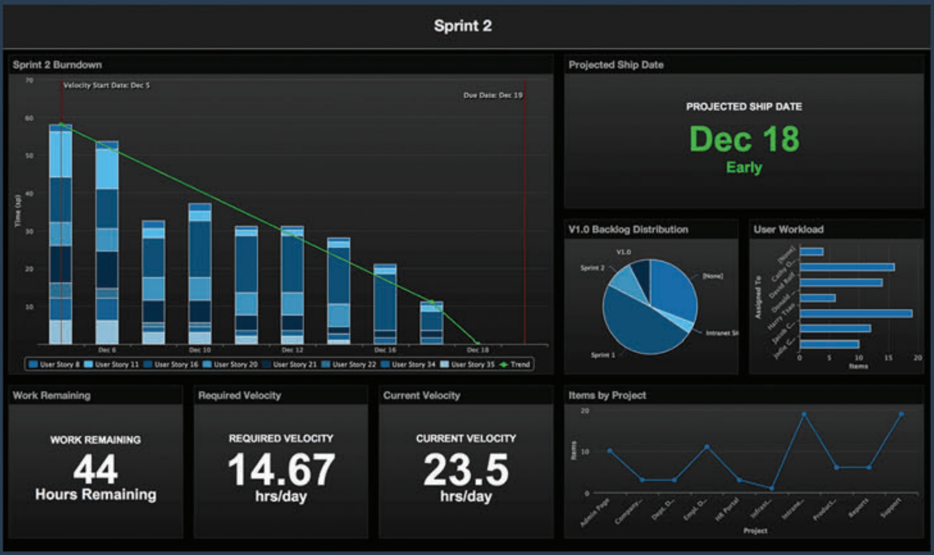
**Worked**

| | | |
|---|---|---|
| User Story 5 | Internal Job Listings | Worked 2 hrs |
| User Story 13 | Sort Empl. List by on-call status | Worked 5 hrs |
| User Story 17 | Auto Dial VOIP from directory | Worked 7 hrs |

**Assigned** — Not Started

| | | | |
|---|---|---|---|
| User Story 267 | Backup Procedures | Parent: Search Policy Documents | Remaining 0 hrs |
| User Story 268 | Add Success Stories Section | Parent: Search Policy Documents | Remaining 0 hrs |

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


Sprint 2

Automated burndown charts, velocity, projected ship dates and more

# msdn magazine

**MARCH 2015** VOLUME 30 NUMBER 3

**Director** Keith Boyd
**Editorial Director** Mohammad Al-Sabt *mmeditor@microsoft.com*
**Site Manager** Kent Sharkey
**Editorial Director, Enterprise Computing Group** Scott Bekker
**Editor in Chief** Michael Desmond
**Features Editor** Lafe Low
**Features Editor** Sharon Terdeman
**Group Managing Editor** Wendy Hernandez
**Senior Contributing Editor** Dr. James McCaffrey
**Contributing Editors** Rachel Appel, Dino Esposito, Kenny Kerr, Julie Lerman, Ted Neward, David S. Platt, Bruno Terkaly, Ricardo Villalobos
**Vice President, Art and Brand Design** Scott Shultz
**Art Director** Joshua Gould

ENTERPRISE COMPUTINGGROUP

**President**
*Henry Allain*

**Chief Revenue Officer**
*Dan LaBianca*

**Chief Marketing Officer**
*Carmel McDonagh*

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

**PRODUCTION STAFF**
*Director, Print Production* **David Seymour**
*Print Production Coordinator* **Anna Lyn Bayaua**

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

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

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

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

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

1105 MEDIA INC

**Chief Executive Officer**
Rajeev Kapur

**Chief Operating Officer**
Henry Allain

**Senior Vice President & Chief Financial Officer**
Richard Vitale

**Executive Vice President**
Michael J. Valenti

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

**Chairman of the Board**
Jeffrey S. Klein

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

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

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

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

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

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

Microsoft

BPA WORLDWIDE

1906 100 2006 AMERICAN BUSINESS MEDIA A Century Moving Business Forward

WPA Western Publications Association

## MADE FOR DEVELOPERS, BY DEVELOPERS

**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.**

- Improved OCR accuracy across all languages including East Asian
- New SVG engine for faster streaming, 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
- Use TWAIN from any browser
- 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 up to 8192 x 4096

.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**

MICHAEL DESMOND

# Not Your Father's .NET Framework

Back in November at the Connect(); event in New York City, Microsoft unveiled a raft of development tools, frameworks and technologies that refined the direction of the development ecosystem. Visual Studio 2015, the Microsoft .NET Framework 4.6, the new .NET Core, and updated frameworks such as ASP.NET 5 marked a move away from monolithic frameworks and libraries, and toward a more componentized, open and cross-platform dev environment.

In our recent Visual Studio 2015 and Microsoft Azure special issue (msdn.microsoft.com/magazine/dn879346) we explored many of the innovations announced at Connect();. Among those was a feature article by Microsoft Senior Program Manager Daniel Roth, "Introducing the ASP.NET 5 Preview" (msdn.microsoft.com/magazine/dn879354), which dove into the latest version of the Microsoft Web application framework and its relationship with the .NET Framework and the new .NET Core. As Roth explained in the article, ASP.NET 5 has been "rebuilt from the ground up" to make it a more flexible, cloud-ready, cross-platform-savvy development platform for Web applications.

This month, Roth returns with the second part of his ASP.NET 5 exploration, fittingly titled "A Deep Dive into the ASP.NET 5 Runtime." I caught up with him to ask about ASP.NET 5 and the impact that changes to the .NET Framework are having on Microsoft development.

"ASP.NET 5 has literally been rebuilt at every layer of the stack, from the tools in Visual Studio all the way down to the CLR itself," Roth explains. "It runs on the new .NET Core, has a new cross-platform runtime environment and completely replaces System.Web.dll with a new lightweight request pipeline."

.NET Core 5 is a refactored version of the .NET Framework, which can be tailored to the specific behaviors of an application and packaged as a private version of the framework for that application. Developers choose to employ as much or as little of the stack as needed. And because each app only calls to its companion .NET Core, developers can run multiple ASP.NET 5 applications on a single server, each unaffected by the other. Roth says Microsoft is busy building a .NET Core CLR for Windows, Mac and Linux.

One new feature of ASP.NET 5 that Roth singles out for attention is Tag Helpers for ASP.NET MVC 6, which lets developers express server-side rendering logic as custom HTML tags.

"They enable you to define your own domain-specific language in HTML. Tag Helpers work seamlessly with the HTML editor and Razor IntelliSense giving you a much more natural way to create views," Roth says. "We are providing a bunch of Tag Helpers out of the box and it's easy to create your own. I think Web developers are really going to love this feature."

"ASP.NET 5 has literally been rebuilt at every layer of the stack, from the tools in Visual Studio all the way down to the CLR itself."

*Daniel Roth, Senior Program Manager for ASP.NET, Microsoft*

Not that Microsoft is ignoring the .NET Framework, which now iterates to version 4.6. ASP.NET Web Forms applications get a welcome boost with this update, adding support for HTTP 2.0 and a "Roslyn" CodeDOM provider.

Beyond new features and capabilities, ASP.NET and its foundational underpinnings in .NET are undergoing a major transformation, as Microsoft takes these software platforms open source. The goal is to reshape both Visual Studio and the .NET Framework to reflect the evolving demands of modern software development.

"This requires us to completely rethink how we ship frameworks and tools so that we can be more agile and get features into the hands of developers faster," Roth says. "By making ASP.NET 5 and .NET Core open source and available completely via NuGet, getting new features is just a NuGet-package update away."

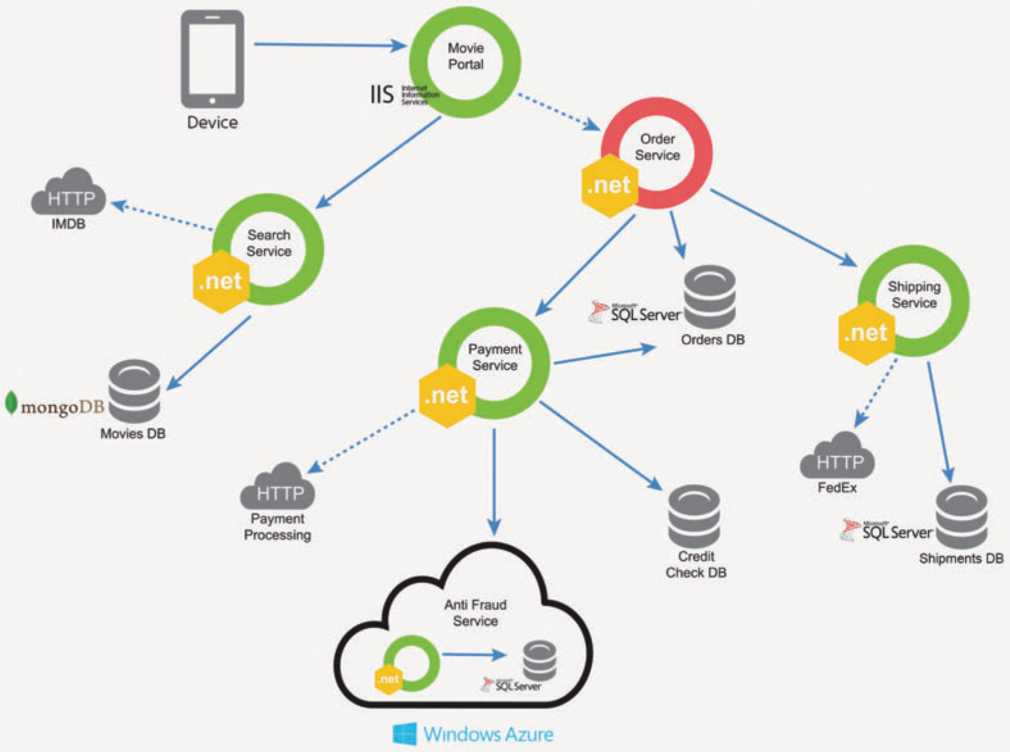
[illegible]

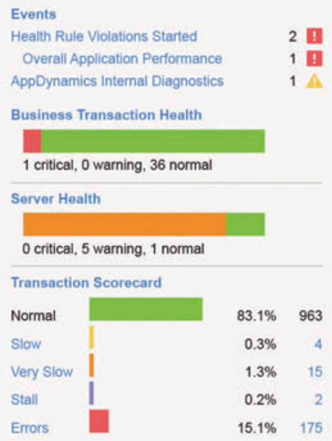
Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

APPDYNAMICS

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

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





**Events**

| | |
|---|---|
| Health Rule Violations Started | 2 ❗ |
| Overall Application Performance | 1 ❗ |
| AppDynamics Internal Diagnostics | 1 ⚠ |

**Business Transaction Health**

1 critical, 0 warning, 36 normal

**Server Health**

0 critical, 5 warning, 1 normal

**Transaction Scorecard**

| | | |
|---|---|---|
| Normal | 83.1% | 963 |
| Slow | 0.3% | 4 |
| Very Slow | 1.3% | 15 |
| Stall | 0.2% | 2 |
| Errors | 15.1% | 175 |

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

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

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

appdynamics.com

# Type Ahead

Since the early days of the Web, most pages feature a search box to help you quickly find content within the page itself or on the site. A well-done search function is a must in larger sites. This helps users find what they want quickly and easily, bypassing the site map and architecture.

In a shopping site, for example, you might want to use query strings to look up products, offers, or news and alerts. In a site built for something such as a professional sports team, the search function must be able to dig out news, results, athlete names, bios and so forth. The data structure upon which a search function has to work is never obvious. It's clearly application-specific.

Instead of reinventing the wheel every time, consider using an ad hoc full-text search engine such as Lucene.Net to back up your search function. An engine like Lucene.Net indexes a bunch of string-based documents and parses any given query string against the index. In doing so, the engine lets you specify complex combinations of query strings. For many pages and sites, Lucene.Net might be overkill, yet they still need some type of search that's smarter than placing an endless list of items in a dropdown.

There are many scenarios in which you might consider using an auto-completed input field.

This article will present a small framework for auto-completion built around Twitter typeahead.js. This framework isn't magical, but it truly simplifies using auto-completion in Web pages. The most enticing aspect of this framework is it lets you combine multiple datasets to query within the same page and retrieve distinct but related information.

## Setting Up Typeahead.js

For this article, I'll clarify the basics of using typeahead in a realistic scenario using the version of typeahead you find on NuGet when typing "typeahead," as shown in **Figure 1**. When you Google search typeahead, you might run into older references, older JavaScript files or just forked versions of the original project code. Documentation is misleading, as well.

The bundle file contains all packages that make up the library, including the Bloodhound engine to manage hints on the local browser. To set up a Web page or Razor view to use typeahead.js, you just need the familiar jQuery-like syntax and activate the plug-in on the selected input field. Here's an example:

```
<form action="@Url.Action("Query", "Home")" method="post">
  <input type="hidden" id="queryCode" name="queryCode" />
    <input type="text" name="queryString" id="queryString">
    <button id="queryButton" type="submit">Get</button>
</form>
```

It's important to note that in order to use auto-completion on a Web page for something useful, you also need a hidden buddy field to collect some sort of unique ID for the selected hint. There are many scenarios in which you may consider using an auto-completed input field. The scenario I'm interested in uses auto-completion to replace an otherwise endless dropdown list. It enables lightweight, Bing-style search in your site, without requiring the help of full-text engines such as Lucene.Net.

## Script Code to Have in the View

To use typeahead.js, reference jQuery 1.9.1 or superior and the typeahead script. The minimal amount of code you you'll need in the view is shown here:

```
$('#queryString').typeahead(
  null,
  {
    displayKey: 'value',
    source: hints.ttAdapter()
  }
});
```

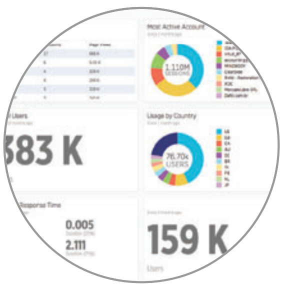
In doing so, you take all default settings and instruct the engine to use the value property in returned data for filling the dropdown list. A property named value is assumed to exist in any data you filter. In theory, you can set up auto-completion on any JavaScript data array. In practice, though, auto-completion makes sense mostly when data is downloaded from a remote source.

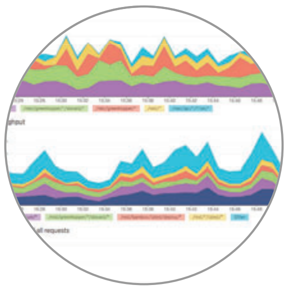
Downloading from a remote source poses many issues—same-origin browser policy, prefetching and caching—to name a few. Twitter typeahead.js comes with a suggestion engine named Bloodhound. This transparently does most of this work for you. If you get the bundle JavaScript file from NuGet, you can just start calling into Bloodhound without worrying about its download and setup. The hints variable in the previous code snippet results from the following and rather standard initialization of Bloodhound:

```
var hints = new Bloodhound({
  datumTokenizer: Bloodhound.tokenizers.obj.whitespace('value'),
  queryTokenizer: Bloodhound.tokenizers.whitespace,
  remote: "/hint/s?query=%QUERY"
});
hints.initialize();
```

# One source of truth

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





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

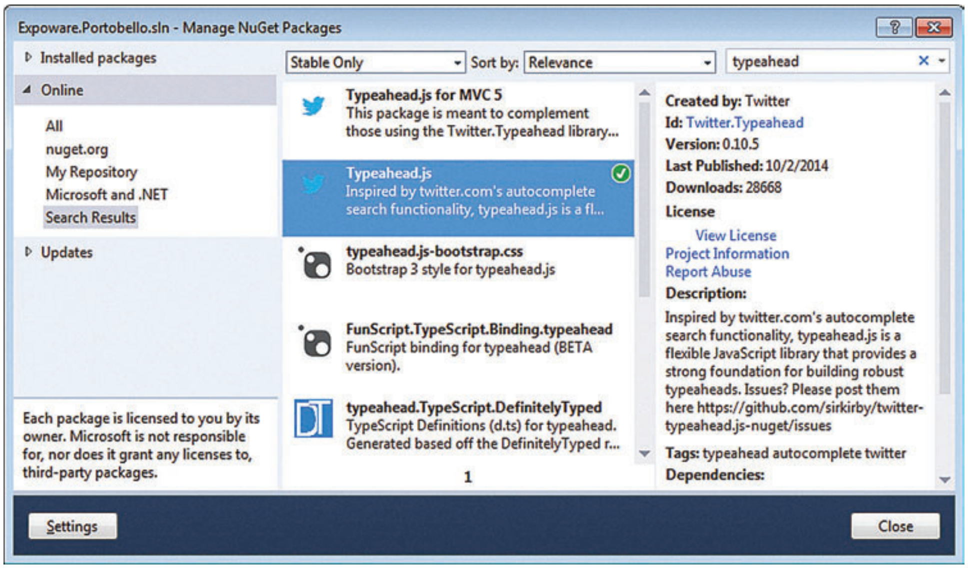
**Front-end Developers**
Deep insights into your browser-side app's engine.

**IT Operations**
Faster delivery. Fewer bottlenecks. More stability.

**App Owners**
Track engagement. Pinpoint issues. Optimize usability.

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

New Relic®



Expoware.Portobello.sln - Manage NuGet Packages

Installed packages
Online
All
nuget.org
My Repository
Microsoft and .NET
Search Results
Updates

Stable Only | Sort by: Relevance | typeahead

Typeahead.js for MVC 5
This package is meant to complement those using the Twitter.Typeahead library...

Typeahead.js
Inspired by twitter.com's autocomplete search functionality, typeahead.js is a fl...

typeahead.js-bootstrap.css
Bootstrap 3 style for typeahead.js

FunScript.TypeScript.Binding.typeahead
FunScript binding for typeahead (BETA version).

typeahead.TypeScript.DefinitelyTyped
TypeScript Definitions (d.ts) for typeahead. Generated based off the DefinitelyTyped r...

1

Created by: Twitter
Id: Twitter.Typeahead
Version: 0.10.5
Last Published: 10/2/2014
Downloads: 28668
License
View License
Project Information
Report Abuse
Description:
Inspired by twitter.com's autocomplete search functionality, typeahead.js is a flexible JavaScript library that provides a strong foundation for building robust typeaheads. Issues? Please post them here https://github.com/sirkirby/twitter-typeahead.js-nuget/issues
Tags: typeahead autocomplete twitter
Dependencies:

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Settings | Close

Figure 1 **The NuGet Package for Twitter Typeahead.js**

Note the remote attribute. That's just the server endpoint responsible for returning hints to display in the dropdown list. Also note the syntax %QUERY. This indicates the string in the input field being sent to the server for hints. In other words, %QUERY is a placeholder for whatever text is in the input field. By default, typeahead.js starts getting hints as soon as you type a single character. If you want to wait for characters to be in the buffer before auto-completion starts, then add a settings object as the first argument of the plug-in:

```
$('#queryString').typeahead(
  {
    minLength: 2
  },
  {
    displayKey: 'value',
    source: hints.ttAdapter()
  }
});
```

Any sufficiently complex plug-in needs a bit of CSS to look nice. Typeahead.js is no exception.

When the buffer is full enough to start remote calls, Bloodhound begins to work. It downloads JSON data and adapts it for display. At this point, you have a barely working auto-completion engine that pops up suggestions based on some logic you have on the server. However, there's a lot more to do before you can use auto-completion effectively in a real page.

## Use Typeahead.js with Bootstrap

Any sufficiently complex plug-in needs a bit of CSS to look nice. Typeahead.js is no exception. The plug-in comes with its own default UI, but you might want to apply some fixes, especially if you use it with Twitter Bootstrap. You might also want to customize some visual attributes, such as colors and padding. **Figure 2** lists some CSS classes you might want to use to personalize the look-and-feel of the typeahead.js component.

**Figure 3** gives an idea of what you can get out of custom CSS classes. You can also customize the overall plug-in behavior from a functional standpoint.

## Add Client-Side Logic

An auto-complete field is faster than any long dropdown. When the number of items from which to choose is in the hundreds, though, any classic dropdown list is slow. Therefore, if you plan to use the auto-complete input field to select a specific value—say, the name of a product or a customer—then a plain typeahead.js plug-in isn't enough. Some additional script code is required that binds to the selected event of the plug-in:

```
$('#queryString').on('typeahead:selected', function (e, datum) {
  $("#queryCode").val(datum.id);
});
```

The major benefit of auto-completion input is users type some intelligible name and the system retrieves the affiliated unique code or ID. This feature has to be explicitly coded. In the selected event handler, you retrieve the ID information from the datum object and store it safely in a hidden field. When the form to which the auto-completion input fields belong is posted, the selected ID is posted, as well. The format of the datum object—the data item being selected from the dropdown—depends on the format of the data you receive from the server side.

What about the text being displayed in the input field? In this scenario, you probably don't need to have any significant text displayed in the input field. The relevant input for further operations is what you store in the hidden field. What you display there is up to you. Just notice if you specify in the plug-in settings a displayKey property, because then the value of that property is automatically displayed in the input field. At any rate, you can set any value programmatically:

```
$("#queryString").val(datum.label);
```

In some cases, the auto-completion textbox is the only element in HTML form. This means you might want to process any selected

Figure 2 **The CSS Classes to Edit in Order to Customize the Typeahead.js Component**

| CSS Class | Description |
|---|---|
| twitter-typeahead | Styles the input field where the user types hints. |
| tt-hint | Styles the text that represents the delta between what you typed and the first hint. This class is only used when the hint property is set to true (false by default). |
| tt-dropdown-menu | Styles the dropdown popup where hints are listed. |
| tt-cursor | Styles highlighted suggestions in the dropdown box. |
| tt-highlight | Styles the portion of the text that matches the query string. |

30 DAYS
FREE TRIAL

Save Time
Working with Files?

CREATE CONVERT COMBINE EDIT PRINT

**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!

Risk Free - 30 Day Trial

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

data as soon as it's selected. By adding the following line to the typeahead.js selected event handler, you simulate a click on the submit button of the form:

```
$("#queryButton").click();
```

Suppose a user starts typing in the input field, causes the drop-down to display and then stops typing without making a selection. What should you do when he resumes typing? Naturally, you would let him type again until he makes a selection. Once the user makes a selection, some code has been stored so when editing resumes that selection must be canceled. You need a local variable to achieve this:

```
var typeaheadItemSelected = false;
$('#queryString').on('typeahead:selected', function (e, datum) {
  $("#queryCode").val(datum.id);
  typeaheadItemSelected = true;
});
```

You also need a handler for the focus event of the input field to reset any stored data:

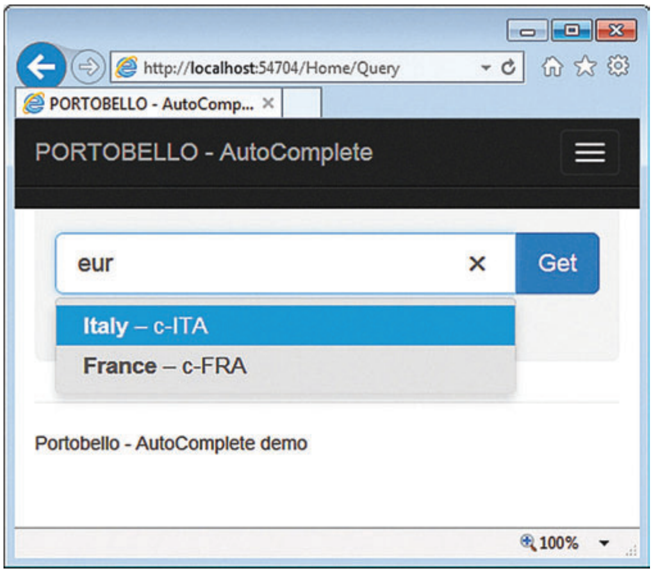
```
$('#queryString').on('input', function () {
  if (typeaheadItemSelected) {
    typeaheadItemSelected = false;
    $('#queryString').val('');
    $("#queryCode").val('');
  }
});
```

The ultimate purpose of this extra client-side logic is to ensure the auto-completion input field works the same as a dropdown list.

## The Server Side of Auto-Complete

Whatever any code on the client side can do strictly depends on the data returned from the server. At the very minimum, the server endpoint is just a URL that returns JSON data. Once you have an endpoint that serves a collection of Product objects, for example, you can use the displayKey property of typeahead.js to perform some sort of data binding on the dropdown list of hints. In its simplest form, a JSON-returning controller method can look like this:

```
public JsonResult P(string query)
{
  var productHints = _service.GetMatchingProducts(query);
  return Json(productHints, JsonRequestBehavior.AllowGet);
}
```



http://localhost:54704/Home/Query
PORTOBELLO - AutoComp...
PORTOBELLO - AutoComplete
eur
Get
Italy – c-ITA
France – c-FRA
Portobello - AutoComplete demo
100%

Figure 3 **Customized CSS Classes Can Accomplish Different Effects for Your App**

If the auto-completion input field is expected to display hints for homogeneous data items, this is an ideal approach. On the client side, in fact, you can easily leverage the templating mechanism built into typeahead.js and arrange custom suggestion views:

```
$('#queryString').typeahead(
null,
{
  templates: {
    suggestion: Handlebars.compile('<b>({{Id}}</b>: {{notes}}')
  },
  source: hints.ttAdapter()
});
```

The templates property replaces displayKey and sets a custom layout for the dropdown content. The dropdown in **Figure 3** results from the previous code snippet. When arranging a template, you might want to use an ad hoc template engine such as Handlebars (handlebarsjs.com). You must link Handlebars to the project separately from typeahead.js. Using Handlebars is optional. You can always format an HTML template through manual JavaScript code or even return a server-side object with preformatted HTML.

Things get trickier when your auto-completion input is expected to return heterogeneous hints, such as products or offers. In this case, you must return an array of some intermediate data type that contains enough information for the user to choose. The auto-completion you get with this article offers a basic AutoCompleteItem class, as shown here:

```
public class AutoCompleteItem
{
  public String label { get; set; }
  public String id { get; set; }
  public String value { get; set; }
}
```

The id property contains a unique ID. When posted, that is meaningful to the receiving controller. It's typically made of two pieces—the actual ID and an identifier that uniquely matches the ID to one of the datasets (products or offers) possibly being returned in the query. The property value is the string content to show in the dropdown list. The other property is sort of a cargo property for whatever else you might need. The value property can also contain server-side-arranged HTML strings for custom suggestion layouts. The server-side code is also responsible for running all required queries and massaging data into a collection of AutoCompleteItem objects.

## Wrapping Up

Usability is more important every day in modern Web sites. It's welcome in the public face of the site, but is even more critical in Web site back ends where the actual information is inserted. On the admin side of one of my Web sites, I once had a dropdown list with more than 700 items. It worked, but it was slow. I replaced it with auto-completion and now it's blazingly fast. I arranged a GitHub project for this and other utilities at bit.ly/1zubJea. ■

**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*

30 days free trial

GROUPDOCS

Document Collaboration API's
to allow you to View, Annotate, Convert,
Sign, Compare and Assemble over
50 document file types.

Viewer Annotation Conversion Signature Comparison Assembly

.NET Libraries Java Libraries Cloud APIs

Sales Inquiries:

+1 214 329 9760 sales@groupdocs.com groupdocs.com

# Using Printf with Modern C++

What would it take to modernize printf? That might seem like an odd question to many developers who believe that C++ already provides a modern replacement for printf. While the claim to fame of the C++ Standard Library is undoubtedly the excellent Standard Template Library (STL), it also includes a stream-based input/output library that bears no resemblance to STL and embodies none of its principles related to efficiency.

"Generic programming is an approach to programming that focuses on design algorithms and data structures so that they work in the most general setting without loss of efficiency," according to Alexander Stepanov and Daniel Rose, in the book, "From Mathematics to Generic Programming" (Addison-Wesley Professional, 2015).

To be honest, neither printf nor cout is in any way representative of modern C++. The printf function is an example of a variadic function and one of the few good uses of this somewhat brittle feature inherited from the C programming language. Variadic functions predate variadic templates. The latter offer a truly modern and robust facility for dealing with a variable number of types or arguments. In contrast, cout doesn't employ variadic anything, but instead relies so heavily on virtual function calls that the compiler isn't able to do much to optimize its performance. Indeed, the evolution of CPU design has favored printf while doing little to improve the performance of the polymorphic approach of cout. Therefore, if you want performance and efficiency, printf is a better choice. It also produces code that's more concise. Here's an example:

```
#include <stdio.h>

int main()
{
  printf("%f\n", 123.456);
}
```

The %f conversion specifier tells printf to expect a floating-point number and convert it to decimal notation. The \n is just an ordinary newline character that may be expanded to include a carriage return, depending on the destination. The floating-point conversion assumes a precision of 6, referring to the number of digits that will appear after the decimal point. Thus, this example will print the following characters, followed by a new line:

```
123.456000
```

Achieving the same end with cout seems relatively straightforward at first:

```
#include <iostream>

int main()
{
  std::cout << 123.456 << std::endl;
}
```

Here, cout relies on operator overloading to direct or send the floating-point number to the output stream. I don't like the abuse of operator overloading in this way, but I admit it's a matter of personal style. Finally, endl concludes by inserting a new line into the output stream. However, this isn't quite the same as the printf example and produces output with a different decimal precision:

```
123.456
```

This leads to an obvious question: How can I change the precision for the respective abstractions? Well, if I only want two digits following the decimal point, I can simply specify this as part of the printf float-point conversion specifier:

```
printf("%.2f\n", 123.456);
```

Now printf will round the number to produce the following result:

```
123.46
```

To get the same effect with cout requires a bit more typing:

```
#include <iomanip> // Needed for setprecision

std::cout << std::fixed << std::setprecision(2)
          << 123.456 << std::endl;
```

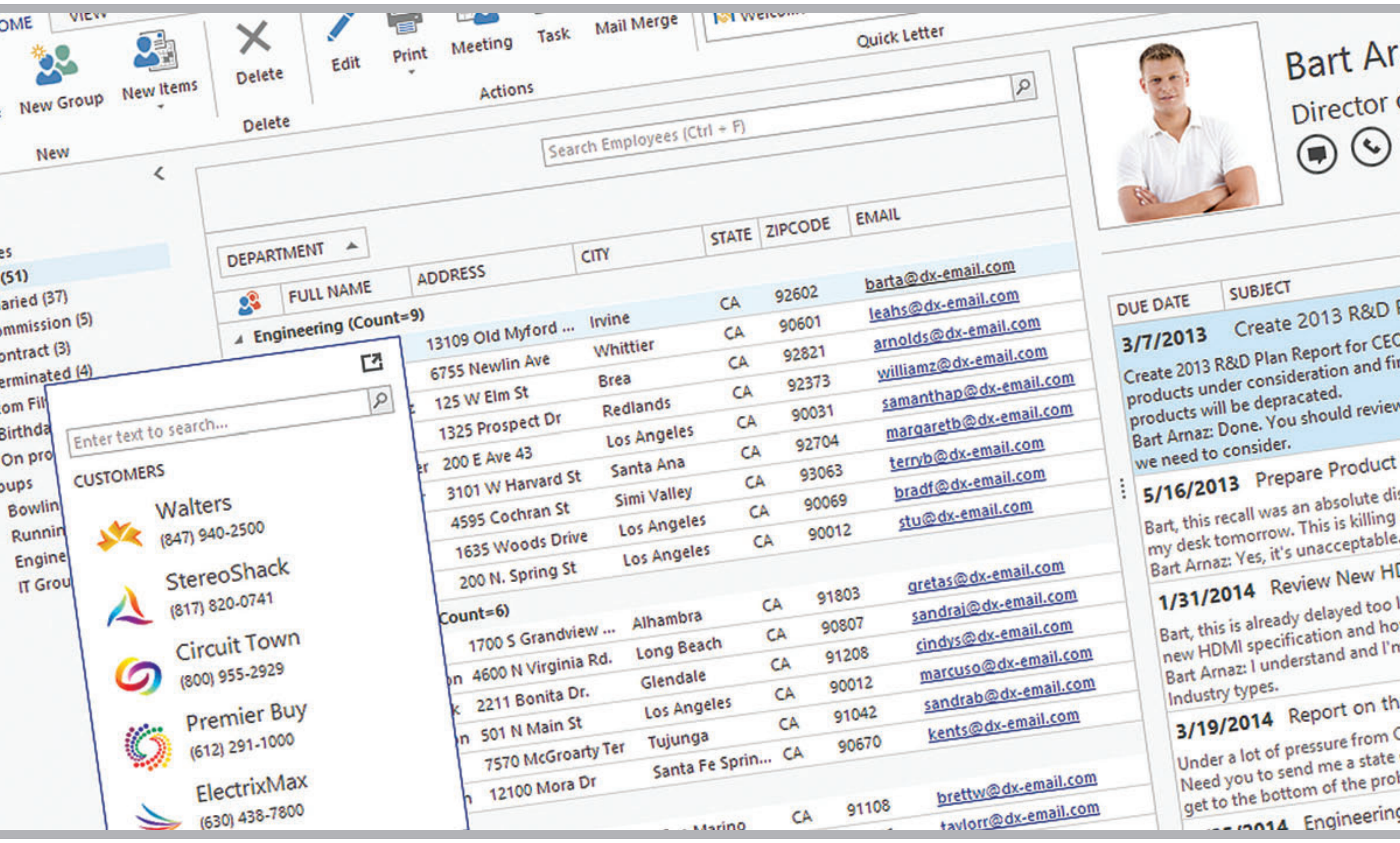
Even if you don't mind the verbosity of all of this and rather enjoy the flexibility or expressiveness, keep in mind that this abstraction comes at a cost. First, the fixed and setprecision manipulators are stateful, meaning their effect persists until they're reversed or reset. By contrast, the printf conversion specifier includes everything required for that single conversion, without affecting any other code. The other cost may not matter for most output, but the day might come when you notice that everyone else's programs can output many times faster than yours can. Apart from the overhead from virtual function calls, endl also gives you more than you might have bargained for. Not only does it send a new line to the output, but it also causes the underlying stream to flush its output. When writing any kind of I/O, whether to the console, a file on disk, a network connection, or even a graphics pipeline, flushing is usually very costly, and repeated flushes will undoubtedly hurt performance.

Now that I've explored and contrasted printf and cout a little, it's time to return to the original question: What would it take to modernize printf? Surely, with the advent of modern C++, as exemplified by C++11 and beyond, I can improve the productivity and reliability of printf without sacrificing performance. Another somewhat unrelated member of the C++ Standard Library is the language's official string class. Although this class has also been maligned over the years, it does offer excellent performance. While not without fault, it provides a very useful way to handle strings in C++. Therefore, any modernization of printf really ought to play

DevExpress®

# Your Next Great App Starts Here

With DevExpress tools, you'll create high-performance, high-impact .NET solutions that fully replicate the look, feel and user-experience of Microsoft Office®



es
(51)
aried (37)
mmission (5)
ontract (3)
erminated (4)
om Fil
Birthda
On pro
oups
Bowlin
Runnin
Engine
IT Grou

DEPARTMENT

Engineering (Count=9)

| FULL NAME | ADDRESS | CITY | STATE | ZIPCODE | EMAIL |
|---|---|---|---|---|---|
| | 13109 Old Myford ... | Irvine | CA | 92602 | barta@dx-email.com |
| | 6755 Newlin Ave | Whittier | CA | 90601 | leahs@dx-email.com |
| | 125 W Elm St | Brea | CA | 92821 | arnolds@dx-email.com |
| | 1325 Prospect Dr | Redlands | CA | 92373 | williamz@dx-email.com |
| | 200 E Ave 43 | Los Angeles | CA | 90031 | samanthap@dx-email.com |
| | 3101 W Harvard St | Santa Ana | CA | 92704 | margaretb@dx-email.com |
| | 4595 Cochran St | Simi Valley | CA | 93063 | terryb@dx-email.com |
| | 1635 Woods Drive | Los Angeles | CA | 90069 | bradf@dx-email.com |
| | 200 N. Spring St | Los Angeles | CA | 90012 | stu@dx-email.com |
| Count=6) | | | | | |
| | 1700 S Grandview ... | Alhambra | CA | 91803 | gretas@dx-email.com |
| | 4600 N Virginia Rd. | Long Beach | CA | 90807 | sandraj@dx-email.com |
| | 2211 Bonita Dr. | Glendale | CA | 91208 | cindys@dx-email.com |
| | 501 N Main St | Los Angeles | CA | 90012 | marcuso@dx-email.com |
| | 7570 McGroarty Ter | Tujunga | CA | 91042 | sandrab@dx-email.com |
| | 12100 Mora Dr | Santa Fe Sprin... | CA | 90670 | kents@dx-email.com |
| | | Marino | CA | 91108 | brettw@dx-email.com |
| | | | | | taylorr@dx-email.com |

Enter text to search...

CUSTOMERS

| | |
|---|---|
| Walters | (847) 940-2500 |
| StereoShack | (817) 820-0741 |
| Circuit Town | (800) 955-2929 |
| Premier Buy | (612) 291-1000 |
| ElectrixMax | (630) 438-7800 |

| DUE DATE | SUBJECT |
|---|---|
| 3/7/2013 | Create 2013 R&D |
| 5/16/2013 | Prepare Product |
| 1/31/2014 | Review New HD |
| 3/19/2014 | Report on th |

Create 2013 R&D Plan Report for CEO products under consideration and fi products will be depracated. Bart Arnaz: Done. You should review we need to consider.

Bart, this recall was an absolute di my desk tomorrow. This is killing Bart Arnaz: Yes, it's unacceptable

Bart, this is already delayed too l new HDMI specification and ho Bart Arnaz: I understand and I'n Industry types.

Under a lot of pressure from Need you to send me a state get to the bottom of the pro

Engineerin

## Unleash the UI Superhero in You

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

WIN ASP MVC WPF HTML 5

nicely with string and wstring. Let's see what can be done. First, let me address what I consider to be the most vexing problem of printf:

```
std::string value = "Hello";

printf("%s\n", value);
```

This really ought to work, but as I'm sure you can plainly see, instead it will result in what is lovingly known as "undefined behavior." As you know, printf is all about text and the C++ string class is the premier manifestation of text in the C++ language. What I need to do is wrap printf in such a way that this just works. I don't want to have to repeatedly pluck out the string's null-terminated character array as follows:

```
printf("%s\n", value.c_str());
```

This is just tedious, so I'm going to fix it by wrapping printf. Traditionally, this has involved writing another variadic function. Perhaps something like this:

```
void Print(char const * const format, ...)
{
  va_list args;
  va_start(args, format);
  vprintf(format, args);
  va_end(args);
}
```

Unfortunately, this gains me nothing. It might be useful to wrap some variant of printf in order to write to some other buffer, but in this case, I've gained nothing of value. I don't want to go back to C-style variadic functions. Instead, I want to look forward and embrace modern C++. Fortunately, thanks to the C++11 variadic templates, I'll never have to write another variadic function in my life. Rather than wrapping the printf function in another variadic function, I can instead wrap it in a variadic template:

```
template <typename ... Args>
void Print(char const * const format,
           Args const & ... args) noexcept
{
  printf(format, args ...);
}
```

At first, it might not seem that I've gained much. If I were to call the Print function like this:

```
Print("%d %d\n", 123, 456);
```

it would cause the args parameter pack, consisting of 123 and 456, to expand inside the body of the variadic template as if I had simply written this:

```
printf("%d %d\n", 123, 456);
```

So what have I gained? Sure, I'm calling printf rather than vprintf and I don't need to manage a va_list and the associated stack-twiddling macros, but I'm still merely forwarding arguments. Don't overlook the simplicity of this solution, however. Again, the compiler will unpack the function template's arguments as if I had simply called printf directly, which means there's no overhead in wrapping printf in this way. It also means this is still first-class C++ and I can employ the language's powerful metaprogramming techniques to inject any requisite code—and in a completely generic way. Rather than simply expanding the args parameter pack, I can wrap each argument to add any adjustment needed by printf. Consider this simple function template:

```
template <typename T>
T Argument(T value) noexcept
{
  return value;
}
```

It doesn't appear to do much and indeed it doesn't, but I can now expand the parameter pack to wrap each argument in one of these functions as follows:

```
template <typename ... Args>
void Print(char const * const format,
           Args const & ... args) noexcept
{
  printf(format, Argument(args) ...);
}
```

I can still call the Print function in the same way:

```
Print("%d %d\n", 123, 456);
```

But it now effectively produces the following expansion:

```
printf("%d %d\n", Argument(123), Argument(456));
```

This is very interesting. Sure, it makes no difference for these integer arguments, but I can now overload the Argument function to handle C++ string classes:

```
template <typename T>
T const * Argument(std::basic_string<T> const & value) noexcept
{
  return value.c_str();
}
```

Then I can simply call the Print function with some strings:

```
int main()
{
  std::string const hello = "Hello";
  std::wstring const world = L"World";

  Print("%d %s %ls\n", 123, hello, world);
}
```

The compiler will effectively expand the inner printf function as follows:

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

This ensures that each string's null-terminated character array is provided to printf and produces a completely well-defined behavior:

```
123 Hello World
```

Along with the Print function template, I also use a number of overloads for unformatted output. This tends to be safer and prevents printf from accidentally misinterpreting arbitrary strings as containing conversion specifiers. **Figure 1** lists these functions.

The first two overloads simply format ordinary and wide-character arrays, respectively. The final function template forwards to the appropriate overload, depending on whether a string or wstring is provided as an argument. Given these functions, I can safely print some conversion specifiers literally, as follows:

```
Print("%d %s %ls\n");
```

That takes care of my most common gripe with printf by handling string output safely and transparently. What about formatting strings themselves? The C++ Standard Library provides different

Figure 1 **Printing Unformatted Output**

```
inline void Print(char const * const value) noexcept
{
  Print("%s", value);
}

inline void Print(wchar_t const * const value) noexcept
{
  Print("%ls", value);
}

template <typename T>
void Print(std::basic_string<T> const & value) noexcept
{
  Print(value.c_str());
}
```

/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**

variants of printf for writing to character string buffers. Of these, I find snprintf and swprintf the most effective. These two functions handle character and wide-character output, respectively. They allow you to specify the maximum number of characters that may be written and return a value that can be used to calculate how much space is needed should the original buffer not be large enough. Still, on their own they're error-prone and quite tedious to use. Time for some modern C++.

The question is whether a string object can be resized faster than printf can parse its format string and calculate the required buffer size.

While C doesn't support function overloading, it's far more convenient to use overloading in C++ and this opens the door for generic programming, so I'll start by wrapping both snprintf and swprintf as functions called StringPrint. I'll also use variadic function templates so I can take advantage of the safe argument expansion I previously used for the Print function. **Figure 2** provides the code for both functions. These functions also assert that the result is not -1, which is what the underlying functions return when there's some recoverable problem parsing the format string. I use an assertion because I just assume this is a bug and should be fixed prior to shipping production code. You might want to replace this with an exception, but keep in mind there's no bulletproof way of turning all errors into exceptions as it's still possible to pass invalid arguments that will lead to undefined behavior. Modern C++ is not idiot-proof C++.

Figure 2 **Low-Level String Formatting Functions**

```
template <typename ... Args>
int StringPrint(char * const buffer,
                size_t const bufferCount,
                char const * const format,
                Args const & ... args) noexcept
{
  int const result = snprintf(buffer,
                              bufferCount,
                              format,
                              Argument(args) ...);

  ASSERT(-1 != result);
  return result;
}

template <typename ... Args>
int StringPrint(wchar_t * const buffer,
                size_t const bufferCount,
                wchar_t const * const format,
                Args const & ... args) noexcept
{
  int const result = swprintf(buffer,
                              bufferCount,
                              format,
                              Argument(args) ...);

  ASSERT(-1 != result);
  return result;
}
```

The StringPrint functions provide a generic way of dealing with string formatting. Now I can focus on the specifics of the string class, and this mostly involves memory management. I'd like to write code like this:

```
std::string result;

Format(result, "%d %s %ls", 123, hello, world);

ASSERT("123 Hello World" == result);
```

There's no visible buffer management. I don't have to figure out how large a buffer to allocate. I simply ask the Format function to logically assign the formatted output to the string object. As usual, Format can be a function template, specifically a variadic template:

```
template <typename T, typename ... Args>
void Format(std::basic_string<T> & buffer,
            T const * const format,
            Args const & ... args)
{
}
```

There are a variety of ways to implement this function. Some experimentation and a good dose of profiling go a long way. A simple but naïve approach is to assume the string is either empty or too small to contain the formatted output. In that case, I'd start by determining the required size with StringPrint, resize the buffer to match, and then call StringPrint again with the properly allocated buffer. Something like this:

```
size_t const size = StringPrint(nullptr, 0, format, args ...);
buffer.resize(size);
StringPrint(&buffer[0], buffer.size() + 1, format, args ...);
```

The + 1 is required because both snprintf and swprintf assume the reported buffer size includes space for the null terminator. This works well enough, but it should be obvious that I'm leaving performance on the table. A much faster approach in most cases is to assume the string is large enough to contain the formatted output and resize only if necessary. This almost reverses the preceding code but is quite safe. I begin by attempting to format the string directly into the buffer:

```
size_t const size = StringPrint(&buffer[0],
                                buffer.size() + 1,
                                format,
                                args ...);
```

If the string is empty to begin with or just not large enough, the resulting size will be greater than the size of the string and I'll know to resize the string before calling StringPrint again:

```
if (size > buffer.size())
{
  buffer.resize(size);
  StringPrint(&buffer[0], buffer.size() + 1, format, args ...);
}
```

If the resulting size is less than the size of the string, I'll know that the format succeeded, but the buffer needs to be trimmed to match:

```
else if (size < buffer.size())
{
  buffer.resize(size);
}
```

Finally, if the sizes match there's nothing to do and the Format function can simply return. The complete Format function template can be found in **Figure 3**. If you're familiar with the string class, you might recall that it also reports its capacity and you might be tempted to set the string's size to match its capacity prior to calling StringPrint the first time, thinking that this might improve your odds of formatting the string correctly the first time. The question is whether a string object can be resized faster than printf can parse its format string and calculate the required buffer size. Based on my informal tests, the answer is: it depends. You see, resizing a string to match its capacity

Figure 3 **Formatting Strings**

```
template <typename T, typename ... Args>
void Format(std::basic_string<T> & buffer,
            T const * const format,
            Args const & ... args)
{
  size_t const size = StringPrint(&buffer[0],
                                  buffer.size() + 1,
                                  format,
                                  args ...);

  if (size > buffer.size())
  {
    buffer.resize(size);
    StringPrint(&buffer[0], buffer.size() + 1, format, args ...);
  }
  else if (size < buffer.size())
  {
    buffer.resize(size);
  }
}
```

is more than simply changing the reported size. Any additional characters must be cleared and this takes time. Whether this takes more time than it takes printf to parse its format string depends on how many characters need to be cleared and how complex the formatting happens to be. I use an even faster algorithm for high-volume output, but I've found that the Format function in **Figure 3** provides good performance for most scenarios.

With this Format function in hand, it also becomes very easy to write various helper functions for common string-formatting operations. Perhaps you need to convert a wide-character string to an ordinary string:

```
inline std::string ToString(wchar_t const * value)
{
  std::string result;
  Format(result, "%ls", value);
  return result;
}

ASSERT("hello" == ToString(L"hello"));
```

Perhaps you need to format floating-point numbers:

```
inline std::string ToString(double const value,
                            unsigned const precision = 6)
{
  std::string result;
  Format(result, "%.*f", precision, value);
  return result;
}

ASSERT("123.46" == ToString(123.456, 2));
```

For the performance obsessed, such specialized conversion functions are also quite easy to optimize further because the required buffer sizes are somewhat predictable, but I'll leave that for you to explore on your own.

This is just a handful of useful functions from my modern C++ output library. I hope they've given you some inspiration for how to use modern C++ to update some old-school C and C++ programming techniques. By the way, my output library defines the Argument functions, as well as the low-level StringPrint functions in a nested Internal namespace. This tends to keep the library nice and simple to discover, but you can arrange your implementation however you wish. ■

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

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**

# A Deep Dive into the ASP.NET 5 Runtime

Daniel Roth

Last November Microsoft announced ASP.NET 5 as a new open source and cross-platform framework for building modern Web and cloud applications using the Microsoft .NET Framework. We (the ASP.NET development team, of which I'm a part) released ASP.NET 5 Preview along with Visual Studio 2015 Preview at the Connect(); event in New York City. I introduced you to the ASP.NET 5 runtime and its new Visual Studio project system in the "Introducing the ASP.NET 5 Preview" article from the special Dec. 15, 2014, Issue of *MSDN Magazine* (bit.ly/1K4PY4U). Since that last look at ASP.NET 5 in December there have been two more preview releases of ASP.NET 5: Beta2 with Visual Studio 2015 CTP 5, and Beta3 with Visual Studio 2015 CTP 6. With each release, the framework continues to evolve and improve. In many cases, these improvements were generously contributed by the .NET community through the public ASP.NET project on GitHub (bit.ly/1DaY7Cd). In this article, I'll take a deeper look under the hood of the new ASP.NET 5 runtime to see what's changed in the most recent release.

## The K Runtime Environment (KRE)

As you saw back in December, ASP.NET 5 is based on a flexible, cross-platform runtime host that can host one of several .NET CLRs. You can run ASP.NET 5 applications on the .NET Framework with its full API set for maximum compatibility. You can also run ASP.NET 5 on the new .NET Core, which enables true side-by-side deployments that you can copy into existing environments without having to change anything else on the machine. In the future, you'll also be able to run ASP.NET 5 cross-platform on .NET Core, and there's community support for running cross-platform on Mono today.

This article discusses a prerelease version of ASP.NET 5. All information is subject to change.

This article discusses:

- The K runtime environment
- Cross-platform SDK tools
- ASP.NET hosting
- Configuring services
- Request processing

Technologies discussed:

ASP.NET 5, Visual Studio 2015

The runtime hosting infrastructure for ASP.NET 5 is currently called the K Runtime Environment (KRE), which is a generic placeholder name until we finalize the official name. The KRE provides an environment that has everything a .NET app needs to run: a host process, CLR hosting logic, managed entry point discovery and so forth. The KRE was built for running cross-platform .NET Web applications, but it can run other types of .NET applications, too, such as console apps. The KRE is based on the same .NET CLR and base class libraries .NET developers have come to know and love, while enabling cross-platform support for running .NET applications on Windows, OS X and Linux.

Logically, the KRE has five layers of functionality. I'll describe each of these layers and their responsibilities.

**Layer 1. Native Process:** The native process is a very thin layer with the responsibility to find and call the native CLR host, passing along arguments given to the process to be used by the rest of the stack. On Windows this is a native executable (klr.exe); on Mac or Linux it's an executable bash script. Running on IIS is accomplished with either a new native HTTP module or by using the Helios loader. The Helios loader leverages extensibility hooks in the .NET Framework 4.5.1 to bootstrap the KRE without requiring installation of new IIS modules. With the native HTTP module you can run .NET Core-based Web apps on IIS without any dependency on the .NET Framework. While the native HTTP module is not yet publicly available, we do plan to make it available in a future preview.

**Layers 2 and 3. Native CLR Host and CLR:** The native CLR host has three main responsibilities:

1. Boot the CLR. How this is achieved depends on the version of the CLR. For example, to boot .NET Core involves loading coreclr.dll, configuring and starting the runtime, and creating the AppDomain that all managed code will run in. For Mono and the .NET Framework 4.5.1, the process varies somewhat, but the outcome is the same.

2. Call the managed entry point, which is the next layer.
3. When the entry point of the native host returns, this process will then clean up and shut down the CLR—that is, unload the app domain and stop the runtime.

**Layer 4: Managed Entry Point:** This layer is the first layer written in managed code. It's responsible for:

1. Loading assemblies and satisfying dependencies from the lib folder.
2. Setting up the IApplicationEnvironment and the core dependency injection infrastructure.
3. Calling the main entry point of the specified application or application host.

At this layer, assemblies are loaded only from the application base path or the specified lib folders. It's the next layer that adds additional loaders for resolving dependencies from NuGet packages or even code compiled at run time.

**Layer 5: Application Host/Application:** If a developer compiles an entire application to assemblies on disk in the lib folder, this layer is the application—the application the end user sees. If you want to do this, you can compile your application and pass the name of the DLL containing a standard Main entry point when launching layer 1.

However, in most scenarios, you'd use an application host to help resolve application dependencies and run your app. Microsoft.Framework.ApplicationHost is the application host provided in the KRE and its responsibilities include:

1. Walking the dependencies in project.json and building up the closure of dependencies the app will use. The dependency-walking logic is described in more detail at bit.ly/1y5lZEm.
2. Adding additional assembly loaders that can load assemblies from various sources, such as installed NuGet packages, sources compiled at run time using Roslyn and so on.
3. Calling the entry point of the assembly whose name was given as an argument when the native process was started. The assembly can be anything with an entry point that the ApplicationHost knows how to load. The ApplicationHost that comes with the KRE knows how to find a public void Main method. This is the entry point used to set up the ASP.NET hosting layer, which knows how to find Startup.cs and run the Configure method for your Web application.

One thing to note as you're learning about the KRE is that this is a low-level part of the stack. When operating at the KRE level, the world is still very much about finding and loading dynamic-link libraries (DLLs). It's the KRE that contains the logic to allow you, at the application level, to think only about packages and other top-level dependencies.

## Cross-Platform SDK Tools

The KRE is packaged with an SDK that contains everything you need to build cross-platform .NET applications. In my previous article on ASP.NET 5, I described how you can use the KRE Version Manager (KVM) tool to list the installed KREs on your machine, install new ones and select the KRE you want to use. You can find instructions for how to install the KVM for your OS at bit.ly/1y5mqyi.

The KVM installs KREs from a NuGet feed configured using the KRE_FEED environment variable. The KREs are not NuGet packages in the traditional sense in that they aren't packages on which you'd ever depend; NuGet is just a convenient way to distribute and version the KREs. By default, a KRE is installed by copying and extracting the KRE .zip file into %USERPROFILE%\.k\runtimes.

I also previously introduced the K Package Manager (KPM) tool for installing, restoring and creating NuGet packages. We plan to rename the KPM tool to nuget, and align it with the existing NuGet client. As part of this alignment, some of the kpm sub-commands have already been renamed. You now use the bundle command to bundle an application for publishing and the pack command for building and creating NuGet packages for a project. The updated build command produces raw build outputs without packaging anything. There's also a new wrap command that allows tooling to reference an existing csproj-based project in project.json. By default, packages are now installed in the %USERPROFILE%\.k\packages folder, but you can control this by setting a packages path in your global.json file.

## A Cross-Platform .NET Console App

I will now show you how to create a simple cross-platform .NET console app using the KRE. First, I need to create a DLL with the entry point and I can do that using the ASP.NET 5 Console Application project template in Visual Studio 2015. The code should look like this:

```
public class Program
{
  public void Main(string[] args)
  {
    Console.WriteLine("Hello World");
    Console.ReadLine();
  }
}
```

This looks pretty familiar, but notice that the entry point is actually an instance method. In addition to a static Program.Main entry point, the KRE supports instance-based entry points. You can even make the main entry point asynchronous and return a Task. By having the main entry point be an instance method, you can have services injected into your application by the runtime environment.

You can run this application from within Visual Studio or on the command line by running "k run" from the directory containing the project.json file for the application. The K command is really just a simple batch file and the "k run" command can be expanded into the following:

```
klr.exe --appbase . Microsoft.Framework.ApplicationHost run
```

Figure 1 **Injecting the IServiceManifest Service into the Application**

```
namespace ConsoleApp1
{
  public class Program
  {
    public Program(IServiceManifest serviceManifest)
    {
      ServiceManifest = serviceManifest;
    }

    IServiceManifest ServiceManifest { get; }

    public void Main(string[] args)
    {
      foreach (Type type in ServiceManifest.Services)
      {
        Console.WriteLine(type.FullName);
      }
    }
  }
}
```

The K command executes the native process (klr.exe), specifies the application base as the current directory and then specifies the default application host. The native process has no special knowledge of the default application host—it simply looks for a standard entry point in the Microsoft.Framework.ApplicationHost assembly and calls into it.

If you don't want to use the default application host you can invoke the native layer to call into your application directly. To do this, first build the project to generate the DLL for the console application. Make sure you checked the "Produce outputs on build" option in the Build properties for the project. You can find the build output under the artifacts directory in your solution folder. Navigate to the directory containing the built-in DLL for the KRE flavor you're using and call "klr.exe <DLL name>" to see the output from the console application.

Invoking a DLL directly is a very low-level and raw way to write an application. You aren't using the default application host so you forego project.json support and the improved NuGet-based dependency management support. Instead, any libraries you depend on are simply loaded from the specified lib folders. For the rest of this article I'll use the default application host.

You can use the IServiceManifest service to enumerate all of the services that are available from the runtime environment. You don't have to specify any additional dependencies to use this assembly at all because it's an Assembly Neutral Interface (ANI). ANIs are types that are identified solely by their name and namespace and they are a feature of the KRE. Two assembly-neutral types in different assemblies but with the same name and namespace are considered equivalent. This means that common abstractions can simply be declared locally instead of having to declare a dependency on a common component.

You can define your own assembly-neutral IServiceManifest interface in your console app like this:

```
namespace Microsoft.Framework.DependencyInjection.ServiceLookup
{
  [AssemblyNeutral]
  public interface IServiceManifest
  {
    IEnumerable<Type> Services { get; }
  }
}
```

But where does the AssemblyNeutralAttribute come from? Well, it's assembly-neutral of course! So you declare it locally, too:

```
namespace Microsoft.Net.Runtime
{
  [AssemblyNeutral]
  [AttributeUsage(AttributeTargets.All, Inherited = false, AllowMultiple = true)]
  public sealed class AssemblyNeutralAttribute : Attribute
  {
  }
}
```

Figure 2 **The Startup Class**

```
namespace WebApplication1
{
  public class Startup
  {
    public void ConfigureService(IServiceCollection services)
    {
      // Add services for your application here
    }

    public void Configure(IApplicationBuilder app)
    {
      // Configure your application pipeline here
    }
  }
}
```

These assembly-neutral types will get unified at run time with the same types being used by the KRE.

The code in **Figure 1** injects the IServiceManifest service into the application by adding a constructor to the Program class and then iterates through the services.

The application host will enable support for project.json, add additional assembly resolvers (for dealing with NuGet packages, project references and the like) and make a number of additional services available to the application before calling the entry point for the application. When you run the console app, the output should now look like this:

```
Microsoft.Framework.Runtime.IAssemblyLoaderContainer
Microsoft.Framework.Runtime.IAssemblyLoadContextAccessor
Microsoft.Framework.Runtime.IApplicationEnvironment
Microsoft.Framework.Runtime.IFileMonitor
Microsoft.Framework.Runtime.IFileWatcher
Microsoft.Framework.Runtime.ILibraryManager
Microsoft.Framework.Runtime.ICompilerOptionsProvider
Microsoft.Framework.Runtime.IApplicationShutdown
```

There are services for doing file watching, traversing the structure of the "libraries" (projects, packages, assemblies) in the application, getting the compilation options being used, and for shutting down the application. Because ASP.NET 5 and the KRE are still in preview and under active development, the exact list of services you see might differ.

## Hosting

The ASP.NET hosting layer runs on top of the KRE and is responsible for finding the Web server to run on, finding the startup logic for the application, hosting the application on the server and then cleaning up when the application is shut down. It also provides a number of additional hosting-related services to applications.

The ASP.NET hosting DLL (Microsoft.AspNet.Hosting, at bit.ly/1uB6ulW) has an entry point method that gets called by the KRE application host. You can configure which Web server you want to use by specifying the --server command-line option or other sources of configuration data, like config.json or environment variables. The hosting layer then loads the specified server assembly/type to find an IServerFactory that can be used to initialize and start the server. Typically, the server must be listed in your dependencies in project.json so that it can be loaded.

Note that commands defined in project.json are really just sets of additional command-line arguments to klr.exe. For example, the default ASP.NET Starter Web App project template includes a bunch of commands in project.json that looks something like this:

```
"commands": {
  /* Change the port number when you are self hosting this application */
  "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Server.WebListener
    --server.urls http://localhost:5000",
  "gen": "Microsoft.Framework.CodeGeneration",
  "ef":  "EntityFramework.Commands"
},
```

So if you were to run "k web," what you're really running is:

```
klr.exe --appbase . Microsoft.Framework.ApplicationHost Microsoft.AspNet.Hosting
  --server Microsoft.AspNet.Server.WebListener --server.urls http://localhost:5000
```

## Startup

The ASP.NET hosting layer is also responsible for finding the startup logic for your application. Typically, your application startup logic is defined in a Startup class, with a Configure method for setting up

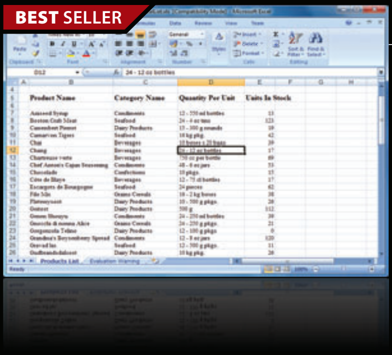
ComponentSource®
The Definitive Source of Software Components
www.componentsource.com

## Help & Manual Professional | from $583.10

ec software

**Easily create documentation for Windows, the Web and iPad.**

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

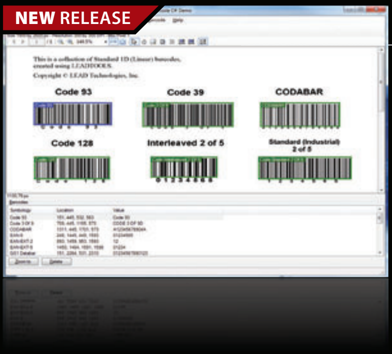


## Aspose.Total for .NET | from $2,449.02

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 Barcode Pro SDK V19 | from $1,295.00 SRP

LEAD TECHNOLOGIES

**Add powerful Barcode detection, reading/writing functionality to desktop, tablet & mobile apps.**

- Supports 100+ 1D linear barcode types including UPC, EAN, Code 128, 2 of 5 & GS1 DataBar
- Supports 2D Barcode types including QR Code, Data Matrix, Aztec, PDF417 & MicroPDF417
- Load, save, view & convert 150+ raster image formats including JPEG, JPEG 2000, PNG & TIFF
- Over 200 image processing functions for enhanced recognition accuracy
- Includes native libraries for .NET, C/C++, WinRT, iOS, OS X, Android, Linux, Silverlight & more

## 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

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 NETWORK

GSA Schedule
Contract GS-35F-0188R

Figure 3 **Implementing Middleware as a Reusable Class**

```
using Microsoft.AspNet.Builder;
using Microsoft.AspNet.Http;
using System.Threading.Tasks;

public class XHttpHeaderOverrideMiddleware
{
  private readonly RequestDelegate _next;

  public XHttpHeaderOverrideMiddleware(RequestDelegate next)
  {
    _next = next;
  }

  public Task Invoke(HttpContext httpContext)
  {
    var headerValue =
      httpContext.Request.Headers["X-HTTP-Method-Override"];
    var queryValue =
      httpContext.Request.Query["X-HTTP-Method-Override"];

    if (!string.IsNullOrEmpty(headerValue))
    {
      httpContext.Request.Method = headerValue;
    }
    else if (!string.IsNullOrEmpty(queryValue))
    {
      httpContext.Request.Method = queryValue;
    }

    return _next.Invoke(httpContext);
  }
}
```

your request pipeline and a ConfigureServices method for configuring any services your application needs, as shown in **Figure 2**.

In your Configure method you use the IApplicationBuilder interface to build the request pipeline for your application. The application builder lets you "Use" middleware, create "New" application builders and "Build" a request delegate. The request delegate is the core runtime construct in ASP.NET 5. A request delegate takes an HttpContext and does something useful with it asynchronously:

```
public delegate Task RequestDelegate(HttpContext context);
```

ASP.NET 5 middleware takes as input the next request delegate in the pipeline and provides a request delegate with the middleware logic. The returned request delegate may or may not call the next request delegate in the pipeline. As a shortcut for running middleware logic that doesn't call the next request delegate, you can use the Run extension method on IApplicationBuilder:

```
app.Run(async context => await context.Response.WriteAsync("Hello, world!"));
```

This would be the same as writing the following inline middleware where the next parameter is simply ignored:

```
app.Use(next => async context =>
  await context.Response.WriteAsync("Hello, world!"));
```

To create reusable middleware, you can write it as a class where, by convention, the next request delegate is injected into the constructor along with any additional services or parameters the middleware needs. You then implement the request delegate for the middleware as an asynchronous Invoke method, as shown in **Figure 3**.

You can use middleware following this convention using the UseMiddleware<T> extension method on IApplicationBuilder. Any additional options you pass into this method will get injected into the middleware constructor after the next request delegate and any injected services. By convention, middleware should also define its own specific Use extension method on IApplicationBuilder, like this:

```
public static class BuilderExtensions
{
  public static IApplicationBuilder UseXHttpHeaderOverride(
    this IApplicationBuilder builder)
  {
    return builder.UseMiddleware<XHttpHeaderOverrideMiddleware>();
  }
}
```

You then add the middleware to the application pipeline, like so:

```
public class Startup
{
  public void Configure(IApplicationBuilder app)
  {
    app.UseXHttpHeaderOverride();
  }
}
```

ASP.NET 5 comes with a rich set of prebuilt middleware. There's middleware for handling static files, routing, error handling, diagnostics and security. You can find, download and install middleware from Microsoft and the community as NuGet packages on nuget.org.

## Configuring Services

As with all the other entry points you've seen in this stack, the Startup class can have services injected from the hosting layer in its constructor or as additional parameters on the Configure and ConfigureServices methods. If you enumerate the services available at startup (using IServiceManifest), you see there are now even more services available:

```
Microsoft.Framework.Runtime.IAssemblyLoaderContainer
Microsoft.Framework.Runtime.IAssemblyLoadContextAccessor
Microsoft.Framework.Runtime.IApplicationEnvironment
Microsoft.Framework.Runtime.IFileMonitor
Microsoft.Framework.Runtime.IFileWatcher
Microsoft.Framework.Runtime.ILibraryManager
Microsoft.Framework.Runtime.ICompilerOptionsProvider
Microsoft.Framework.Runtime.IApplicationShutdown
Microsoft.AspNet.Hosting.IHostingEnvironment
Microsoft.Framework.Runtime.IProjectResolver
Microsoft.Framework.Logging.ILoggerFactory
Microsoft.Framework.Runtime.ICache
Microsoft.Framework.Runtime.ICacheContextAccessor
Microsoft.Framework.DependencyInjection.ITypeActivator
Microsoft.Framework.Runtime.IAssemblyLoadContextFactory
```

The IHostingEnvironment services gives you access to the Web root path for your application (typically your www folder) and also an IFileProvider abstraction for your Web root. The IProjectResolver can be used to find other projects in your solution. There are logging and caching services and a type activator that's dependency injection-aware. The IAssemblyLoadContextFactory gives you an abstraction for creating new assembly load contexts.

You configure existing services and add new ones for your application in the ConfigureServices method in your Startup class. The ConfigureServices method takes an IServiceCollection to which you can add additional services or modify existing ones. ASP.NET 5 comes with a simple built-in IoC container that's set up in the Managed Entry Point layer in order to bootstrap the system, but you can easily replace the built-in container with your container of choice.

Frameworks typically provide an Add extension method for adding their services to the IServiceCollection. For example, here's how you add the services used by ASP.NET MVC 6:

```
public void ConfigureServices(IServiceCollection services)
{
  // Add MVC services to the services container
  services.AddMvc();
}
```

When you add services to the service collection, they can be one of three types: transient, scoped or singleton. Transient services are

# Data Quality Tools for Developers

## A better way to build in data verification

Since 1985, Melissa Data has provided the tools developers need to enhance databases with clean, correct, and current contact data. Our powerful, yet affordable APIs and Cloud services provide maximum flexibility and ease of integration across industry-standard technologies, including .NET, Java, C, and C++. Build in a solid framework for data quality and protect your investments in data warehousing, business intelligence, and CRM.

- **Verify international addresses for over 240 countries**
- **Enhance contact data with phone numbers and geocodes**
- **Find, match, and eliminate duplicate records**
- **Sample source code for rapid application development**
- **Free trials with 120-day ROI guarantee**

**Address Verification**

**Phone Verification**

**Email Verification**

**Geocoding**

**Matching/ Dedupe**

**Change of Address**

***Melissa Data.***
***Architecting data quality success.***

MELISSA DATA®

| United States | United Kingdom | Germany | India |

created each time they're requested from the container. Scoped services are created only if they don't already exist in the current scope. For Web applications, a container scope is created for each request, so you can think of scoped services as per request. Singleton services are only ever created once.

Web.config and System.Configuration-style app.config files aren't supported in ASP.NET 5. Instead, ASP.NET 5 comes with a new, simplified Configuration API (Microsoft.Framework.ConfigurationModel, at bit.ly/1yxC7gA) for working with configuration data. The new Configuration API lets you retrieve and manipulate configuration data from a variety of sources. Default configuration providers are included for JSON, XML, INI, command-line parameters and environment variables. You can specify multiple providers and they're used in the order they're added. By supporting environment-based configuration, you can more easily deploy an application into an environment and have it pick up the appropriate settings for that environment.

To initialize the configuration data, add your desired configuration providers to a new Configuration instance:

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

You can then request configuration values by name:

```
string user = Configuration.Get("user");
string password = Configuration.Get("password");
```

You can make configuration data available in a strongly typed way throughout your application using the options model. Options are just Plain Old C# Object (POCO) classes that have a bunch of properties used for configuration. You make options available in your application by calling AddOptions, which adds the IOptions<TOption> service to the container. This service can be used to access options of different types wherever dependency injection is supported.

To configure options, you add IConfigureOptions<TOptions> services to the container. The default implementation of the IOptions<TOption> service will gather up all the IConfigureOptions<TOption> services and use them to configure the options instance before making the options instance available to consumers. As a shortcut for adding options configuration logic, you can use the Configure<TOption> method, like this:

```
services.Configure<MvcOptions>(options => options.Filters.Add(
  new MyGlobalFilter()));
```

The options model and the configuration model work hand-in-hand. You can easily bind configuration data to options, matching by property name by using the Configure overloads that take a Configuration instance, like this:

```
services.Configure<MyOptions>(Configuration);
```

## Request Processing

Once the Web server is started, it starts listening for requests and invoking the application pipeline with each request. The server surfaces each request as a server environment object that exposes a set of fine-grained feature interfaces. There are feature interfaces for sending files, Web sockets, session support, client certificates and more. You can see the full list of supported feature interfaces on GitHub at bit.ly/1Dh7eBC. For example, here's the feature interface for the HTTP request:

```
[AssemblyNeutral]
public interface IHttpRequestFeature
{
  string Protocol { get; set; }
  string Scheme { get; set; }
  string Method { get; set; }
  string PathBase { get; set; }
  string Path { get; set; }
  string QueryString { get; set; }
  IDictionary<string, string[]> Headers { get; set; }
  Stream Body { get; set; }
}
```

You don't typically code directly against the various feature interfaces. Web servers use feature interfaces to expose low-level functionality to the host layer. The hosting layer wraps these feature interfaces in a strongly typed HttpContext that's then passed through to the application. This level of loose coupling between the Web server, the hosting layer and the applications allows servers to be implemented and reused without being tied to one specific hosting model, and it allows applications to be hosted on different servers. ASP.NET 5 comes with built-in support for running on IIS (Microsoft.AspNet.Server.IIS), directly on a thin HTTP.SYS wrapper (Microsoft.AspNet.Server.WebListener), and on a new .NET cross-platform Web server called Kestrel (Microsoft.AspNet.Server.Kestrel).

The Open Web Interface for .NET (OWIN) community standard (owin.org) shares similar goals of loose coupling for Web applications. OWIN standardizes how .NET servers and applications should talk to each other. ASP.NET 5 supports OWIN through the Microsoft.AspNet.Owin package (bit.ly/15IQwA5). You can host ASP.NET 5 on an OWIN-based Web server (bit.ly/1DaU4FZ), and you can use OWIN middleware within the ASP.NET 5 pipeline (bit.ly/1EqmoIB).

The Katana Project (katanaproject.codeplex.com) was Microsoft's initial effort to implement support for OWIN, and many of the ideas and concepts in ASP.NET 5 were derived from it. Katana had a similar model for building up a middleware pipeline and for hosting on multiple servers. However, unlike Katana, which exposed the low-level OWIN abstractions directly to the developer, ASP.NET 5 instead bridges from OWIN to more convenient and user-friendly abstractions. With a little extra code you can still use Katana middleware in ASP.NET 5 using the OWIN bridge (see an example of how to do this at bit.ly/1BpaXe2).

## Wrapping Up

The ASP.NET 5 runtime has been built from the ground up to support cross-platform Web applications. ASP.NET 5 has a flexible, layered architecture that can run on the .NET Framework, .NET Core or even cross-platform on Mono (and in the future on .NET Core as well!). The new ASP.NET 5 hosting model makes it easy to compose applications and host them on the Web server of your choice. The entire runtime is built to support dependency injection and to be easy to configure. I hope you've enjoyed learning about the new ASP.NET 5 runtime! Your feedback and contributions are welcome through our GitHub project at github.com/aspnet/home. ■

**Daniel Roth** *is a senior program manager on the ASP.NET team currently working on ASP.NET 5. His passions include .NET development and delighting customers by making frameworks simple and easy-to-use.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Glenn Condron, David Fowler, Murat Girgin and Chris Ross*

WPF lives!

XCEED **Business Suite** for WPF

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

XCEED
MULTI-TALENTED COMPONENTS
xceed.com

# The Microsoft .NET Framework in Embedded Applications

Colin Miller

The world has come a long way from the first context-sensitive C code editor in the early 1980s. Tooling and languages developed initially for desktop apps have been extended to encompass server, cloud and other environments. The Microsoft .NET Framework and Visual Studio provide state-of-the-art tools that greatly increase programmer efficiency. The .NET Framework tools and languages provide rich libraries and collaboration tools, and protect programmers from the most common mistakes.

These advances have not, however, been made available to developers working on embedded devices. The market for embedded tools has lacked the scale necessary to attract the kind of investments made in tools for PC, server and cloud applications. The emergence of "intelligent devices" of all descriptions is bringing about a change in this imbalance.

This article discusses:

- Controlling devices with phone commands
- Becoming an embedded programmer
- Developing device controls with embedded programming

Technologies discussed:

Microsoft .NET Framework, Visual Studio 2013, .NET Micro Framework, Internet of Things

Code download available at:

msdn.microsoft.com/magazine/msdnmag0315

With the .NET Micro Framework, there has been a new stream of products, including small devices that used to require a different set of embedded skills to develop code. Instead these were developed by individuals and teams with .NET skills. If you're a .NET developer and have thought about how to leverage small embedded devices in your applications, hopefully this article will convince you that you can use your skills to participate in the explosion of the Internet of Things (IoT) and intelligent devices.

Intelligent devices have local computing capability and can communicate with other devices and potentially the cloud.

Why should you care? One recent analysis estimated 94 percent of the processors expected to ship in 2017 won't be of sufficient capacity to support a traditional .NET application environment. With the .NET Micro Framework, you can extend your reach.

The .NET Micro Framework was initially developed in the SPOT watch project. It has been part of the .NET family for the last seven years. It runs on 32-bit processors too small to support full OSes. It's actually .NET running directly on the hardware with the execution engine and type system integrated into the executable image.

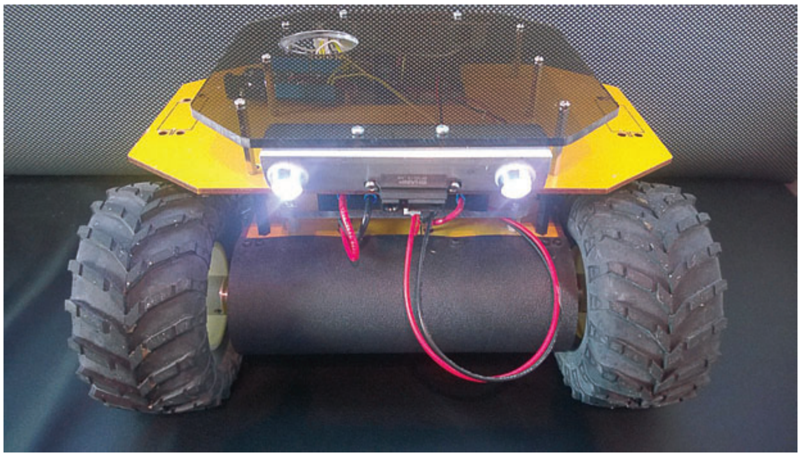


Figure 1 **Bert the Robot**

The goal of the .NET Micro Framework is to help .NET developers create an application that runs on a small 32-bit MCU-based device for embedded applications. At the Build conference in 2014, Microsoft described a range of OS platforms covering the IoT space with Windows Embedded Standard at the high end and the .NET Micro Framework for the small device end.

## Embedded Development

Intelligent devices have local computing capability and can communicate with other devices and potentially the cloud. For this article, I've created a small mobile robot I can control from a smartphone running the Windows Phone OS. The objective is to show how .NET skills you use for the Windows Phone app are just as applicable on the robot side. The tools and language support are new in this environment, and the development environment for working back and forth between the two is seamless.

Meet Bert—Bert is a mobile robot with four powered wheels (see **Figure 1**). He includes a Netduino 2 board available for about $35 by Secret Labs. A simple controller lets you control the motors on each side of the robot to go forward and reverse. There's no speed control, though.

For communication, Bert includes a Bluetooth module that communicates with the Netduino over a serial interface. To ensure you won't crash Bert into anything or run him off a cliff, there are IR proximity sensors included in the front and rear. There are also headlights and turn signals just for fun.

The simple Windows Phone app is derived from a demo presented at the Build conference. It connects to the correct device and uses the onboard accelerometer to steer Bert around by sending commands over Bluetooth. You can see the interface in **Figure 2**, with the dropdown selection in **Figure 3**. Select Enumerate to pick the connected device to which you'll send commands. The commands are reflected in the directional letters on the screen.

You can port the .NET Micro Framework to another hardware platform with the .NET Micro Framework Porting Kit (bit.ly/1wp57qm). Use the kit to write the low-level interfaces to the specific hardware upon which you're running. This is closer to classical embedded programming activities. The good news is vendors have already done this for you. The Netduino board used in this application is an example of having all you need to start coding to the board in the Netduino SDK (bit.ly/1CQygzz).

Netduino is a Cortex-M3 processor from STMicro, with 192KB to 384KB of Flash and 60KB to 100KB of RAM and is pin-compatible with Arduino shields. You can find other devices that support the .NET Micro Framework at GHI Electronics, Mountaineer, Sparkfun, Adafruit, Amazon and other stores.

The board is also open source hardware, so once you write your application and want to commercialize it, you have several options. You can just use the Netduino board for low-volume applications, but if you're targeting a higher volume or have specific requirements such as wanting to integrate additional electronics on the same PCB, you can take the open source design files to a board manufacturer and go from there.

Because I started with the Netduino board, I installed the Netduino SDK. This includes the .NET Micro Framework SDK with Visual Studio templates and all the necessary library support. For other development boards, you might need to install the .NET Micro Framework SDK separately. The SDK comes with an emulator, so you don't need to get any hardware to play with the .NET Micro Framework, although using hardware is more productive if you already know what sensors you want to use.

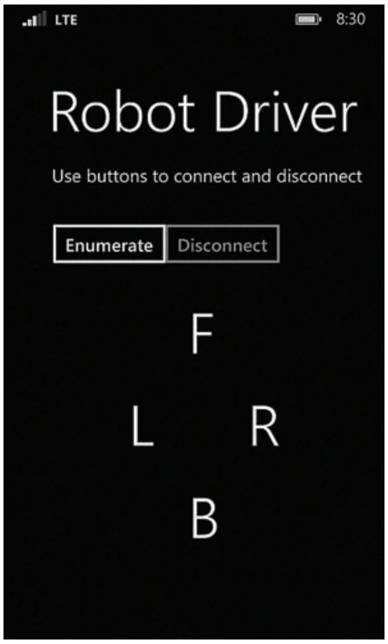


Figure 2 **Windows Phone App Interface**

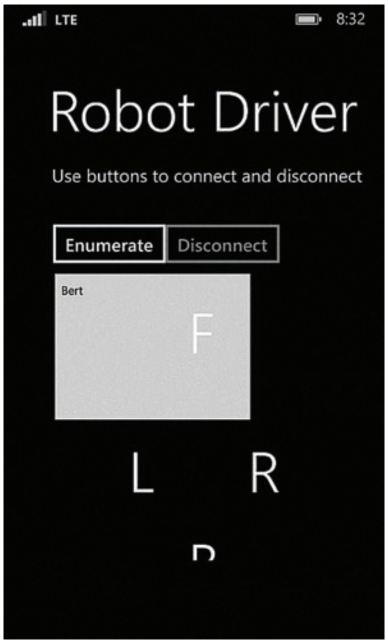


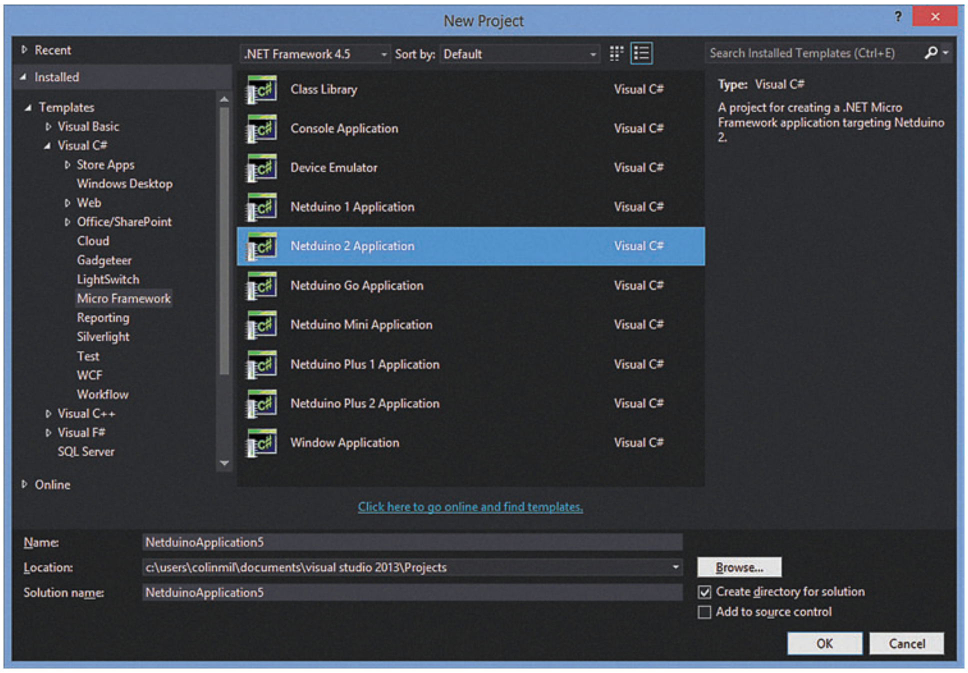
Figure 3 **Device Selection Dropdown**

## Create the Embedded Application

After you install the Netduino SDK (which contains the .NET Micro Framework SDK), starting a .NET Micro Framework embedded application is just like starting any other project within Visual Studio. You select New Project and the type of application you want to start. As shown in **Figure 4**, I've selected a Netduino 2 Application to match the board on Bert.

Once you've started the project, you can jump right into creating the communication link between the robot and the phone.

Like many embedded applications, Bert runs on batteries. This means your code has to be written to conserve power usage. The good news is the .NET Framework and the .NET Micro Framework are perfectly suited for that. If there's no thread to run, the .NET Micro Framework can automatically put the processor into a lower power state. It's also becoming common to add a second efficient processor to monitor the interrupts and put the main processor into an even lower power state. You'll soon see if the .NET Framework is well-suited for addressing this type of programming.



| | |
|---|---|
| Class Library | Visual C# |
| Console Application | Visual C# |
| Device Emulator | Visual C# |
| Netduino 1 Application | Visual C# |
| Netduino 2 Application | Visual C# |
| Netduino Go Application | Visual C# |
| Netduino Mini Application | Visual C# |
| Netduino Plus 1 Application | Visual C# |
| Netduino Plus 2 Application | Visual C# |
| Window Application | Visual C# |

Type: Visual C#
A project for creating a .NET Micro Framework application targeting Netduino 2.

Name: NetduinoApplication5
Location: c:\users\colinmi\documents\visual studio 2013\Projects
Solution name: NetduinoApplication5
[✓] Create directory for solution
[ ] Add to source control

Figure 4 **Start a .NET Micro Framework Embedded Application**

The code in this application runs in response to an external event and then only what's needed to respond. So you'll see all execution resides in an event handler or in a thread that's suspended most of the time, like so:

```
public class Program
  {
    public static void Main()
    {
      RunRobot myRunRobot = new RunRobot();
      myRunRobot.Start();
      while (true)
      {
        Thread.Sleep(Timeout.Infinite);
      }
    }
  }
```

The RunRobot constructor initializes the IO instances that I'll look at in more detail and the RunRobot.Start method initializes the interrupts that trigger the actual execution:

```
public void Start()
  {
  // Enable interrupts on BT input on the HC-06 radio
  hc06.DataReceived += hc06_DataReceived;

  // Use onboard button to turn on/off accepting commands
  button = new InterruptPort(Pins.ONBOARD_SW1, false, Port.ResistorMode.Disabled,
    Port.InterruptMode.InterruptEdgeHigh);

  // Start monitoring for button presses
  button.OnInterrupt += new NativeEventHandler(button_OnInterrupt);
  }
```

## Get Bluetooth Working

There are three steps to getting the Bluetooth communication working between Bert and the Windows Phone app:

1. Wire up the Bluetooth module on the robot
2. Send commands from the phone app
3. Receive those commands on Bert

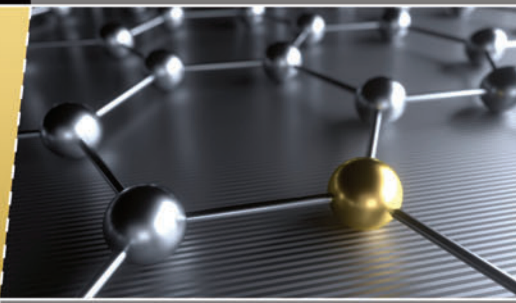
Because the Netduino is Arduino-compatible, for many of the connections, you might be able to find a shield you can plug in that makes all the necessary connections for you. The Arduino is a popular hobbyist platform that has a rich ecosystem of peripherals. You can craft a broad range of solutions from existing modules.

If you don't want to spend time with wiring, look into the Gadgeteer products from GHI Electronics. With these boards and components, you can use a wizard to decide where to plug the components into standard connections (see **Figure 5**), so you don't need any knowledge of wiring connections. The module interfaces are at a similarly high level. Like Arduino, Gadgeteer has a rich set of existing modules with which to create your solution. Once you have the modules connected, you're still coding in the .NET Micro Framework. This is particularly useful for rapidly prototyping your ideas.

The code in this application runs in response to an external event and then only what is needed to respond.

In the case of Bert, I'm not using Gadgeteer, but collecting my own modules. The Bluetooth module (HC-06) uses a serial interface with four connections—VIN, GND, RX and TX. Connect the VIN (Voltage In) to either the 5V or 3.3V outputs from the Netduino, then connect GND to one of the GND connections. Looking at the I/O table for the Netduino in **Figure 6**, you'll see to connect the BT module RX (Receive) to the Netduino TX (Transmit) and

# 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

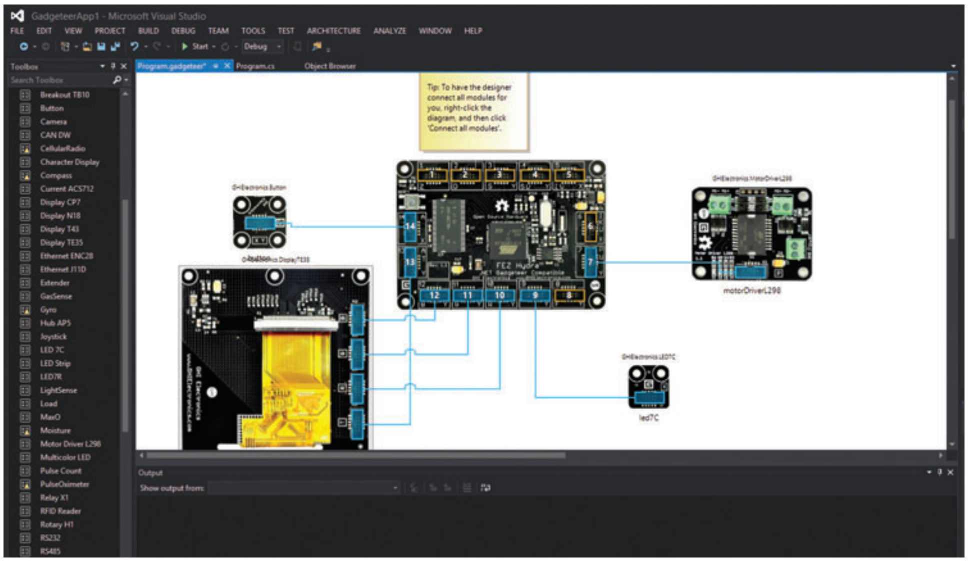
**www.amyuni.com**

**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

the BT Module TX to the Netduino RX of the COM1 serial port. If needed, there are a number of examples of connecting these modules to common microcontrollers on the Web.

With those connections made, you can power up the Netduino (which powers the BT module) and pair the device with the phone. The default name for the BT module is HC-06, so you can pair with that. I changed the name to Bert using AT commands. Send an array of bytes with the characters "AT+NAMEBert" to change the name of the BT module. You can also use this interface to configure the Bluetooth module, including things like the communication baud rate.


GadgeteerApp1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Toolbox
Program.gadgeteer*
Program.cs
Object Browser
Output
Show output from:

Tip: To have the designer connect all modules for you, right-click the diagram, and then click 'Connect all modules'.

Figure 5 **The Gadgeteer Wizard Showing Connections**

Now you can use the phone to connect to Bert as you would any other BT device and start sending commands. First, I access the onboard accelerometer and set up a reasonable reporting interval, so as not to swamp poor Bert:

```
accelerometer = Accelerometer.GetDefault();
if (accelerometer != null)
{
  uint minReportInterval = accelerometer.MinimumReportInterval;
  desiredReportInterval = minReportInterval > 250 ? minReportInterval : 250;
    // Slowing this way down to prevent flooding with commands
  accelerometer.ReportInterval = desiredReportInterval;
    // Add event for accelerometer readings
  accelerometer.ReadingChanged += new TypedEventHandler<Accelerometer,
    AccelerometerReadingChangedEventArgs>(ReadingChanged);
}
```

The ReadingChanged event handler sends commands, as shown in **Figure 7**.

## Receive and Execute Commands

Now you can consider receiving a command on the robot side. The BT module is wired to the first serial port on the Netduino. Create a SerialPort instance and a message buffer, just as you would on other versions of the .NET Framework:

```
SerialPort hc06;                 // The BT radio
byte[] cmdBuffer = new byte[32]; // The input buffer for receiving commands
```

Then configure the serial port for communication with the Bluetooth module (9600 baud is the default; I could reset this using AT commands, but this will be a very sparse communication to simplify the program, so faster speeds are unnecessary):

```
// Set up the Bluetooth radio communication
hc06 = new SerialPort(SerialPorts.COM1, 9600, Parity.None, 8, StopBits.One);
hc06.ReadTimeout = 1000;
hc06.Open();
```

Then I set up a DataReceived event handler that triggers most of the processing within the app. In other words, Bert sleeps until he receives a command. The processor stays up long enough to set the motors in the right state and goes back to sleep:

```
// Enable interrupts on BT input
hc06.DataReceived += hc06_DataReceived;
```

Incoming commands are processed in the event handler, as shown in **Figure 8**.

None of that code is any different than what you would write on the desktop.

## Set Bert in Motion

The motor controller I use simply requires me to set four inputs in some logical state to determine how the wheels turn. The logic is:

```
/*
A2  A1  Motor 1         B2  B1  Motor 2
0   0   Shaft locked    0   0   Shaft locked
0   1   Clockwise       0   1   Clockwise
1   0   Anti-clockwise  1   0   Anti-clockwise
1   1   Shaft locked    1   1   Shaft locked
*/
```

Like many embedded applications, Bert runs on batteries. This means your code has to be written to conserve power usage.

To write to these, I create an OutputPort and tie it to one of the Netduino digital pins shown in **Figure 6**. OutputPort is a unique class in the .NET Framework because devices that run the full .NET Framework typically can't address individual digital I/O ports. There are a handful of new classes in the .NET Micro Framework to handle the additional I/O types and other issues unique to this application space like power management:

```
// Motor controller output
A1 = new OutputPort(Pins.GPIO_PIN_D2, false);
  // A is the left tires when viewed from the front
A2 = new OutputPort(Pins.GPIO_PIN_D3, false);
B1 = new OutputPort(Pins.GPIO_PIN_D4, false);
  // B is the right tires when viewed from the front
B2 = new OutputPort(Pins.GPIO_PIN_D5, false);
```

Of drivers who own family cars, **86%** would rather be ***driving one of these.***

***Move into the Fast Lane with MobileTogether®***

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

ALTOVA®
MobileTogether®

## *Ditch the Minivan of Mobile Development*

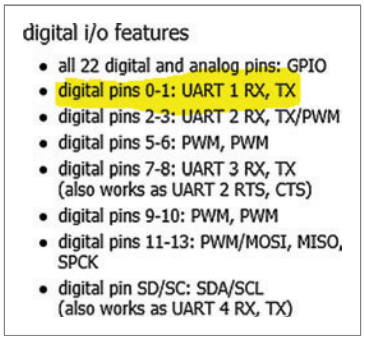
It's no longer practical to wait weeks or months for a cutting-edge mobile business solution. Your team needs to create, test, and deploy to all devices at the speed of business. With MobileTogether, we've removed the roadblocks.

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

*www.altova.com/MobileTogether*

Setting them is as simple as writing true or false:

```
switch (cmd)
{
  case 'F':
  {
    Debug.Print("Forward");
    A1.Write(false);   // Left forward = false/true
    A2.Write(true);
    B1.Write(true);    // Right forward = true/false
    B2.Write(false);
    break;
  }
```


digital i/o features

- all 22 digital and analog pins: GPIO
- digital pins 0-1: UART 1 RX, TX
- digital pins 2-3: UART 2 RX, TX/PWM
- digital pins 5-6: PWM, PWM
- digital pins 7-8: UART 3 RX, TX (also works as UART 2 RTS, CTS)
- digital pins 9-10: PWM, PWM
- digital pins 11-13: PWM/MOSI, MISO, SPCK
- digital pin SD/SC: SDA/SCL (also works as UART 4 RX, TX)

Figure 6 **Netduino IO Pins Defined**

## Drive Carefully

I don't want Bert to run into things for his own sake and for the safety of things and people around him, so I installed InfraRed proximity sensors from Sharp in the front and back. These sensors return a value from 0.0 to 1.0 depending on the distance to the nearest object that reflects the projected IR light.

The front sensor is pointed diagonally down sensing the ground about 18 inches in front of Bert. This lets me use one sensor to detect objects, as well as the ground falling away if it's approaching a set of stairs, for example. The values these sensors return are analog, which means I create an instance of another class unique in the .NET Framework to the .NET Micro Framework—the AnalogInput class:

```
AnalogInput foreRange;    // Forward range finder input
AnalogInput backRange;    // Rear range finder input
// Set up range finder for collision and edge avoidance
foreRange = new AnalogInput(Cpu.AnalogChannel.ANALOG_5, 100, 0, -1);
backRange = new AnalogInput(Cpu.AnalogChannel.ANALOG_4, 100, 0, -1);
```

This constructor identifies the specific analog port to which I'm connecting, a multiplier I can use to convert the output to a more convenient range, an offset to modify the output for convenience and the precision of the output with -1 meaning maximum precision. To make these measurements periodically and let the processor sleep as much as possible, the readings are triggered by a timer. I set this when I start accepting commands from the phone and dispose of it when I'm not:

```
// Start timer for rangeFinders
Timer rangeReader = new Timer(new TimerCallback(rangeTimerHandler), null, 0, 100);
```

## General Purpose Interrupt

The last new class I need to introduce is the InterruptPort class. This lets me monitor changes in digital input and triggers on changes like the leading edge of a change from Low to High, for example. I'm using this for the button on the Netduino board that gets Bert to respond to commands from the phone. This means he ignores commands until this button is pressed and ignores them after pressing it again:

```
// Use onboard button to turn on/off accepting commands

button = new InterruptPort(Pins.ONBOARD_SW1, false,
  Port.ResistorMode.Disabled,
  Port.InterruptMode.InterruptEdgeHigh);
```

Pins.ONBOARD_SW1 is the internal connection for the switch on the Netduino board. You can connect a switch to any of the digital ports listed in **Figure 6**. The second argument is whether I need a glitch filter applied to this switch input to avoid multiple triggers for a single button press.

The next parameter is whether there's a pull up or pull down resistor on the switch. This defines the default port position when the button isn't pressed (High or Low). The next parameter is set to trigger the interrupt on the transition back to High so your finger is out of the way if Bert takes off immediately. With the InterruptPort instance created, I can assign the interrupt handler:

```
// Start monitoring for button presses
button.OnInterrupt += new NativeEventHandler(button_OnInterrupt);
```

The interrupt handler starts Bert monitoring any commands coming from the phone, as you can see in **Figure 9**.

Deploying code to your device is as easy as connecting it and pressing F5.

## Deploy and Debug

Deploying code to your device is as easy as connecting it and pressing F5, just as with the Windows Phone app. However, the setup is a little different. In the Project | Properties dialog, there's a .NET Micro Framework tab where you can specify the transport (USB, serial or TCP/IP) and identify the device. The default is USB. It will automatically identify a .NET Micro Framework device, so unless you're connected to several devices, you may never need to be in this dialog box. At this point, the solution is built and the assemblies deployed to the device. You can follow all this in the Output window in Visual Studio. Execution starts immediately and you can start debugging the code running on the device.

Figure 7 **The Event Handler for Sending Commands to Bert**

```
AccelerometerReading reading = args.Reading;
if (reading.AccelerationX > .4)  // These are arbitrary levels
                                 // set to what seemed easy
{
                        // Send Right
  if( cmdSent != 'R')   // Don't resend the same command over and over
  {
      // Update the phone display
    txtRight.Text = "On";
    txtLeft.Text = " L";
    txtForward.Text = " F";
    txtBack.Text = " B";
      // Send the command
    packet = Encoding.UTF8.GetBytes("R");
    cmdSent = 'R';
    SendData(packet);
  }
```

Figure 8 **The Inbound Commands to Start Bert Moving**

```
do
{
  try
  {
    count = hc06.Read(cmdBuffer, 0, 1);  // Should read one byte at a time
  }
  catch (Exception) { }        // Just eat exceptions
  if (count > 0)
  {
    cmd = (char)cmdBuffer[0];
    executeCmd(cmd);   // Execute command immediately
  }
} while (count > 0);
```

rackspace®

the #1 **managed cloud** company

MANAGED
**CLOUD**

**Focus on building your .NET app, we'll help with the IT ops.**

Get the tools you need to start building your app or website right away on a world-class, Windows® based infrastructure. Use our .NET SDK and client-side utilities when you build your application on Rackspace.

Run unlimited domains for one monthly price. We continually update our platform with the latest versions of Microsoft® .NET® and SQL Server®, with continued support for .NET 2.0 and later to accommodate all of your app needs.

We'll also handle tasks like patching and upgrading, and our constant monitoring will help you catch issues before they become visible to your users. These are just some of the ways we put **Fanatical Support®** into action for you. So get back to focusing on dev, and let us help with the ops.

www.**rackspace**.com

## Embedded Applications and the .NET Micro Framework

**Embedded applications** typically perform a targeted set of functions on hardware crafted to meet the minimum requirements. Crafting targeted and tuned embedded applications requires a higher initial investment. That investment is recouped over a longer product lifecycle.

One of the biggest challenges for traditional embedded development is the time it takes to complete projects. A large proportion of embedded projects are abandoned because the cost becomes too large or the requirements change before the projects can get to market.

A number of things are affecting the dynamics of embedded applications. When Microsoft first released the .NET Micro Framework seven years ago, most embedded applications were built on 8-bit processors to reduce cost, achieve needed performance requirements and meet power requirements (the ability to run on a battery for extended periods, for example).

At the same time, most of the growth in the embedded space targeted 32-bit processors like the ARM7. These processors were continuing to come down in price and offered adequate performance for many applications and power consumption was improving. The newer Cortex processors are continuing to attract many new embedded applications to the 32-bit family.

Recently, there have been more devices that are integral parts of broader applications, including embedded devices, gateways, cloud connectivity, databases, business intelligence applications and so on. These new applications change the economics of embedded applications.

It becomes more important to adhere to common communication protocols. More of these new devices are in the consumer realm, so expecting long development cycles means increased product risks. Time to market is even more important for these applications and during the coming explosion of "intelligent devices."

The other impact of these new types of applications is their breadth alone. They potentially bring together the desktop developer and his expertise in high-scale communications, databases and business analytics, and cloud protocols with the embedded developer and her expertise in hardware and low-level programming. The tools and languages help same-skill sets to work across the entire applications for more efficient projects.

This was the background for which the .NET Micro Framework was developed. The beginnings of this platform were created as internal tools for developing Microsoft SPOT watch applications (only nine years ahead of the market). When the platform became useful for other internal projects, the product was rounded out as a general platform in 2007.

The embedded application drivers of cost, performance and power efficiency haven't gone away. If anything, when you're talking about applications where you want to deploy thousands of sensors in a building, cost becomes even more critical. Performance and power requirements vary with the application, as well. Continuing to improve the range of applications that can be reached with the .NET Micro Framework is an ongoing area of focus for Microsoft development efforts.

You can also deploy the app to an emulator running on a desktop PC. The difference is Windows Phone has a comparatively limited number of permutations, so you can select the closest emulator from a set provided in Visual Studio. For the .NET Micro Framework, there's a generic Microsoft emulator. You'll most likely have to create your own emulator to match your hardware. This is definitely worth looking into if you want to start coding before your hardware is available.

In the past, the scenario for debugging the communication between the robot and the Windows Phone app might have been done with a JTAG debugger. This is separate from the development environment and foreign to a .NET developer, so you might have one developer monitoring what's happening on the phone and one monitoring (in separate tools) what's happening on the embedded device. You can monitor both simultaneously in Visual Studio. You can step through the phone code and break in on command transmission, then step into what happens to that command on Bert.

Figure 9 **Interrupt Handler Activating Command Monitoring**

```
private void button_OnInterrupt(uint data1, uint data2, DateTime time)
{
  cmdListenState = !cmdListenState;
  if (cmdListenState)
  {
    // Set up timer for rangeFinders
    rangeReader = new Timer(new TimerCallback(rangeTimerHandler), null, 0, 100);
  }
  else
  {
    rangeReader.Dispose(); // Drop the timer because Bert isn't processing
                           // commands so he shouldn't be moving
  }
  led.Write(cmdListenState);  // The onboard LED indicates whether
                              // Bert is accepting commands
}
```

### Wrapping Up

The full picture of how to create any significant embedded application on the .NET Micro Framework would take an entire book. The objective of this article was to demonstrate the learning curve isn't very steep to use your .NET skills for embedded applications and the .NET Micro Framework.

This project requires a few classes that aren't part of the full .NET libraries. There will also be some things you're familiar with on the full .NET Framework that you couldn't squeeze into the .NET Micro Framework, but the Visual Studio environment is completely supported with IntelliSense, breakpoints, watch points, call stack, and whatever else you're used to. There are embedded applications out there for which the .NET Micro Framework isn't the right solution. For many, though, it works quite well. Efforts continue to broaden the applicability of the platform to larger segments of this growing and increasingly critical part of computer science.

So now you're ready to try a small project as an embedded programmer. ■

**Colin Miller** *is a principal program manager on the Internet of Things team at Microsoft. Reach him at colin.miller@microsoft.com. He's been at Microsoft for 20 years, and has worked on the Internet of Things, Embedded, Developer Tools, Research, MSN, Internet Explorer, Word and other projects.*

**Thanks** to the following Microsoft technical expert for reviewing this article: *Lorenzo Tessiore*

## 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

### 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™

**FREE Download**

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

# Go Cross-Platform with the .NET Framework

Cheryl Simmons

Most developers prefer to write their business logic code once and reuse it later. This is a much easier approach than building different apps to target multiple platforms. Recent announcements about .NET Core—a componentized version of the Microsoft .NET Framework—and the close partnership of Microsoft and Xamarin mean that if you create Portable Class Libraries (PCLs) compatible with .NET Core, you're closer to this reality than you've ever been.

What about your existing .NET Framework libraries, though? How much work will it take to make these compatible across platforms and convert them to PCLs? Enter the .NET Portability Analyzer. Using a couple of simple techniques and making some project file changes, this could help ease the process.

The .NET Portability Analyzer tool is a Visual Studio extension created by the .NET Framework team. You can use it with any recent version of Visual Studio that supports extensions. Simply point the Portability Analyzer at your assemblies or projects and the tool provides a summary, detailed report and recommendations about the APIs you should use to increase compatibility. For projects, the tool lists error messages, and takes you to the lines of code you need to change. The tool also provides results for key Microsoft platforms, and you can configure it to provide results for other platforms such as Mono and Xamarin.

This article discusses:

- Sharing Portable Class Libraries across platforms
- Using the .NET Portability Analyzer
- Identifying and mitigating compatibility issues

Technologies discussed:

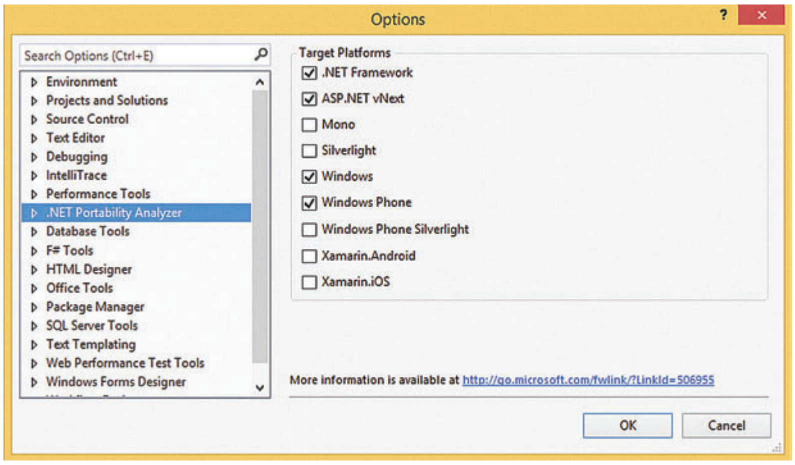
Microsoft .NET Framework, Visual Studio

The .NET Portability Analyzer has a console app sibling called the API Portability Analyzer (you can download it at aka.ms/w1vcle) that generates results similar to what the Portability Analyzer generates. For this article, I'll focus on using the Visual Studio extension. It's also important to note there might be some updates to the .NET Portability Analyzer extension between when this article was written and its publish date, so the appearance of the tool might differ from the images you see here.

## Get Set for Success

For a library to be successfully taken across platforms, it should be well factored and contain mostly business logic. The UI code should be separated into other projects. However, because .NET Core is a subset of the .NET Framework, even if your code is well factored, your libraries might be using APIs that aren't supported in .NET Core.

There are alternative APIs in some cases that accomplish the same thing. In these cases, the Portability Analyzer will suggest an alternative API. In other cases, there's no substitute and you'll need to factor out platform-specific code. Finally, even if you have no idea how well an assembly is factored, you can use Portability Analyzer to perform a quick assessment.



Target Platforms

- [✓] .NET Framework
- [✓] ASP.NET vNext
- [ ] Mono
- [ ] Silverlight
- [✓] Windows
- [✓] Windows Phone
- [ ] Windows Phone Silverlight
- [ ] Xamarin.Android
- [ ] Xamarin.iOS

More information is available at http://go.microsoft.com/fwlink/?LinkId=506955

Figure 1 **Select the Target Platforms for Your Project**

To use the extension, you'll need Visual Studio 2013 or 2014. The next step is to install the extension. You can find it by searching for .NET Portability Analyzer in the Visual Studio Gallery or go directly to aka.ms/lah74y.

Click the Download button and choose Open. The next dialog lets you choose the version of Visual Studio to which you want to apply the extension. Click Install, which starts the installation and then Close to dismiss the dialog. Now you're ready to choose your target platforms and analyze assemblies or projects.

## Choose Your Target Platforms

The Portability Analyzer provides results for the .NET Framework, ASP.NET vNext (aka .NET Core), Windows and Windows Phone, by default. You can specify additional options by accessing the .NET Portability Analyzer entry from the Tools | Options menu in Visual Studio and selecting the set of platforms you'd like to target, as shown in **Figure 1**.

## Run the Portability Analyzer

There are two ways you can analyze your assemblies and projects:

- To analyze already built assemblies or executable files, access the Portability Analyzer from the Analyze menu in Visual Studio and browse to the assembly location. With this option, the tool generates a summary and detailed report.
- To analyze projects, right-click the target project in Solution Explorer. Choose Analyze | Analyze Assembly Portability (see **Figure 2**), which is specific to the project you selected.

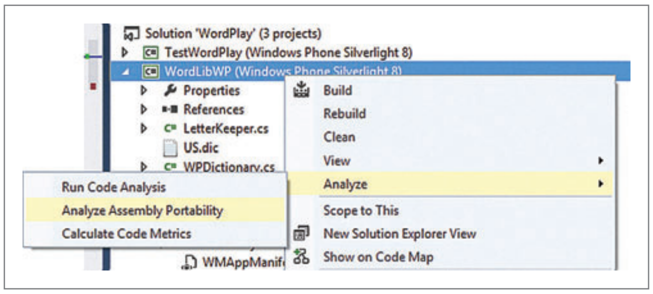

Solution 'WordPlay' (3 projects)
TestWordPlay (Windows Phone Silverlight 8)
WordLibWP (Windows Phone Silverlight 8)
Properties
References
LetterKeeper.cs
US.dic
WPDictionary.cs
WMAppManif
Build
Rebuild
Clean
View
Analyze
Scope to This
New Solution Explorer View
Show on Code Map
Run Code Analysis
Analyze Assembly Portability
Calculate Code Metrics

Figure 2 **Selecting Analyze Assembly Portability for a Specific Project**

With this option, the tool generates a summary, a detailed report, and outputs a message to the Error List that provides a file name and line number where the issue occurs. You can also double-click each message and the tool navigates to the specified line of code.

To test the tool, I opened a project I had worked on about a year ago—a word game that targets Windows Phone Silverlight 8. I started with my business logic in a Portable Class Library (PCL) targeting Windows Phone 8 and Windows 8. The idea was to reuse the library to implement the same app for Windows. However, I ran into some issues and was in a hurry, so I changed my approach and my library now targets only Windows Phone Silverlight 8.

My plan is to analyze the library's portability, make the necessary code changes, and then convert the project to a PCL project and retarget it at Windows 8.1, Windows Phone 8.1, and Windows Phone Silverlight 8.1. I also want to test its compatibility with .NET Core. The Portability Analyzer gives me a peek at the work I need to do without actually converting the project, changing the targets and trying to sort out the compile errors. I'm also curious to see if I can use my library to build an Android app, so I've configured the tool to provide results for Xamarin.Android, as well.

I run the tool and the results are encouraging. **Figure 3** shows the summary, detailed report, error messages and report URL. According to the summary, I can see my library is pretty compatible across all of the platforms. It's 100 percent compatible with Windows Phone Silverlight 8.1, which isn't surprising considering Windows Phone Silverlight 8 was the original target.

The .NET Portability Analyzer has a console app sibling called the API Portability Analyzer.

The detailed results are a spreadsheet-like display of only the APIs that aren't supported by one or more of my target platforms. The details are easy to scan. They're marked with a red X to indicate where an API isn't supported and a green checkmark to indicate support. It's important to note APIs supported across all of the platforms, and therefore not requiring any refactoring, won't be listed in this report.

The details also include a recommended changes column, which points me to alternative APIs that will work across multiple platforms. At the bottom of the details, the report includes a Back to Summary link. This navigates back to the summary at the top. My results are very short, but for longer reports, this "return to top" functionality is useful.

Because I've analyzed a project, my report contains Error List messages that indicate the file and line number where the usage occurs. If I click the message, the tool takes me to the file and line indicated by the message.

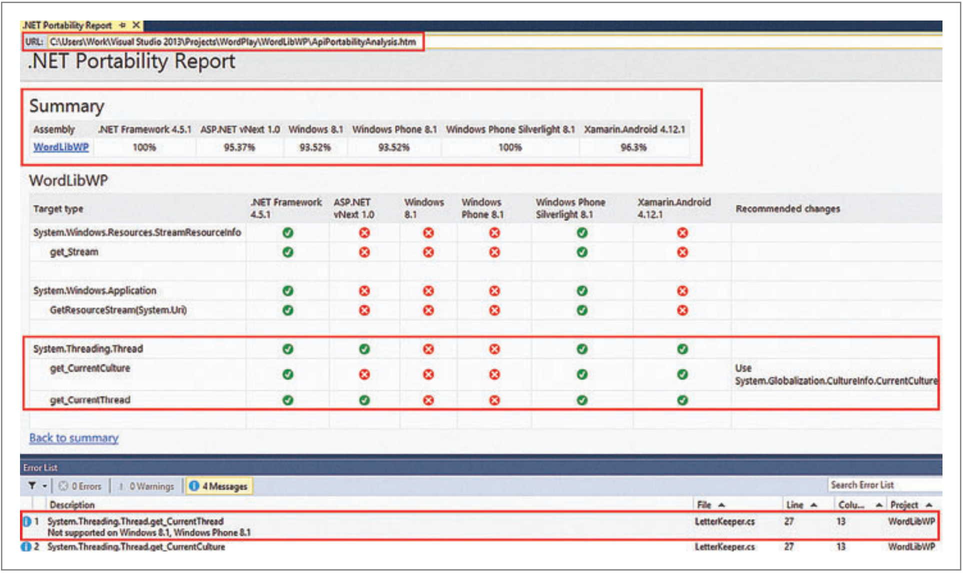
If you want to access the results outside Visual Studio, they're stored in an HTML file (ApiPortability-Analysis.htm) located in the same project directory as the target assembly. The location is indicated in the URL section at the top of the report, as shown in **Figure 4**.

## Work in Progress

The .NET Portability Analyzer, its results and guidance are expected to change as the .NET team gathers more information (see **Figure 5**). The team collects anonymous data about API usage when you use the tool or its console app counterpart and it's summarized at bit.ly/14vciaD.

This site defaults to show you the APIs that require the most code changes. You can change this by changing the value in the Show drop-down (not shown in the image). You can also hover over the "R" or "S" icons and see the same code recommendations shown by Portability Analyzer.

Each API name in the list is a link that takes you to a page reporting on which platforms that API is supported. The team plans to continue to update the site and will hopefully open it to customer contributions, so check back periodically.



.NET Portability Report

URL: C:\Users\Work\Visual Studio 2013\Projects\WordPlay\WordLibWP\ApiPortabilityAnalysis.htm

.NET Portability Report

Summary

| Assembly | .NET Framework 4.5.1 | ASP.NET vNext 1.0 | Windows 8.1 | Windows Phone 8.1 | Windows Phone Silverlight 8.1 | Xamarin.Android 4.12.1 |
|---|---|---|---|---|---|---|
| WordLibWP | 100% | 95.37% | 93.52% | 93.52% | 100% | 96.3% |

WordLibWP

| Target type | .NET Framework 4.5.1 | ASP.NET vNext 1.0 | Windows 8.1 | Windows Phone 8.1 | Windows Phone Silverlight 8.1 | Xamarin.Android 4.12.1 | Recommended changes |
|---|---|---|---|---|---|---|---|
| System.Windows.Resources.StreamResourceInfo | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | |
| get_Stream | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | |
| System.Windows.Application | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | |
| GetResourceStream(System.Uri) | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | |
| System.Threading.Thread | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | |
| get_CurrentCulture | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | Use System.Globalization.CultureInfo.CurrentCulture |
| get_CurrentThread | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | |

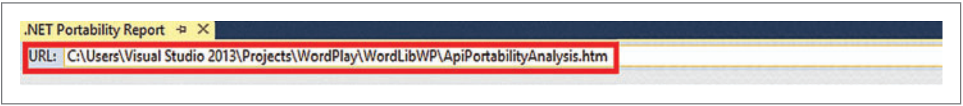
Back to summary

Error List

0 Errors | 0 Warnings | 4 Messages | Search Error List

| | Description | File | Line | Colu... | Project |
|---|---|---|---|---|---|
| 1 | System.Threading.Thread.get_CurrentThread Not supported on Windows 8.1, Windows Phone 8.1 | LetterKeeper.cs | 27 | 13 | WordLibWP |
| 2 | System.Threading.Thread.get_CurrentCulture | LetterKeeper.cs | 27 | 13 | WordLibWP |

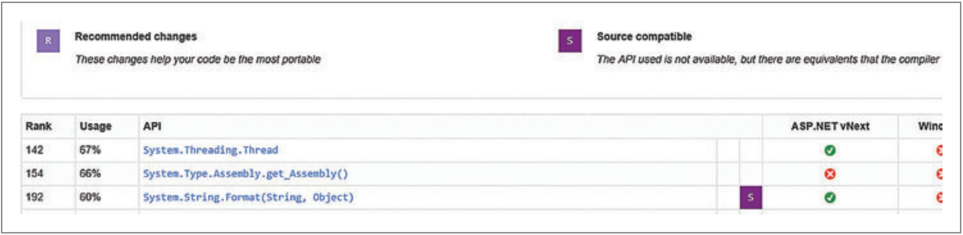
Figure 3 **The Detailed Compatibility Report Showing Compatible Platforms**



.NET Portability Report

URL: C:\Users\Visual Studio 2013\Projects\WordPlay\WordLibWP\ApiPortabilityAnalysis.htm

Figure 4 **Portability Analysis Results Stored for Access Outside Visual Studio**



R Recommended changes

*These changes help your code be the most portable*

S Source compatible

*The API used is not available, but there are equivalents that the compiler*

| Rank | Usage | API | | ASP.NET vNext | Winc |
|---|---|---|---|---|---|
| 142 | 67% | System.Threading.Thread | | ✓ | ✗ |
| 154 | 66% | System.Type.Assembly.get_Assembly() | | ✗ | ✗ |
| 192 | 60% | System.String.Format(String, Object) | S | ✓ | ✗ |

Figure 5 **This Site Shows .NET API Usage Levels**

## Take the Library Cross-Platform

Now it's time to fix my library. I know I want my library to work with Windows Phone 8.1 and Windows 8.1, and I see some issues. I'll take a different approach for each of the issues.

**Use Platform Abstraction** There are two resource stream-related entries. According to the report, these APIs aren't supported on Windows 8.1 or Windows Phone 8.1. The tool doesn't recommend any API I can substitute, so I need to remove this code from my library. The report tells me these two members are used in the same few lines of code, which load a static dictionary file:

```
WordList = new ObservableCollection<string>();
StreamResourceInfo info =
  Application.GetResourceStream(new
  Uri("/WordLibWP;component/US.dic", UriKind.Relative));
StreamReader reader = new StreamReader(info.Stream);
string line;

while ((line = reader.ReadLine()) != null)
{
  WordList.Add(line);
}
reader.close();
```

I know I can use the Windows.Storage APIs on Windows 8.1 and Windows Phone 8.1 to load a resource file, but then my library wouldn't be compatible with Windows Phone Silverlight. For the broadest reach, I decide to abstract this bit of platform-specific code. Platform abstraction is a useful technique for providing a library that defines behavior, but implementing that behavior differently in the platform-specific code. If you want your code to be compatible across platforms, it's a technique with which you should be familiar.

Here are the basic steps I took to perform this:

1. **Define the behavior in the cross-platform library:** To do this, I create an abstract class in the the library that defines a method for reading the dictionary file. I also defined a property that's an instance of my Dictionary-Reader class. Something like this:

```
public abstract class DictionaryReader
{
  public abstract Task<ObservableCollection<string>>
    ReadDictionaryAsync(string path);
  public static DictionaryReader Instance { get; set; }

}
```

2. **Implement the behavior in the platform-specific code:** To do this, I derive from the DictionaryReader class in the Windows Phone Silverlight project. I provide an implementation of the ReadDictionaryAsync method that loads and reads the dictionary file as an app resource. Notice

# Finally!

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

Edit and create MS Word documents

Create and modify Adobe® PDF documents

Create reports and mail merge templates

Integrate with Microsoft® Visual Studio

Try it online and download your 30-day trial version today:

www.textcontrol.com/html5

Visual Studio Partner

HTML 5

TEXT CONTROL

Figure 6 **Phone Implementation to Load and Read Dictionary**

```
class DictionaryReaderPhone : DictionaryReader
{
  public override async Task<ObservableCollection<string>>
  ReadDictionaryAsync(string resourcePath)
{
  ObservableCollection<string> wordList =
    new ObservableCollection<string>();
  StreamResourceInfo info =
    Application.GetResourceStream(new Uri(resourcePath,
    UriKind.Relative));
  if (info == null)
    throw new ArgumentException("Could not load resource: " +
      resourcePath +
      ". For Windows Phone this should be in the
      Format:[project];component/[filename]");
  else
  {
    StreamReader reader = new StreamReader(info.Stream);
    string line;

    while ((line = await reader.ReadLineAsync()) != null)
    {
    wordList.Add(line);
    }
    reader.Close();
    return wordList;
    }
  }
}
```

this code is essentially the same as the code in my library, with some error checking on the resource path as I need a specific format for my phone code to work. My implementation for other platforms will be different depending on the techniques for reading an app-local resource for those platforms. However, with the abstraction I've added—as shown in **Figure 6**—this shouldn't be an issue.

3. **Initialize the library-defined instance to the platform-specific implementation:** To do this, I add code to the App class for my Windows Phone project. This code initializes the instance of the DictionaryReader to my phone-specific implementation that reads the file as a resource. Again, this code is in my platform-specific project, and not in the project I analyzed:

```
public App()
{
  DictionaryReader.Instance = new DictionaryReaderPhone();
...
}
```

For another example that shows how to abstract out platform-specific code, see the MSDN Library article, "Platform Abstraction with the Portable Class Library," at aka.ms/q4sq7l.

**Use a Different API** Next, I evaluate the entry for getting the name of the current culture. I double-click the error message and find this method:

```
public string CheckLanguage()
{
  return Thread.CurrentThread.CurrentCulture.Name.Split('-')[0];
}
```

According to Portability Analyzer, I can use CultureInfo.CurrentCulture. Because CurrentCulture is a static property of CultureInfo and provides the same information I was getting from the current thread, it should work just fine. I replace the code that relies on getting Thread class with the following code, which uses CultureInfo:

```
public string CheckLanguage()
{
  return System.Globalization.CultureInfo.CurrentCulture.Name.Split('-')
[0];
}
```

## So Far, So Good

I test my changes and everything looks good. Next, I rerun Portability Analyzer. My report is clean, with only a few code changes, and I've picked up support for Xamarin.Android in addition to my original platform targets, as shown in **Figure 7**.

The final step for my cross-platform conversion is to convert the library project from a phone-specific library project to a PCL I can reference from multiple apps. Initially, I thought the easiest and quickest way to do this would be to do an "in-place" conversion by manually modifying the project file.

After doing some research and trying some of the steps I found, I came to the conclusion there are no discernible advantages to modifying the project file manually. I double-checked with a member of the .NET team and he agreed. Much of the help you'll find for manually modifying a project assumes a certain starting point.

Depending on your project history and any other upgrades you've done, you could end up with a broken project. It's likely you'll need to create a new PCL project after spending time trying to manually convert it. Also, even if your manually converted project works initially, it could break if you make other changes later.

As a result, I created a new PCL project, and recommend you do the same. I copied my code files over to the new project. Once I copied the code, I got a few compile errors because of using statements that no longer applied to the new PCL. I removed these and my library was ready to go. I switched my Windows Phone Silverlight app to reference my new PCL and my app was up and running again.

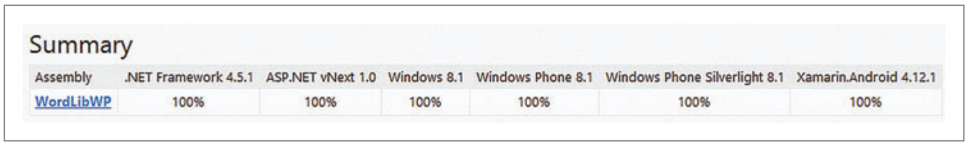
## Try It Out

Using Portability Analyzer not only helped me quickly evaluate the work I needed to do to make my library cross-platform, but identified any platform-specific issues within my code down to method calls and property usage. It also made API suggestions where alternatives were available.

I've also shown you techniques for factoring out platform-specific code. My new Portability Analyzer results indicate my library will run on the additional platforms I wanted to target, as well as Xamarin.Android. Finally, I was able to create a new PCL with the new targets and reference this PCL from my existing app. The new .NET Portability Analyzer will help you take your libraries to new platforms. ■

**Cheryl Simmons** *has been a programming writer at Microsoft for 10 years and has written about the .NET Base Class Libraries, Windows Forms, Windows Presentation Foundation, Silverlight, Windows Phone and related tools.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Matt Galbraith, Daniel Plaisted and Taylor Southwick*



Summary

| Assembly | .NET Framework 4.5.1 | ASP.NET vNext 1.0 | Windows 8.1 | Windows Phone 8.1 | Windows Phone Silverlight 8.1 | Xamarin.Android 4.12.1 |
|---|---|---|---|---|---|---|
| WordLibWP | 100% | 100% | 100% | 100% | 100% | 100% |

Figure 7 **Rerun the Analysis Report After Code Changes**

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com
NAVIGATE THE .NET HIGHWAY

## Join us on the Ultimate Code Trip in 2015!

### Las Vegas
**MARCH 16-20**
BALLY'S HOTEL & CASINO
**vslive.com/lasvegas**
See pages 54 – 55 for more info

### Austin
**JUNE 1-4**
HYATT REGENCY
**vslive.com/austin**
See pages 42 – 43 for more info

### San Francisco
**JUNE 15-18**
THE FAIRMONT
**vslive.com/sf**
See pages 44 – 45 for more info

### Redmond
**AUGUST 10-14**
MICROSOFT HEADQUARTERS
**vslive.com/redmond**
See pages 62 – 63 for more info

### New York
**SEPTEMBER 28 - OCTOBER 1**
NY MARRIOTT AT BROOKLYN BRIDGE

### Orlando
**NOVEMBER 16 - 20**
LOEWS ROYAL PACIFIC RESORT

**CONNECT WITH VISUAL STUDIO LIVE!**

twitter.com/vslive – @VSLive
facebook.com – Search "VSLive"
linkedin.com – Join the "Visual Studio Live" group!

TURN THE PAGE FOR MORE EVENT DETAILS.

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/austin
**Austin** JUNE 1-4
HYATT REGENCY, AUSTIN, TX

## DON'T MESS WITH CODE

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

**DEVELOPMENT TRACKS INCLUDE:**

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

**Register NOW and Save $300!**

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

**Use promo code VAUMAR2**

**vslive.com/austin**

GOLD SPONSOR
GitHub

SUPPORTED BY
Microsoft
Visual Studio
msdn magazine
Visual Studio MAGAZINE

PRODUCED BY
1105 MEDIA INC

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Austin AGENDA AT-A-GLANCE

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

## Visual Studio Live! Pre-Conference Workshops: Monday, June 1, 2015 (Separate entry fee required)

| START TIME | END TIME | | | |
|---|---|---|---|---|
| 7:30 AM | 9:00 AM | Pre-Conference Workshop Registration — Coffee and Morning Pastries | | |
| 9:00 AM | 6:00 PM | **M01 Workshop: SQL Server 2014 for Developers** - *Leonard Lobel* | **M02 Workshop: Hybrid Mobile Apps with Visual Studio, Cordova, Angular, and Azure** - *Brian Noyes* | **M03 Workshop: ALM and DevOps with the Microsoft Stack** - *Brian Randell* |
| 6:45 PM | 9:00 PM | Dine-A-Round with Speakers | | |

## Visual Studio Live! Day 1: Tuesday, June 2, 2015

| START TIME | END TIME | | | | |
|---|---|---|---|---|---|
| 7:00 AM | 8:00 AM | Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 9:00 AM | **KEYNOTE:** To Be Announced - *Jay Schmelzer, Director of Program Management, Visual Studio Team, Microsoft* | | | |
| 9:15 AM | 10:30 AM | **T01 Building Mobile Cross-Platform Apps with C# and Xamarin** - *Nick Landry* | **T02 UX Design Principle Fundamentals for Non-Designers** - *Billy Hollis* | **T03 Azure 10-10: Top 10 Azure Announcements in "T-10" Months** - *Vishwas Lele* | **T04 A Lap Around Visual Studio 2015** - *Robert Green* |
| 10:45 AM | 12:00 PM | **T05 Building Mobile Cross-Platform Apps in C# with Azure Mobile Services** - *Nick Landry* | **T06 Designing and Building UX for Finding and Visualizing Data in XAML Applications** - *Billy Hollis* | **T07 Cloud or Not, 10 Reasons Why You Must Know "Websites"** - *Vishwas Lele* | **T08 Putting CodedUI Tests on Steroids** - *Donovan Brown* |
| 12:00 PM | 1:30 PM | Lunch | | | |
| 1:30 PM | 2:45 PM | **T09 ASP.NET 5 in All Its Glory** - *Adam Tuliper* | **T10 Getting Started with Universal Apps for Windows and Windows Phone** - *Philip Japikse* | **T11 Microsoft Azure SQL Database: SQL Server in the Cloud** - *Leonard Lobel* | **T12 Moving DevOps to the Cloud Using Visual Studio Online, Release Management Online, and Azure** - *Donovan Brown* |
| 3:00 PM | 4:15 PM | **T13 Hack Proofing Your Web Applications** - *Adam Tuliper* | **T14 Building Windows 10 LOB Apps** - *Robert Green* | **T15 Database Development with SQL Server Data Tools** - *Leonard Lobel* | **T16 What's New in ALM** - *Brian Randell* |
| 4:15 PM | 5:30 PM | Welcome Reception | | | |

## Visual Studio Live! Day 2: Wednesday, June 3, 2015

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

## Visual Studio Live! Day 3: Thursday, June 4, 2015

| START TIME | END TIME | | | | |
|---|---|---|---|---|---|
| 7:00 AM | 8:00 AM | Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 9:15 AM | **TH01 Automated UI Testing for Android and iOS Mobile Apps** - *James Montemagno* | **TH02 Busy JavaScript Developer's Guide to ECMAScript 6** - *Ted Neward* | **TH03 SQL 2014 Columnstore Indexes and In-Memory Optimized Tables** - *Kevin Goff* | **TH04 What's New in C# 6.0** - *Jason Bock* |
| 9:30 AM | 10:45 AM | **TH05 You Too Can Easily Create an Amazing 3D Application with Unity** - *Adam Tuliper* | **TH06 Building Web APIs to Support Your HTML/JavaScript Client** - *Brian Noyes* | **TH07 SQL Server Reporting Services - Attendees Choose Topics** - *Kevin Goff* | **TH08 Managing the .NET Compiler** - *Jason Bock* |
| 11:00 AM | 12:15 PM | **TH09 Strike Up A Conversation with Cortana on Windows Phone** - *Walt Ritscher* | **TH10 To Be Announced** | **TH11 Build Real-Time Websites and Apps with SignalR** - *Rachel Appel* | **TH12 Busy Developer's Guide to NoSQL** - *Ted Neward* |
| 12:15 PM | 1:30 PM | Lunch | | | |
| 1:30 PM | 2:45 PM | **TH13 Windows, NUI, and You** - *Brian Randell* | **TH14 Hate JavaScript? Try TypeScript** - *Ben Hoelting* | **TH15 ASP.NET MVC: All Your Tests Are Belong To Us** - *Rachel Appel* | **TH16 Microsoft's .NET is Now Open Source and Cross-Platform. Why It Matters.** - *Mark Rosenberg* |
| 3:00 PM | 4:15 PM | **TH17 WPF Data Binding in Depth** - *Brian Noyes* | **TH18 JavaScript Patterns for the C# Developer** - *Ben Hoelting* | **TH19 Building Rich UI with ASP.NET MVC and Bootstrap** - *Walt Ritscher* | **TH20 Async and Await Best Practices** - *Mark Rosenberg* |

*Sessions and speakers subject to change.*

vslive.com/austin

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/sf

San Francisco JUNE 15-18

THE FAIRMONT, SAN FRANCISCO, CA

# CODE BY THE BAY

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

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

**DEVELOPMENT TRACKS INCLUDE:**

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

**REGISTER NOW AND SAVE $300!**

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

Use promo code VSFMAR2

**vslive.com/sf**

SUPPORTED BY

Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

# San Francisco AGENDA AT-A-GLANCE

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

| START TIME | END TIME | Visual Studio Live! Pre-Conference Workshops: Monday, March 16, 2015 (Separate entry fee required) | | |
|---|---|---|---|---|
| 7:30 AM | 9:00 AM | Pre-Conference Workshop Registration — Coffee and Morning Pastries | | |
| 9:00 AM | 6:00 PM | **M01 Workshop: Big Data, Analytics and NoSQL: Everything You Wanted to Learn But Were Afraid to Ask** - *Andrew Brust* | **M02 Workshop: Native Mobile App Development for iOS, Android and Windows Using C#** - *Marcel de Vries* | **M03 Workshop: ALM and DevOps with the Microsoft Stack** - *Brian Randell* |
| 6:45 PM | 9:00 PM | Dine-A-Round with Speakers | | |

| START TIME | END TIME | Visual Studio Live! Day 1: Tuesday, June 16, 2015 | | | |
|---|---|---|---|---|---|
| 7:00 AM | 8:00 AM | Registration — Coffee and Morning Pastries | | | |
| 8:00 AM | 9:00 AM | **KEYNOTE:** To Be Announced | | | |
| 9:15 AM | 10:30 AM | **T01 Azure 10-10: Top 10 Azure Announcements in "T-10" Months** - *Vishwas Lele* | **T02 AngularJS 101** - *Deborah Kurata* | **T03 UX Design Principle Fundamentals for Non-Designers** - *Billy Hollis* | **T04 A Lap Around Visual Studio 2015** - *Robert Green* |
| 10:45 AM | 12:00 PM | **T05 Cloud or Not, 10 Reasons Why You Must Know "Web Sites"** - *Vishwas Lele* | **T06 AngularJS Forms and Validation** - *Deborah Kurata* | **T07 Designing and Building UX for Finding and Visualizing Data in XAML Applications** - *Billy Hollis* | **T08 To Be Announced** |
| 12:00 PM | 1:30 PM | Lunch — Visit Exhibitors | | | |
| 1:30 PM | 2:45 PM | **T09 Building Mobile Cross-Platform Apps with C# and Xamarin** - *Nick Landry* | **T10 ASP.NET 5 in all its Glory** - *Adam Tuliper* | **T11 Getting Started with Universal Apps for Windows and Windows Phone** - *Philip Japikse* | **T12 To Be Announced** |
| 3:00 PM | 4:15 PM | **T13 Building Mobile Cross-Platform Apps in C# with Azure Mobile Services** - *Nick Landry* | **T14 Hack Proofing Your Web Applications** - *Adam Tuliper* | **T15 Building Windows 10 LOB Apps** - *Robert Green* | **T16 What's New in ALM** - *Brian Randell* |
| 4:15 PM | 5:30 PM | Welcome Reception | | | |

| START TIME | END TIME | Visual Studio Live! Day 2: Wednesday, June 17, 2015 | | | |
|---|---|---|---|---|---|
| 7:00 AM | 8:00 AM | Registration — Coffee and Morning Pastries | | | |
| 8:00 AM | 9:00 AM | **KEYNOTE: To Be Announced** - *Kris Lankford, Sr. Product Manager, Microsoft* | | | |
| 9:15 AM | 10:30 AM | **W01 iOS Native Development** - *Jon Flanders* | **W02 Implementing M-V-VM (Model-View-View Model) for WPF** - *Philip Japikse* | **W03 Moving Web Apps to the Cloud** - *Eric D. Boyd* | **W04 Stop the Waste and Get Out of (Technical) Debt** - *Richard Hundhausen* |
| 10:45 AM | 12:00 PM | **W05 Swift for .NET Developers** - *Jon Flanders* | **W06 Strike Up a Conversation with Cortana on Windows Phone** - *Walt Ritscher* | **W07 Solving Security and Compliance Challenges with Hybrid Clouds** - *Eric D. Boyd* | **W08 Best Practices for Using Open Source Software in the Enterprise** - *Marcel de Vries* |
| 12:00 PM | 1:30 PM | Birds-of-a-Feather Lunch — Visit Exhibitors | | | |
| 1:30 PM | 2:45 PM | **W09 Building Cross Platform UI with Xamarin.Forms** - *Walt Ritscher* | **W10 Take a Gulp, Make a Grunt, and Call Me Bower** - *Adam Tuliper* | **W11 To Be Announced** | **W12 Automated Build, Test & Deploy with TFS, ASP.NET, and SQL Server** - *Benjamin Day* |
| 3:00 PM | 4:15 PM | **W13 Adding Analytics to Your Mobile Apps on Any Platform with Microsoft Application Insights and Hockeyapp** - *Marcel de Vries* | **W14 Securing Angular Apps** - *Brian Noyes* | **W15 Getting Up & Running with Docker on Microsoft Azure - Rick Garibay** | **W16 Professional Scrum Development Using Visual Studio 2015** - *Richard Hundhausen* |
| 4:30 PM | 5:45 PM | **W17 Creating Applications Using Android Studio** - *Kevin Ford* | **W18 - Build Data-Centric HTML5 Single Page Applications with Breeze** - *Brian Noyes* | **W19 A Pragmatic Reference Architecture for The Internet of Things** - *Rick Garibay* | **W20 Load Testing ASP.NET & WebAPI with Visual Studio** - *Benjamin Day* |
| 7:00 PM | 9:00 PM | Visual Studio Live! Evening Event | | | |

| START TIME | END TIME | Visual Studio Live! Day 3: Thursday, June 18, 2015 | | | |
|---|---|---|---|---|---|
| 7:00 AM | 8:00 AM | Registration — Coffee and Morning Pastries | | | |
| 8:00 AM | 9:15 AM | **TH01 Using Multi-Device Hybrid Apps to Create Cordova Applications** - *Kevin Ford* | **TH02 Build Real-Time Websites and Apps with SignalR** - *Rachel Appel* | **TH03 Windows, NUI, and You** - *Brian Randell* | **TH04 To Git or Not to Git for Enterprise Development** - *Benjamin Day* |
| 9:30 AM | 10:45 AM | **TH05 Everything You Always Wanted To Know About REST (But Were Afraid To Ask)** - *Jon Flanders* | **TH06 Knocking it Out of the Park with Knockout.JS** - *Miguel Castro* | **TH07 Building Rich Data Web APIs with ASP.NET OData** - *Brian Noyes* | **TH08 What's New in C# 6.0** - *Jason Bock* |
| 11:00 AM | 12:15 PM | **TH09 Comparing Performance of Different Mobile Platforms** - *Kevin Ford* | **TH10 To Be Announced** | **TH11 Power BI 2.0: Analytics in the Cloud and in Excel** - *Andrew Brust* | **TH12 Microsoft's .NET is Now Open Source and Cross-Platform. Why it Matters.** - *Mark Rosenberg* |
| 12:15 PM | 1:30 PM | Lunch | | | |
| 1:30 PM | 2:45 PM | **TH13 Programming WPF with MVVM - Advanced Topics** - *Miguel Castro* | **TH14 Busy JavaScript Developer's Guide to ECMAScript 6** - *Ted Neward* | **TH15 Big Data and Hadoop with Azure HDInsight** - *Andrew Brust* | **TH16 Managing the .NET Compiler** - *Jason Bock* |
| 3:00 PM | 4:15 PM | **TH17 Extending XAML To Overcome Pretty Much Any Limitation** - *Miguel Castro* | **TH18 ASP.NET MVC: All Your Tests Are Belong To Us** - *Rachel Appel* | **TH19 Busy Developer's Guide to NoSQL** - *Ted Neward* | **TH20 Async and Await Best Practices** - *Mark Rosenberg* |

*Sessions and speakers subject to change.*

**vslive.com/sf**

# A Web Game in an Hour

Michael Oneppo

You don't need an entirely new skill set to develop games. In fact, your current Web development skills in HTML, JavaScript, CSS and so on are just fine for a wide range of games. When you build a game with Web technologies, it will run on pretty much any device with a browser.

To prove this, I'll demonstrate building a game from scratch using Web technologies and just two external libraries, and I'll do it in less than one hour. I'll cover a variety of game development topics, from basic design and layout, controls and sprites, to artificial intelligence (AI) for a simple opponent. I'm even going to develop the game so it works on PCs, tablets and smartphones. If you have some experience with programming as a Web developer or another development domain, but no experience writing games, this article will get you started. If you give me one hour, I promise to show you the ropes.

## Get Up and Running

I'll do all development in Visual Studio, which will allow fast execution of the Web app as I make changes. Be sure to have the latest version of Visual Studio (download at bit.ly/1xEjEnX) so you can follow along. I used Visual Studio 2013 Pro, but updated the code with Visual Studio 2013 Community.

This app will require no server code, so I start by creating a new, empty Web page project in Visual Studio. I'll use the empty C# template for a Web site by selecting the Visual C# option after selecting File | New | ASP.NET Empty Web Site.

The index HTML file requires just three resources: jQuery, a main style sheet and a main JavaScript file. I add an empty CSS file to the project called style.css and an empty JavaScript file called ping.js to avoid errors when loading the page:

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js"></script>
  <script src="ping.js"></script>
  <link rel="stylesheet" href="style.css"></script>
</head>
<body>
</body>
</html>
```

## Basic Design

The game I'm building is a variant of Pong that I call Ping. Ping has essentially the same rules as Pong, except that either player grabs the ball when it comes to them and can then fire the ball back either directly or at an angle up or down. It's often best to draw how you would like the game to look before you build it. For this game, the overall layout I want to see is shown in **Figure 1**.

Once I've developed the game design layout, it's just a matter of adding each element to HTML to build the game. One thing to note, though, is I'll group the scoreboard and controls to ensure they sit together. So one by one, you can see I've added the elements, as shown in **Figure 2**.

This article discusses:

- Basic game development philosophy
- Using Web technologies for game development
- Adding game controls and AI

Technologies discussed:

Visual Studio 2013 Pro, Visual Studio 2013 Community, ASP.NET

Code download available at:

msdn.microsoft.com/magazine/msdnmag0315

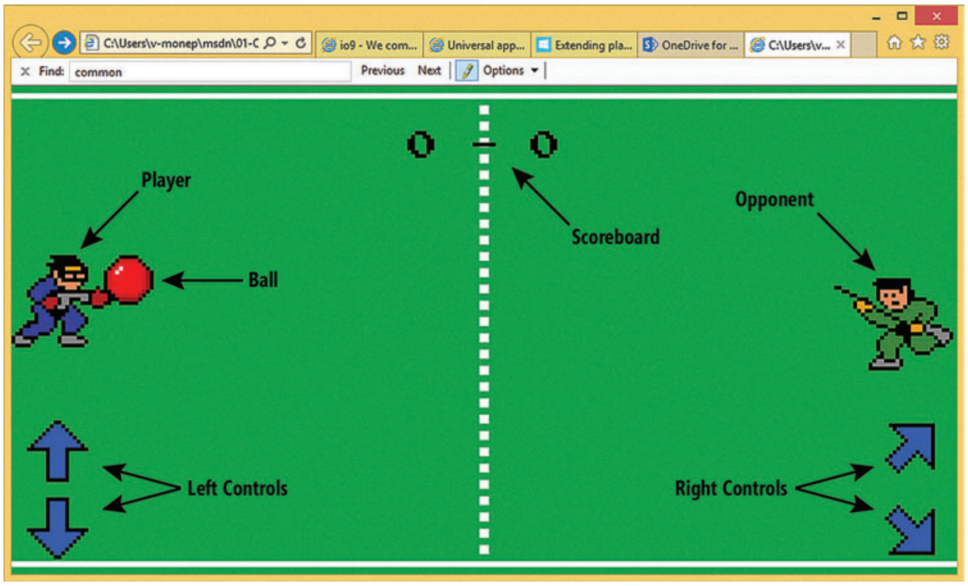


Figure 1 **The Overall Design of Ping**

## Play with Style

If you were to load this page, you wouldn't see anything because there's no style applied. I've already set up a link to a main.css file in my HTML, so I'll place all my CSS in a new file with that name. The first thing I'll do is position everything on the screen. The body of the page needs to take up the whole screen, so I'll set that up first:

```
body {
  margin: 0px;
  height: 100%;
}
```

Second, I need to have the arena fill the whole screen with the arena background image (see **Figure 3**) applied:

```
#arena {
  background-image: url(arena.png);
  background-size: 100% 100%;
  margin: 0px;
  width: 100%;
  height: 100%;
  overflow: hidden;
}
```

Next, I'll position the scoreboard. I want this to appear top and center, over the other elements. The command position: absolute lets me place it wherever I want and left: 50% places it halfway across the top of the window, but starting at the leftmost side of the scoreboard element. To ensure it's perfectly centered, I use the transform property and the z-index property ensures it's always at the top:

```
#score {
  position: absolute;
  z-index: 1000;
  left: 50%;
  top: 5%;
  transform: translate(-50%, 0%);
}
```

I also want the text font to be retro-themed. Most modern browsers let me include my own fonts. I found the appropriate Press Start 2P font from codeman38 (zone38.net). To add the font to the scoreboard, I have to create a new font face:

```
@font-face {
  font-family: 'PressStart2P';
  src: url('PressStart2P.woff');
}
```

Now, the scores are in an h1 tag, so I can set the font for all h1 tags. Just in case the font is missing, I'll provide a few backup options:

```
h1 {
  font-family: 'PressStart2P', 'Georgia', serif;
}
```

For the other elements, I'll use a sprite sheet of images. A sprite sheet contains all the images I need for the game in one file (see **Figure 4**).

It's often best to draw how you would like the game to look before you build it.

Any element that has an image on this sheet will have a sprite class assigned. Then, for each element, I'll use background-position to define what part of the sprite sheet I want to show:

```
.sprite {
  background-image: url("sprites.png");
  width: 128px;
  height: 128px;
}
```

Next, I'll add the sprite class to all elements that will use the sprite sheet. I'll have to briefly jump back to HTML to do this:

```
<div id="player" class="sprite"></div>
<div id="opponent" class="sprite"></div>
<div id="ball" class="sprite"></div>
<div id="controls-left">
  <div id="up" class="sprite"></div>
  <div id="down" class="sprite"></div>
</div>
<div id="controls-right">
  <div id="left" class="sprite"></div>
  <div id="right" class="sprite"></div>
</div>
```

Now I need to indicate the positions of each sprite on the sheet for each element. Again, I'll do this using background-position, as shown in **Figure 5**.

Figure 2 **The Initial HTML Layout**

```
<div id="arena">
  <div id="score">
    <h1>
      <span id="playerScore">0</span>
      -
      <span id="opponentScore">0</span>
    </h1>
  </div>
  <div id="player"></div>
  <div id="opponent"></div>
  <div id="ball"></div>
  <div id="controls-left">
    <div id="up"></div>
    <div id="down"></div>
  </div>
  <div id="controls-right">
    <div id="left"></div>
    <div id="right"></div>
  </div>
</div>
```

The position: absolute property on the player, opponent and ball will let me move them around using JavaScript. If you look at the page now, you'll see the controls and the ball have unnecessary pieces attached to them. This is because the sprite sizes are smaller than the default 128 pixels, so I'll adjust these to the right size. There's only one ball, so I'll set its size directly:

```
#ball {
  position: absolute;
  width: 64px;
  height: 64px;
  background-position: 128px 128px;
}
```

There are four control elements (buttons the user can press to move the player about), so it behooves me to make a special class for them. I'll also add a margin so they have a little space around them:

```
.control {
  margin: 16px;
  width: 64px;
  height: 64px;
}
```

After adding this class, the game has much better looking controls:

```
<div id="controls-left">
  <div id="up" class="sprite control"></div>
  <div id="down" class="sprite control"></div>
</div>
<div id="controls-right">
  <div id="left" class="sprite control"></div>
  <div id="right" class="sprite control"></div>
</div>
```

The last thing I need to do is position the controls so they're by the user's thumbs when the page is running on a mobile device. I'll stick them to the bottom corners:

```
#controls-left {
  position: absolute;
  left: 0; bottom: 0;
}

#controls-right {
  position: absolute;
  right: 0; bottom: 0;
}
```

One nice thing about this design is everything is set with relative positions. This means the screen can be a number of different sizes while still making the game look good.

## Follow the Bouncing Ball

Now I'll make the ball move around. For the JavaScript code, I've referenced a file called ping.js in HTML, just as I did with the CSS.

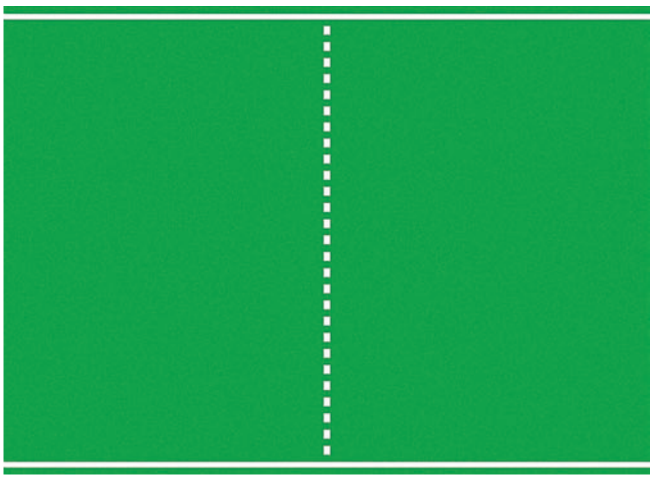


Figure 3 **The Arena Background Image**

I'll add this code to a new file with that name. I'm going to make objects for the ball and each of the players, but I'll use the factory pattern for the objects.

This is a simple concept. The Ball function creates a new ball when you call it. There's no need to use the new keyword. This pattern reduces some of the confusion around the this variable by clarifying the available object properties. And because I only have an hour to make this game, I need to minimize any confusing concepts.

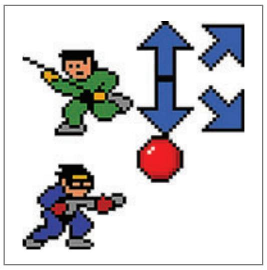


Figure 4 **The Sprite Sheet for Ping**

The structure of this pattern, as I make the simple Ball class, is shown in **Figure 6**.

To create a new ball, I simply call this function I've defined:

```
var ball = Ball();
```

Now I want to make the ball move and bounce around the screen. First, I need to call the update function at some interval to create an animation of the ball. Modern browsers provide a function meant for this purpose called requestAnimationFrame. This takes a function as an argument, and will call that passed-in function the next time it runs its animation cycle. This lets the ball move around in smooth steps when the browser is ready for an update. When it calls the passed-in function, it will give it the time in seconds since the page was loaded. This is critical for ensuring animations are consistent over time. In the game, the use of requestAnimationFrame appears as follows:

```
var lastUpdate = 0;
var ball = Ball();

function update(time) {
  var t = time - lastUpdate;
  lastUpdate = time;
  ball.update(t);
  requestAnimationFrame(update);
}

requestAnimationFrame(update);
```

Figure 5 **Adding Offsets for the Sprite Sheet**

```
#player {
  position: absolute;
  background-position: 0px 128px;
}

#opponent {
  position: absolute;
  background-position: 0px 0px;
}

#ball {
  position: absolute;
  background-position: 128px 128px;
}

#right {
  background-position: 64px 192px;
}

#left {
  background-position: 64px 0px;
}

#down {
  background-position: 128px 192px;
}

#up {
  background-position: 128px 0px;
}
```

```
40, 0); AddEAN13Sup5(page.Elements, 480, 0); AddUPCVersionA(page.Elements, 0, 135); AddUPCVersionASup2(page.Elements, 240, 135); AddUPCVersionASup5(page.Elements, 480, 135); AddEAN8(page.Elemen
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); BarCode barCode = new Ean13Sup5("12345678901212345", x, y + 21); barCode.X += (204 - barCode.
PCVersionA(Group elements, float x, float y) { AddCaptionAndRectangle(elements, "UPC Version A Bar Code", x, y, 204, 99); BarCode barCode = new UpcVersionA("12345678901", x, y + 21); barCode.X += (204 - ba
PCVersionASup2(Group elements, float x, float y) { AddCaptionAndRectangle(elements, "UPC Version E Bar Code, 2 digit supplement", x, y, 204, 99); BarCode barCode = new UpcVersionASup2("1234567890112", x,
dd(barCode); } private void AddUPCVersionASup5(Group elements, float x, float y) { AddCaptionAndRectangle(elements, "UPC Version E Bar Code, 5 digit supplement", x, y, 204, 99); BarCode barCode = new UpcVe
GetSymbolWidth()) / 2; elements.Add(barCode); } private void AddEAN8(Group elements, float x, float y) { AddCaptionAndRectangle(elements, "EAN/JAN 8 Bar Code", x, y, 204, 99); BarCode barCode = new Ean8("12
fileDialog.Title = "Open File Dialog";  fileDialog.Filter = "Adobe PDF files (*.pdf)|*.pdf|All Files (*.*)|*.*";  if (fileDialog.ShowDialog() == DialogResult.OK)  {  pdfViewer.OpenFile(fileDialog.FileName, "");  }  SaveFileDialog sa
e Dialog";  saveFileDialog.Filter = "Adobe PDF files (*.pdf)|*.pdf|All Files (*.*)|*.*";  if (saveFileDialog.ShowDialog()==DialogResult public ViewerForm().OK)  {  pdfViewer.SaveAs(saveFileDialog.FileName);  }  if (pdfViewer.P
Dialog();  }  else  {  MessageBox.Show("Please open a file to print");  }   OpenFileDialog fileDialog = new OpenFileDialog();   file{ Dialog.Title = "Open File Dialog";  fileDialog.InitialDirectory = @"c:\";  fileDialog.Filter = "
DialogResult.OK)  {  DynamicPDFViewerClass test = new DynamicPDFViewerClass();  PDFPrinter printer = test.OpenFileForPrinter// Convert RTF file to PDF (fileDialog.FileName);  printer.PrintQuiet();  }  byte[] content
rvices;  GCHandle gch = GCHandle.Alloc(contents, GCHandleType.Pinned);   IntPtr contentsIntPtr =gch.AddrOfPinnedObject();pdf Converter.Convert(@"C:\Input.rtf", @"C:\Ouptut.pdf"); Viewer.OpenBuffer(contentsInt
ark Page Element:", x, y); pageElements.Add(new Bookmark("Bookmarked Text", x + 5, y + 20, parentOutline)); pageElements.Add  // Output PDF to Custom Viewer (new Label("This text is bookmarked.", x + 5, y + 2
Elements, float x, float y) { // Adds a circlt to the pageElements AddCaptionAndRectangle(pageElements, "Circle Page Element:", x, pdfViewer.OpenFile(@"C:\Ouptut.pdf", ""); y); pageElements.Add(new Circle(x + 112
arge)); } private void AddFormattedTextArea(Group pageElements, float x, float y) { // Adds a formatted text area to thepageEle}ments string formattedHtml = "<p><i>Dynamic</i><b>PDF</b>&tm; Generator v6.0 for
tting support for text that appears in the document. You have " + "complete control over 8 paragraph properties: spacing before, spacing after, first line " + "indentation, left indentation, right indentation, alignment, all
nt face='Times'>font face, </font><font pointSize='6'>font " + "size, </font><font color='FF0000'>color, </font><b>bold, </b><i>italic and </i><u>underline</u>; " + "and 2 line properties: leading, and leading type.
rea = new FormattedTextArea(formattedHtml, x + 5, y + 20, 215, 60, FontFamily.Helvetica, 9, false); // Sets the indent property formattedTextArea.Style.Paragraph.Indent = 18; AddCaptionAndRectangle(pageElement
Elements, "FormattedTextArea Overflow Text:", x + 279, y); pageElements.Add(formattedTextArea); // Create an overflow formatted text area for the overflow text FormattedTextArea overflowFormattedTextArea = form
+ 284, y + 20); pageElements.Add(overflowFormattedTextArea); } private void AddImage(Group pageElements, float x, float y) { // Adds an image to the pageElements AddCaptionAndRectangle(pageElements, "Image
DPDFLogo.png"), x + 112.5f, y + 50f, 0.24f); // Image is sized and centered in the rectangle image.SetBounds(215, 60); image.VAlign = VAlign.Center; image.Align = Align.Center; pageElements.Add(image); } private vo
eElements AddCaptionAndRectangle(pageElements, "Label & PageNumberingLabel Page Elements:", x, y); string labelText = "Labels can be rotated"; string numberText = "PageNumberingLabels contain page numbe
x + 5, y + 12, 220, 80, Font.TimesRoman, 12, TextAlign.Center); label.Angle = 8; PageNumberingLabel pageNumLabel = new PageNumberingLabel(numberText, x + 5, y + 55, 220, 80, Font.TimesRoman, 12, TextAlign
ts.Add(label); } private void AddLine(Group pageElements, float x, float y) { // Adds a line to the pageElements AddCaptionAndRectangle(pageElements, "Line Page Element:", x, y); pageElements.Add(new Line(x + 5,
ne(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 pageElements Font font = Font.TimesRoman; string text = "This is a link to Dyna
t:", x, y); Label label = new Label(text, x + 5, y + 20, 215, 80, font, 12, RgbColor.Blue); label.Underline = true; Link link = new Link(x + 5, y + 20, font.GetTextWidth(text, 12), 12 - font.GetDescender(12), new UrlAction("h
ments.Add(link); } private void AddPath(Group pageElements, float x, float y) { // Adds a path to the pageElements ceTe.DynamicPDF.PageElements.Path path = new ceTe.DynamicPDF.PageElements.Path(x + 5, y + 2
hs.Add(new LineSubPath(x + 215, y + 40)); path.SubPaths.Add(new CurveToSubPath(x + 108, y + 80, x + 160, y + 80)); path.SubPaths.Add(new CurveSubPath(x + 5, y + 40, x + 65, y + 80, x + 5, y + 60)); AddCaption
(path); } private void AddRectangle(Group pageElements, float x, float y) orderedList = orderedList.GetOverFlowList(x + 5, y + 20); AddCaptionAndRectangle(page.Elements, "Ordered List Page Element Overflow:", x,
/ Create an unordered list UnorderedList unorderedList = new UnorderedList(x + 5, y + 20, 400, 90, Font.Helvetica, 10); unorderedList.Items.Add("Fruits"); unorderedList.Items.Add("Vegetables"); UnorderedSubList un
unorderedSubList.Items.Add(" Citrus"); unorderedSubList.Items.Add(" Non-Citrus"); AddCaptionAndRectangle(page.Elements, "Unordered List Page Element:", x, y, 225, 110); UnorderedSubList unorderedSubList2 =
SubList2.Iten                                                                                     s[0].SubLists.AddUnorderedSubList(); subUnorderedSubList.Items.Add("Lime
subUnordere                                                                                        norderedSubList2.Items.Add("Banana"); UnorderedSubList subUnorderedSub
subUnorderec                                                                                        UnorderedSubList(); subUnorderedSubList4.Items.Add("String Bean"); subUn
"Kidney Bean                                                                                        AndRectangle(page.Elements, "Unordered List Page Element Overflow:", x, y,
AddTextField(                                                                                       his is a Scrollable TextField"; txt.BorderColor = RgbColor.Black; txt.Backgroun
t); TextField tx                                                                                    gth = 9; txt1.BorderColor = RgbColor.Black; txt1.BackgroundColor = RgbColo
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                                                                                   ectangle(pageElements, "TextField Form Page Element:", x, y, 504, 85); } private void AddCo
nboBox("cmb                                                                                     cb.Font = Font.Helvetica; cb.FontSize = 12; cb.Items.Add("Item 1"); cb.Items.Add("Item 2")
ble"); cb.Item                                                                                  ComboBox cb1 = new ComboBox("cmb1name", x + 303, y + 40, 150, 20); cb1.BorderColor
nt.Helvetica;                                                                                   "Item 4"); cb1.Items.Add("Non-Editable"); cb1.Items["Non-Editable"].Selected = true; cb1.E
Add(cb1); Cor                                                                                   put.pdf");System.Diagnostics.Process.Start("Output.pdf"); AsyncConverter aConverter = ne
Converter_Co                                                                                    nverter.Convert(@"C:\temp\DocumentA.rtf", @"C:\temp\OutputA.pdf"); aConverter.Convert(
(@"C:\temp\DocumentC.rtf", @"C:\temp\OutputC.pdf"); aConverter.Convert("http://www.yahoo.com", @"C:\Temp\yahoo.pdf"); ConversionOptions options = new ConversionOptions(720, 720, 72, true); ceTe.Dynamic
p\\output.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 inc
le border=\"1\"><tr><td>100</td><td>200</td>" + "<td>300</td></tr><tr><td>400</td><td>500</td><td>600</td></tr></table></body></html>";Conversion.Converter.ConvertHtmlString(sampleHtml, "C:\\temp\\
ame", Path.Combine(GetPath(), "LetterPortrait.pdf")); printJob.DocumentName = "Letter Portrait"; if (printJob.Printer.Color) printJob.PrintOptions.Color = true; if (printJob.Printer.Collate) printJob.PrintOptions.Collate =
```

# DEVELOPED FOR INTUITIVE USE

## DynamicPDF—Comprehensive PDF Solutions for .NET Developers

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

DynamicPDF®

WWW.DYNAMICPDF.COM

**TRY OUR PDF SOLUTIONS FREE TODAY!**

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

ceTesoftware

Figure 6 **The Ball Class**

```
var Ball = function( {
  // List of variables only the object can see (private variables).
  var velocity = [0,0];
  var position = [0,0];
  var element = $('#ball');
  var paused = false;

  // Method that moves the ball based on its velocity. This method is only used
  // internally and will not be made accessible outside of the object.
  function move(t) {
  }

  // Update the state of the ball, which for now just checks
  // if the play is paused and moves the ball if it is not.
  // This function will be provided as a method on the object.
  function update(t) {
    // First the motion of the ball is handled
    if(!paused) {
      move(t);
    }
  }

  // Pause the ball motion.
  function pause() {
    paused = true;
  }

  // Start the ball motion.
  function start() {
    paused = false;
  }

  // Now explicitly set what consumers of the Ball object can use.
  // Right now this will just be the ability to update the state of the ball,
  // and start and stop the motion of the ball.
  return {
    update:       update,
    pause:        pause,
    start:        start
}
```

Note that requestAnimationFrame is called again in the function, as the ball has finished updating. This ensures continuous animation.

While this code will work, there may be an issue where the script starts running before the page is fully loaded. To avoid this, I'll kick off the code when the page is loaded, using jQuery:

```
var ball;
var lastUpdate;

$(document).ready(function() {
  lastUpdate = 0;
  ball = Ball();
  requestAnimationFrame(update);
});
```

Because I know the speed of the ball (velocity) and the time since its last update, I can do some simple physics to move the ball forward:

```
var position = [300, 300];
var velocity = [-1, -1];
var move = function(t) {
  position[0] += velocity[0] * t;
  position[1] += velocity[1] * t;

  element.css('left', position[0] + 'px');
  element.css('top', position[1] + 'px');
}
```

Try running the code and you'll see the ball move at an angle and off the screen. This is fun for a second, but once the ball goes off the edge of the screen, the fun stops. So the next step is to make the ball bounce off the edges of the screen, as implemented in **Figure 7**. Add this code and running the app will show a continuously bouncing ball.

Figure 7 **Simple Ball Bouncing Physics**

```
var move = function(t) {
  // If the ball hit the top or bottom, reverse the vertical speed.
  if (position[1] <= 0 || position[1] >= innerHeight) {
   velocity[1] = -velocity[1];
  }

  // If the ball hit the left or right sides, reverse the horizontal speed.
  if (position[0] <= 0 || position[0] >= innerWidth) {
   velocity[0] = -velocity[0];
  }
  position[0] += velocity[0] * t;
  position[1] += velocity[1] * t;

  element.css('left', (position[0] - 32) + 'px');
  element.css('top', (position[1] - 32) + 'px');
}
```

Figure 8 **Movement Controls for the Player Sprite**

```
var move = function(y) {
  // Adjust the player's position.
  position[1] += y;

  // If the player is off the edge of the screen, move it back.
  if (position[1] <= 0)  {
    position[1] = 0;
  }

  // The height of the player is 128 pixels, so stop it before any
  // part of the player extends off the screen.
  if (position[1] >= innerHeight - 128) {
    position[1] = innerHeight - 128;
  }

  // If the player is meant to stick to the right side, set the player position
  // to the right edge of the screen.
  if (side == 'right') {
    position[0] = innerWidth - 128;
  }

  // Finally, update the player's position on the page.
  element.css('left', position[0] + 'px');
  element.css('top', position[1] + 'px');
}
```

Figure 9 **Adding Touch and Keyboard Controls**

```
var distance = 24;  // The amount to move the player each step.
$(document).ready(function() {
  lastUpdate = 0;
  player = Player('player', 'left');
  player.move(0);
  opponent = Player('opponent', 'right');
  opponent.move(0);

  ball = Ball();

  // pointerdown is the universal event for all types of pointers -- a finger,
  // a mouse, a stylus and so on.
  $('#up')    .bind("pointerdown", function() {player.move(-distance);});
  $('#down')  .bind("pointerdown", function() {player.move(distance);});

  requestAnimationFrame(update);
});

$(document).keydown(function(event) {
  var event = event || window.event;

  // This code converts the keyCode (a number) from the event to an uppercase
  // letter to make the switch statement easier to read.
  switch(String.fromCharCode(event.keyCode).toUpperCase()) {
    case 'A':
      player.move(-distance);
      break;
    case 'Z':
      player.move(distance);
      break;
  }
  return false;
});
```

Figure 10 **Make the Ball Follow Its Owner**

```
var move = function(t) {
  // If there is an owner, move the ball to match the owner's position.
  if (owner !== undefined) {
    var ownerPosition = owner.getPosition();
    position[1] = ownerPosition[1] + 64;
    if (owner.getSide() == 'left') {
      position[0] = ownerPosition[0] + 64;
    } else {
      position[0] = ownerPosition[0];
    }

  // Otherwise, move the ball using physics. Note the horizontal bouncing
  // has been removed -- ball should pass by a player if it
  // isn't caught.
  } else {
    // If the ball hits the top or bottom, reverse the vertical speed.
    if (position[1] - 32 <= 0 || position[1] + 32 >= innerHeight) {
      velocity[1] = -velocity[1];
    }
    position[0] += velocity[0] * t;
    position[1] += velocity[1] * t;
  }
  element.css('left', (position[0] - 32) + 'px');
  element.css('top',  (position[1] - 32) + 'px');
}
```

## A Moveable Player

Now it's time to make the Player objects. The first step in fleshing out the player class will be to make the move function change the position of the player. The side variable will indicate which side of the court the player will reside, which will dictate how to position the player horizontally. The y value, passed into the move function, will be how much up or down the player will move:

```
var Player = function (elementName, side) {
  var position = [0,0];
  var element = $('#'+elementName);
  var move = function(y) {
  }

  return {
    move: move,
    getSide:      function()  { return side; },
    getPosition:  function()  { return position; }
  }
}
```

**Figure 8** lays out player movement, stopping the motion if the player sprite reaches the top or bottom of the window.

I can now create two players and have them move to their appropriate side of the screen:

```
player = Player('player', 'left');
player.move(0);

opponent = Player('opponent', 'right');
opponent.move(0);
```

## Keyboard Input

So in theory you can move the player, but it won't move without instruction. Add some controls to the player on the left. You want two ways to control that player: using the keyboard (on PCs) and tapping the controls (on tablets and phones).

To ensure consistency between touch inputs and mouse inputs on various platforms, I'll use the great unifying framework Hand.js (handjs.codeplex.com). First, I'll add the script to HTML in the head section:

```
<script src="hand.minified-1.3.8.js"></script>
```

**Figure 9** demonstrates using Hand.js and jQuery to control the player when you press keyboard keys A and Z, or when you tap the controls.

## Catch the Ball

As the ball bounces around, I want to let players catch it. When it's caught, the ball has an owner, and it follows the motion of that owner. **Figure 10** adds functionality to the ball's move method, allowing for an owner, which the ball will then follow.

Currently, there's no way to get the position of a Player object, so I'll add the getPosition and getSide accessors to the Player object:

```
return {
  move: move,
  getSide:      function()  { return side; },
  getPosition:  function()  { return position; }
}
```

Now, if the ball has an owner, it will follow that owner around. But how do I determine the owner? Somebody has to catch the ball. **Figure 11** shows how to determine when one of the player sprites touches the ball. When that happens, I'll set the owner of the ball to that player.

The first step in fleshing out the player class will be to make the "move" function change the position of the player.

If you try playing the game now, you'll find the ball bounces off the top of the screen, and you can move the player to catch it. Now, how do you throw it? That's what the right-hand controls are for—aiming the ball. **Figure 12** adds a "fire" function to the player, as well as an aim property.

**Figure 13** augments the keyboard function to set the player's aim and fire functions. Aiming will work slightly differently. When the aiming key is released, the aim will return to straightforward.

Figure 11 **Collision Detection for the Ball and Players**

```
var update = function(t) {

// First the motion of the ball is handled.
if(!paused) {
    move(t);
}

// The ball is under control of a player, no need to update.
if (owner !== undefined) {
    return;
}

// First, check if the ball is about to be grabbed by the player.
var playerPosition = player.getPosition();
  if (position[0] <= 128 &&
      position[1] >= playerPosition[1] &&
      position[1] <= playerPosition[1] + 128) {
    console.log("Grabbed by player!");
    owner = player;
}

// Then the opponent...
var opponentPosition = opponent.getPosition();
  if (position[0] >= innerWidth - 128 &&
      position[1] >= opponentPosition[1] &&
      position[1] <= opponentPosition[1] + 128) {
    console.log("Grabbed by opponent!");
    owner = opponent;
}
```

The final addition will be touch support on all controls. I'll make the controls on the right change the aim of the player. I'll also make it so touching anywhere on the screen fires the ball:

```
$('#left')  .bind("pointerdown", function() {player.setAim(-1);});
$('#right') .bind("pointerdown", function() {player.setAim(1);});
$('#left')  .bind("pointerup",   function() {player.setAim(0);});
$('#right') .bind("pointerup",   function() {player.setAim(0);});
$('body')   .bind("pointerdown", function() {player.fire();});
```

## Keep Score

When the ball passes a player, I want to change the score and give the ball to that player. I'll use custom events so I can separate scoring from any of the existing objects. The update function is getting long, so I'll add a new private function called checkScored:

```
function checkScored() {
  if (position[0] <= 0) {
    pause();
    $(document).trigger('ping:opponentScored');
  }

  if (position[0] >= innerWidth) {
    pause();
    $(document).trigger('ping:playerScored');
  }
}
```

Figure 12 **Aim and Fire the Ball**

```
var aim = 0;
var fire = function() {
  // Safety check: if the ball doesn't have an owner, don't not mess with it.
  if (ball.getOwner() !== this) {
    return;
  }

  var v = [0,0];

  // Depending on the side the player is on, different directions will be thrown.
  // The ball should move at the same speed, regardless of direction --
  // with some math you can determine that moving .707 pixels on the
  // x and y directions is the same speed as moving one pixel in just one direction.
  if (side == 'left') {
    switch(aim) {
    case -1:
      v = [.707, -.707];
      break;
    case 0:
      v = [1,0];
      break;
    case 1:
      v = [.707, .707];
    }
  } else {
    switch(aim) {
    case -1:
      v = [-.707, -.707];
      break;
    case 0:
      v = [-1,0];
      break;
    case 1:
      v = [-.707, .707];
    }
  }
  ball.setVelocity(v);

  // Release control of the ball.
  ball.setOwner(undefined);
}

// The rest of the Ball definition code goes here...

return {
  move: move,
  fire: fire,
  getSide:      function()  { return side; },
  setAim:       function(a) { aim = a; },
  getPosition:  function()  { return position; },
}
```

**Figure 14** shows the code that reacts to those events to update the score and hand over the ball. Add this code to the bottom of the JavaScript document.

Now when the ball makes it past your opponent (which isn't that difficult, as the opponent doesn't move) your score will go up, and the ball will be handed to the opponent. However, the opponent will just hold onto the ball.

## Get Smart

You almost have a game. If only you had someone with whom to play. As a last step, I'll show how to control the opponent with simple AI. The opponent will try to stay parallel with the ball as it moves about. If the opponent catches the ball, it will move randomly and fire the ball in a random direction. To make the AI feel a little more human, I'll add delays in everything done. This isn't highly intelligent AI, mind you, but it will be something to play the game against.

When designing this kind of system, it's good to think in states. The opponent AI has three possible states: following, aiming/shooting and waiting. I'll be the state between following actions to add a more human element. Start with just that for the AI object:

```
function AI(playerToControl) {
  var ctl = playerToControl;
  var State = {
    WAITING: 0,
    FOLLOWING: 1,
    AIMING: 2
  }
  var currentState = State.FOLLOWING;
}
```

Depending on the state of the AI, I'll want it to do a different action. Just like the ball, I'll make an update function I can call in requestAnimationFrame to have the AI act according to its state:

Figure 13 **Set the Player's Aiming Function**

```
$(document).keydown(function(event) {
  var event = event || window.event;
  switch(String.fromCharCode(event.keyCode).toUpperCase()) {
    case 'A':
      player.move(-distance);
      break;
    case 'Z':
      player.move(distance);
      break;
    case 'K':
      player.setAim(-1);
      break;
    case 'M':
      player.setAim(1);
      break;
    case ' ':
      player.fire();
      break;
  }

  return false;
});

$(document).keyup(function(event) {
  var event = event || window.event;
  switch(String.fromCharCode(event.keyCode).toUpperCase()) {
    case 'K':
    case 'M':
      player.setAim(0);
      break;
  }

  return false;
});
```

Figure 14 **Update the Scoreboard**

```
$(document).on('ping:playerScored', function(e) {
  console.log('player scored!');
  score[0]++;
  $('#playerScore').text(score[0]);
  ball.setOwner(opponent);
  ball.start();
});

$(document).on('ping:opponentScored', function(e) {
  console.log('opponent scored!');
  score[1]++;
  $('#opponentScore').text(score[1]);
  ball.setOwner(player);
  ball.start();
});
```

Figure 15 **A Simple FOLLOWING AI**

```
function moveTowardsBall() {
  // Move the same distance the player would move, to make it fair.
  if(ball.getPosition()[1] >= ctl.getPosition()[1] + 64) {
    ctl.move(distance);
  } else {
    ctl.move(-distance);
  }
}

function update() {
  switch (currentState) {
    case State.FOLLOWING:
      moveTowardsBall();
      currentState = State.WAITING;
    case State.WAITING:
      setTimeout(function() {
        currentState = State.FOLLOWING;
      }, 400);
      break;
    }
  }
}
```

Figure 16 **Aiming and Firing AI**

```
function repeat(cb, cbFinal, interval, count) {
  var timeout = function() {
    repeat(cb, cbFinal, interval, count-1);
  }
  if (count <= 0) {
    cbFinal();
  } else {
    cb();
    setTimeout(function() {
      repeat(cb, cbFinal, interval, count-1);
    }, interval);
  }
}
function aimAndFire() {
  // Repeat the motion action 5 to 10 times.
  var numRepeats = Math.floor(5 + Math.random() * 5);

  function randomMove() {
    if (Math.random() > .5) {
      ctl.move(-distance);
    } else {
      ctl.move(distance);
    }
  }

  function randomAimAndFire() {
    var d = Math.floor( Math.random() * 3 - 1 );
    opponent.setAim(d);
    opponent.fire();

    // Finally, set the state to FOLLOWING.
    currentState = State.FOLLOWING;
  }

  repeat(randomMove, randomAimAndFire, 250, numRepeats);
}
```

```
function update() {
  switch (currentState) {
    case State.FOLLOWING:
      // Do something to follow the ball.
      break;
    case State.WAITING:
      // Do something to wait.
      break;
    case State.AIMING:
      // Do something to aim.
      break;
  }
}
```

The FOLLOWING state is straightforward. The opponent moves in the vertical direction of the ball, and the AI transitions to the WAITING state to inject some slowed reaction time. **Figure 15** shows these two states.

When designing this kind of system, it's good to think in states.

With the code in **Figure 15**, the AI alternates between having to follow the ball and wait a split second. Now add the code to the game-wide update function:

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

  ball.update(t);
  ai.update();

  requestAnimationFrame(update);
}
```

When you run the game, you'll see the opponent follow the ball's movements—not a bad AI in less than 30 lines of code. Of course, if the opponent catches the ball, it won't do anything. So for the last trick of the hour, it's time to handle the actions for the AIMING state. I want the AI to move randomly a few times and then fire the ball in a random direction. **Figure 16** adds a private function that does just that. Adding the aimAndFire function to the AIMING case statement makes a fully functional AI against which to play.

## Wrapping Up

By now, you have a full-fledged Web game that works on PCs, smartphones and tablets. There are many possible improvements to this game. It will look a little awkward in portrait mode on a smartphone, for example, so you need to make sure you're holding the phone in landscape for it to work properly. This is just a small demonstration of the possibilities for game development for the Web and beyond. ■

**Michael Oneppo** *is a creative technologist and former program manager at Microsoft on the Direct3D team. His recent endeavors include working as CTO at the technology nonprofit Library For All and exploring a master's degree at the NYU Interactive Telecommunications Program.*

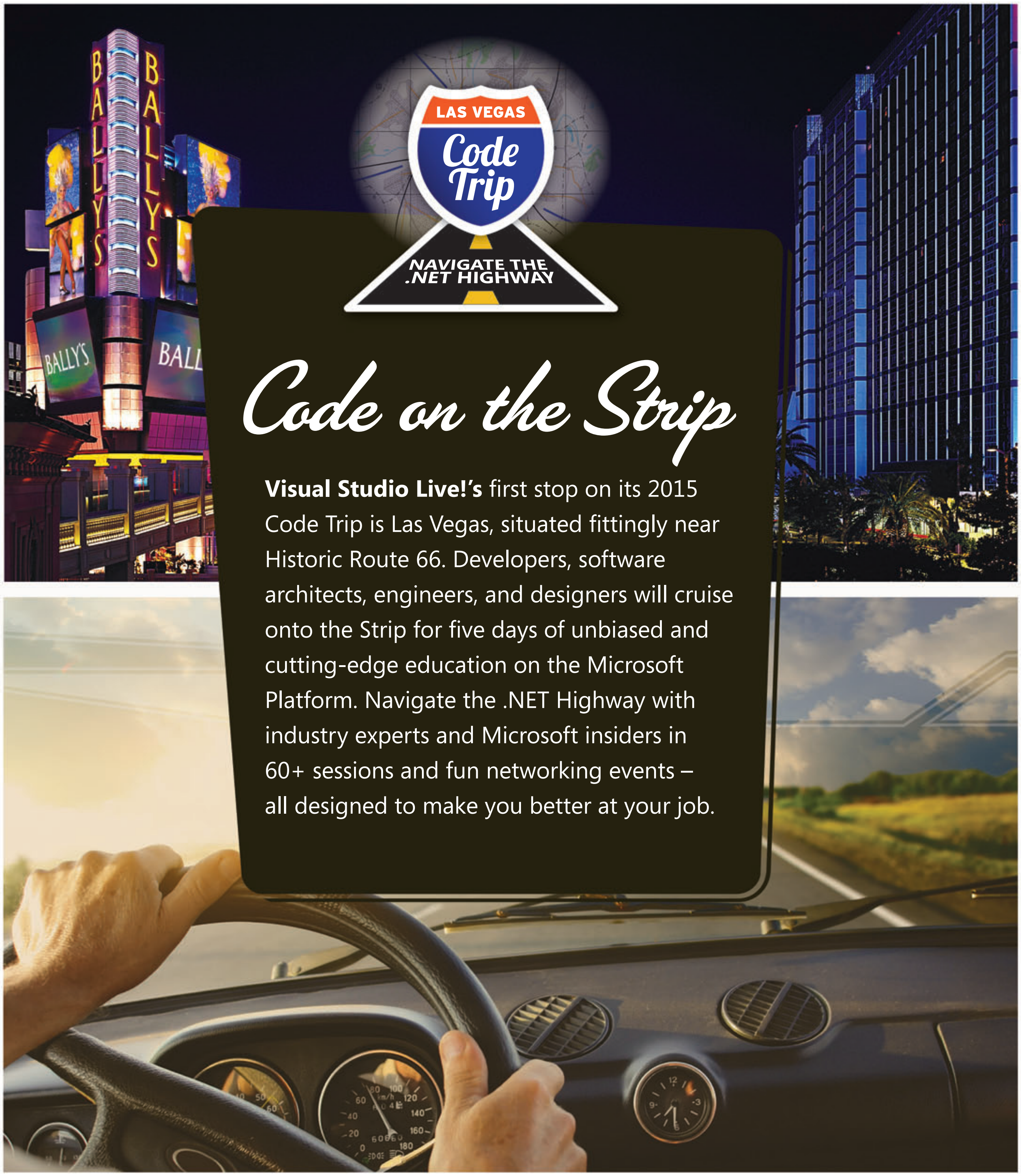
**Thanks** to the following technical expert for reviewing this article:
*Mohamed Ameen Ibrahim*

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 Framework
- JavaScript / HTML5 Client
- Modern App Development
- ASP.NET
- Cross-Platform Mobile Development
- Database and Analytics
- Windows Client
- Cloud Computing
- Windows Phone
- SharePoint Cloud Business Apps

*Sessions are filling up quickly. Register Today!*

Use promo code VSLMAR2

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**

# Create an IoT Device Using the Gadgeteer with Azure Blob Storage

Benjamin Perkins

In the next five years, the number of Internet-connected devices will be more than three times the total human population. These devices won't be restricted to the smartphones and tablets we use now, but will include additional devices embedded into home appliances, elevators, automobiles, business environments, arm bands, clothing and much more. These devices will capture information about energy consumption, speed, temperatures, the presence of various gases, blood pressure, heart rate, program or hardware exceptions, and just about anything you can imagine.

The data captured by these devices needs to be stored in a highly scalable, highly reliable and highly usable environment. By usable I mean once the data is stored, the platform on which it exists should provide services, features and processing power to analyze, learn from and act on the gathered data. In this article, I'm going to show you how to use the Gadgeteer—a rapid hardware development platform based on the Microsoft .NET Micro Framework—to build a device for capturing data; Microsoft Azure Storage to store the data; and the Azure platform to analyze and consume the data. By following along, you'll begin your journey into the Internet of Things (IoT) generation.

I'll discuss those three components, the Gadgeteer, Azure Blob Storage, and some data capturing and analysis components of the Azure platform in detail. In addition, I'll provide thorough code-level instructions on how you can use the Gadgeteer to insert an image into an Azure Blob Storage container.

This article discusses:

- Configuring an IoT device using the Gadgeteer
- Connecting the device to the Internet
- Uploading an image to an Azure Blob container in the cloud

Technologies discussed:

Gadgeteer, .NET Micro Framework, Microsoft Azure, Visual Studio

Code download available at:

msdn.microsoft.com/magazine/msdnmag0315

## Gadgeteer

The Gadgeteer, provided by GHI Electronics LLC (bit.ly/11ko85B), includes kits, modules and mainboards (such as the FEZ Raptor, FEZ Hydra and the FEZ Spider used for this project) for creating many different types of devices. These devices can be used to capture numerous varieties of data. For example, they might use a gas-sensing module for detecting gases in the air, a motion detector or a module for measuring relative temperature and humidity. The FEZ Spider-compatible modules necessary to complete the project discussed in this article are: a camera, an Ethernet jack and a power module. In the first section, I'll show you how to configure the modules and develop the code required to prepare the captured image for insertion into an Azure Blob Storage container.

**Figure 1** shows the configuration of the project. In addition to the camera, Ethernet jack, and power supply, there's also a character

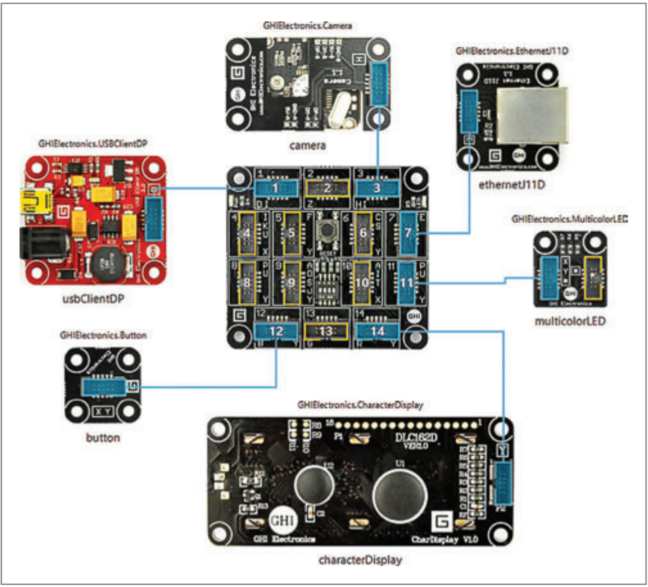


Figure 1 **FEZ Spider Configuration for Taking a Picture and Storing in an Azure Blob Container**

display for displaying the IP address and date and time on the device; an LED light for visual notifications; and a button for triggering the capture of the picture and the process to upload it.

After installing the .NET Micro Framework, GHI binaries (Package 2014 R5) and the .NET Gadgeteer SDK onto your development machine, begin by creating a new project in Visual Studio 2012 (Visual Studio 2013 is also supported, although not extensively tested) and selecting the Gadgeteer template. Once you've given the project a name, the wizard will walk you through the selection of a mainboard and Micro Framework version. Then, using the Toolbox, add the modules to build your device, similar to what's shown in **Figure 1**.

When the modules are all physically connected to the sockets on the mainboard, begin to make the code changes required to achieve the project goal—connecting to the network and taking a picture. The module used to make the network connection is the ethernetJ11D, which supports the common RJ-45 network adapter. Here's some code to make the ethernetJ11D module perform the connection and get an allocated IP address:

```
ethernetJ11D.NetworkInterface.Open();
ethernetJ11D.NetworkSettings.EnableDhcp();
ethernetJ11D.NetworkSettings.EnableDynamicDns();
ethernetJ11D.UseDHCP();
```

Of course, this isn't the only way to achieve the connection; however, it's what worked best and was simplest in this context.

Once connected, you may want to execute a simple System.Net.HttpWebRequest request, similar to what's shown in **Figure 2**, to make sure the connection to the Internet is working as expected.

Note that the HTTPS protocol won't work here without some configuration. If you receive an exception such as "A first chance exception of type 'System.NotSupportedException' occurred in Microsoft.SPOT.Net.Security.dll," the SSL Seed on the device must be updated. You can do this using the .NET Micro Framework Deployment Tool (MFDeploy.exe), located in the C:\Program Files (x86)\Microsoft .NET Micro Framework\v4.3\Tools directory. With the device connected to the development machine, navigate to Target | Manage Device Keys and click the Update SSL Seed, as shown in **Figure 3**. When that's done, HTTPS should work as expected.

Once you've gotten the connection to the network and the Internet configured and working, it's important to set the date and time on the device. As discussed later, the date and time are required parts of the PUT Blob REST API (bit.ly/1BTiIu9), which you'll use to insert the image into the Azure Blob container. To set the date and time, use the TimeServiceSettings class, which is part of the Microsoft.SPOT.Time namespace, as shown here:

```
TimeServiceSettings time = new TimeServiceSettings()
{
  ForceSyncAtWakeUp = true
};
IPAddress[] address = Dns.GetHostEntry("time.windows.com").AddressList;
time.PrimaryServer = address[0].GetAddressBytes();
TimeService.Settings = time;
TimeService.SetTimeZoneOffset(0);
TimeService.Start();
```

Once the TimeService is running, you can retrieve a current timestamp using the standard DateTime.UtcNow or DateTime.Now properties.

Capturing an image using a GHI FEZ Spider-compatible camera requires just a single line of code:

```
camera.TakePicture()
```

Recall that I added the Button module to the device. It's from the Button's pressed event that the TakePicture method is called. When the picture is taken, the camera.PictureCaptured event is triggered and in turn calls the method camera_PictureCaptured(Camera sender, GT.Picture e). I use standard C# code to wire up the event handler and the method for linking the Pressed and Captured events to their corresponding methods:

```
button.ButtonPressed += button_ButtonPressed;
camera.PictureCaptured += camera_PictureCaptured;
```

The camera_PictureCaptured method receives GT.Picture as a parameter. After being converted to a byte[] array using the e.PictureData property, the image data is passed to the custom method for inserting into the Azure Blob container. Reading this, the process may seem complicated, but it's really quite simple, as shown in **Figure 4**. Also, you can download the code sample that accompanies this article to see the whole thing.

**Figure 5** shows the code used to consume the AzureBlob class, discussed in the next section where I create the Authorization header (bit.ly/1z0gThK) and call the REST API that inserts the image into the Azure Blob container.

Figure 2 **Testing Your Internet Connection**

```
void makeGenericHTTPRequest()
{
  try
  {
    string url = "https://www.contoso.com/";
    using (var req = System.Net.HttpWebRequest.Create(url))
    {
      using (var res = req.GetResponse())
      {
        Debug.Print("HTTP Response length: " + res.ContentLength.ToString());  }
    }
  }
  catch (Exception ex)
  {
    Debug.Print(ex.Message);
  }
}
```

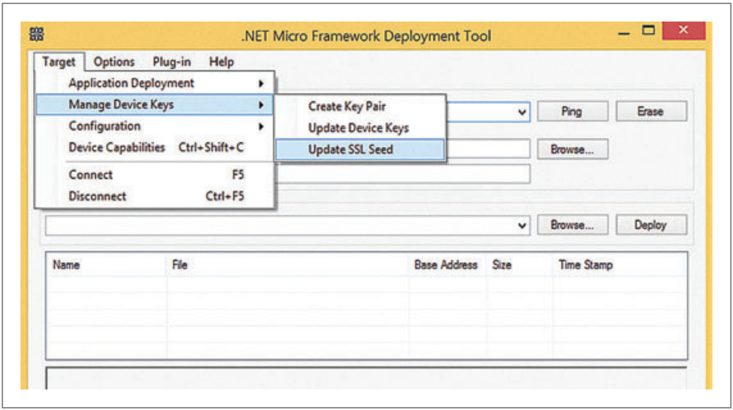


Figure 3 **The .NET Micro Framework Deployment Tool MFDeploy**

From the device perspective, that's it. The device is now assembled and connected to the Internet, and the logic to capture the image and send it in the correct format to the PUT Blob REST API is all complete. Next, I create the Azure Blob Storage account and container and configure the code required to insert the image.

## Azure Blob Storage

Azure Blob Storage is a useful service that provides access to images, documents, videos, and so forth from anywhere using HTTP or HTTPS. For example, if you have an image named home.bmp, a public Azure Storage account named contosox and a container named blob, accessing the .bmp file from a browser is as simple as entering http://contosox.blob.core.windows.net/blob/home.bmp into a browser or referencing the URL from HTML or source code.

Inserting, listing, downloading and deleting blobs from a standard .NET application on an Azure Blob container is accomplished using the Azure .NET Storage Client library via the Microsoft.Windows-Azure.Storage assembly. This assembly, unfortunately, isn't available or compatible with the .NET Micro Framework or on the Gadgeteer device. This isn't a problem, though, because the Azure Blob Storage service provides publicly accessible REST APIs that support the same insert, list, download and delete capabilities (see bit.ly/1xPN55v for more information). Therefore, consuming these features from an IoT device is as simple as calling a standard REST API.

I'll show you how to create an Azure Blob Storage account and container; how to create the REST API Authorization header using the .NET Micro Framework 4.3; and how to get the image captured in the previous section uploaded into the Azure Blob container.

Create the Azure Storage account from within the Azure Management Console and then click the + Add button to add the container.

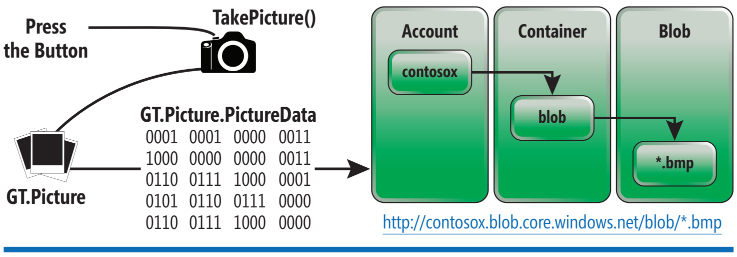


Figure 4 **Gadgeteer Picture-Capturing Process Flow**

In this example, the storage account name is *contosox* and the container is *blob*, as illustrated in **Figure 6**.

The example in **Figure 7** is a breakdown of what's required to consume the PUT Blob REST API. Appending the name of the image to the URL shown in **Figure 6** and calling the GetRequestStream method of the System.Net.HttpWebRequest class returns a System.IO.Stream object. And using the Write method of the System.IO.Stream class successfully saves the image to the Azure Blob container. Note that the blobContent parameter of the Write method is the byte[] content contained in the GT.Picture.PictureData property.

That's not too complicated; it's very much like calling a REST API from a standard ASP.NET or Microsoft .NET Framework application. The difference—and the complexity—come from the creation of the Authorization header, defined here:

```
Authorization="[SharedKey|SharedKeyLite] <AccountName>:<Signature>"
```

When the Azure Blob container is created, it's made public, meaning that anyone can read from it via the previously mentioned URL. However, adding or deleting a Blob requires an Authorization header to be present in the request. The SharedKey value in the header notifies the server that a shared access key exists and should be used for request authentication. The SharedKey is associated to the Azure Storage account, in this example contosox, and is acquired by clicking on Manage Access Keys for the given storage account. The value contained in the Primary Access Key textbox, illustrated in **Figure 8**, is the SharedKey used for accessing the container.

Azure Blob Storage is a very useful service that provides access to images, documents, video, and so forth from anywhere in the world using HTTP or HTTPS.

Additionally, the Signature portion of the Authorization header must be a Hash-based Message Authentication Code (HMAC) constructed using a number of request attributes, computed using the System.Security.Cryptography.HashAlgorithm.ComputeHash method and encoded by the System.Convert.ToBase64String method.

I'll break that down a little, starting with the components necessary for constructing the Authorization header. These include attributes such as x-ms-version, content-length, content-type, shared access key and many other header values described in detail at bit.ly/1BTiIu9. One of the more important attributes is the x-ms-date, which is the reason you had to set the date and time during the initialization of the device. The x-ms-date must be a UTC timestamp. The timestamp on the device must have a differential of less

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

Figure 5 **Inserting the Image into Azure Blob Storage**

```
void insertImageintoAzureBlob(GT.Picture picture)
{
  AzureBlob storage = new AzureBlob()
  {
    Account = "ACCOUNT-NAME",
    BlobEndPoint = "https://ACCOUNT-NAME.blob.core.windows.net/",
    Key = "CONTAINER PRIVATE KEY"
  };
  if (ethernetJ11D.IsNetworkUp)
  {
    storage.PutBlob("CONTAINER-NAME", picture.PictureData);
  }
  else
  {
    characterDisplay.Print("NO NETWORK CONNECTION");
  }
}
```

than 15 minutes when compared to the timestamp on the server where the storage service is hosted. The storage service ensures the request isn't older than 15 minutes; if the time frame has a greater differential, the service returns a 403 (Forbidden). The other attributes don't change and can be hardcoded or retrieved from some other source—a configuration file, for example.

Once the Authorization header is formatted correctly, which takes about 35 lines of code, it needs to be hashed and encoded per the requirements. Using the full version of the .NET Framework, the code required resembles this:

```
using (HashAlgorithm hashSHA256 = new HashAlgorithm(HashAlgorithmType.SHA256))

{
  Byte[] dataToHmac = System.Text.Encoding.UTF8.GetBytes(canonicalizedstring);
  signature = Convert.ToBase64String(hashSHA256.ComputeHash(dataToHmac));
}
```

The System.Security.Cryptography.HashAlgorithm.ComputeHash method required to hash the Authorization header does exist in the .NET Micro Framework version 4.3 (bit.ly/1wlI770) and is a valid option for hashing the Signature. I chose, however, to develop a WebAPI that accepts the constructed header as a parameter, encodes it, hashes it and returns it to the Gadgeteer for use with the PUT Blob REST API call. I chose this approach primarily because I wanted to test calling a WebAPI from the Gadgeteer and found this to be a logical place to do it. The following code shows how the WebAPI was called from the Gadgeteer device:

```
string queryString = "constructedHeader=" + constructedHeader;
Uri uri = new Uri("https://*??.azurewebsites.net/api/HMACSHA256?" + queryString);
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
Stream dataStream = response.GetResponseStream();
StreamReader reader = new StreamReader(dataStream);
string responseFromServer = reader.ReadToEnd();
reader.Close();
response.Close();
```

The response from a WebAPI, as you likely already know, is a JSON object and is the contents of the responseFromServer string. There is an open source MicroJSON library at microjson.codeplex.com, which is useful for parsing the results of the WebAPI and getting the hashed and encoded Authorization header value:

```
var jdom = (JObject)JsonHelpers.Parse(responseFromServer);
var hashedvalue = jdom["HashedValue"];
StringBuilder authorizationHeader = new StringBuilder("{0} {1}:{2}")
  .Replace("{0}", SharedKeyAuthorizationScheme)
  .Replace("{1}", "contosox")
  .Replace("{2}", hashedValue);
```

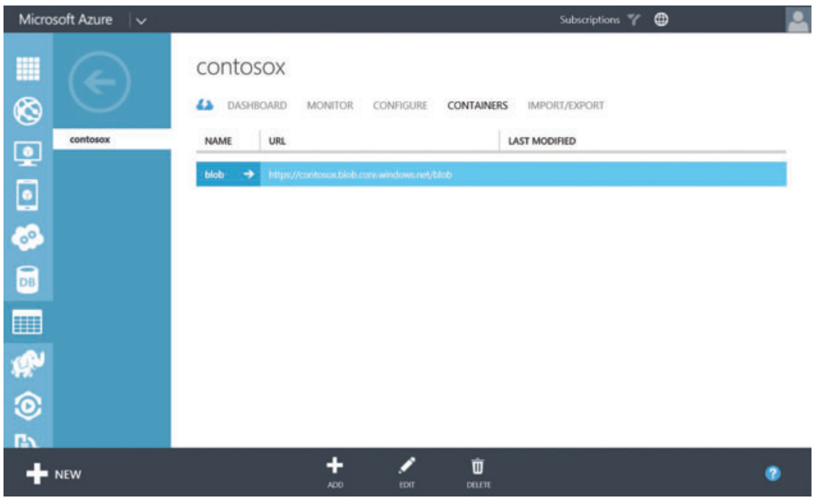
Alternatively, using the string.IndexOf method in combination with the string.Substring method can achieve the same result.

The string.Format method isn't currently supported in the .NET Micro Framework; therefore, I used StringBuilder.Replace to construct the authorizationHeader. As noted earlier, the Authorization header consists of the SharedKeyAuthorizationScheme, which is a combination of either SharedKey or SharedKeyLite; the Azure Blog Storage name, contosox; and the encoded and hashed Signature. When the Authorization header is correctly formed, add it to the PUT Blob REST API request object and execute it. Once the image is authenticated, it's added to the Blob container:

```
request.Headers.Add("Authorization", authorizationHeader);
```

In this example, the event that triggers the insertion of the image into the Azure Blob container is the press of a button. I chose this approach simply to create this proof-of-concept project. A real-world implementation might instead use a motion, gas, barometric pressure, temperature or moisture sensor. Really, anything that a module exists for that triggers an event when a given threshold is breached can be used to take a picture and upload it to Azure. In many cases, the requirements for the IoT device will not include an image, only the storage of captured data such as temperature, speed, gas level, time and so on. Such data can be stored in a database via a simple WebAPI call, similar to the one I discussed previously, but without the need for the Authorization header because there's no Blob or image.

Now that the data is captured and stored on the Azure platform, the question is what to do with it and how to analyze and learn from it. In the next section I'll provide some ideas and concluding thoughts for continued review, development and future discussions. My intent here hasn't necessarily been to show all



| NAME | URL | LAST MODIFIED |
|---|---|---|
| blob | https://contosox.blob.core.windows.net/blob | |

Figure 6 **Creating the Azure Storage Container for Storing the Image Taken from the IoT Device**

Figure 7 **Consuming the PUT Blob REST API**

```
Uri uri = new Uri("ACCOUNT-URL\CONTAINER\" + "IMAGE-NAME");
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
try
{
  using (Stream requestStream = request.GetRequestStream())
  {
    requestStream.Write(blobContent, 0, blobLength);
  }
}
catch (WebException ex)
{
  Debug.Print(ex.Message);
}
```

the details of an end-to-end IoT solution, but rather to provide food for thought and to drive these concepts forward.

## Microsoft Azure

Most IT professionals have a good idea of what the term "Big Data" means. However, how to use and harvest it are likely not as clear. When I first started learning about Big Data, the initial hurdle I had was finding a data source I could use for working with tools like HDInsight, Power BI, Event Hubs, Machine Learning or Stream Analytics. Nonetheless, I started using these services, learning the features and capabilities, but without a large source of data to run my algorithms on, I quickly lost interest. What finally dawned on me was that the IoT—those Internet-connected devices—could be used for collecting data and building large sources of information for use with these Azure services.

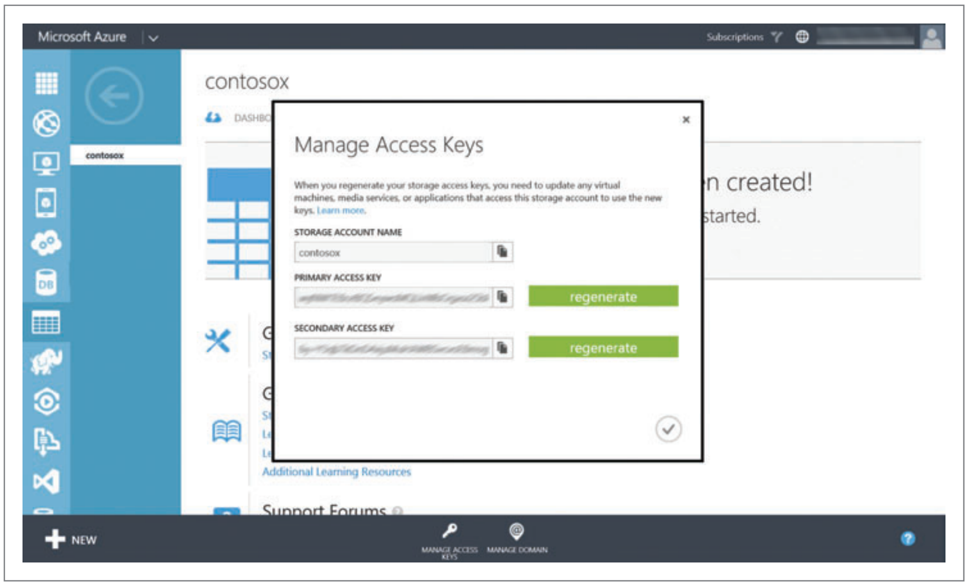
It's clear that each of these services is designed with the IoT in mind, considering Event Hubs and Stream Analytics for real-time decision making and HDInsight and Machine Learning for analyzing massive amounts of data, in search of longer-term trends. For example, Event Hubs, which is specifically designed to ingest millions of device-triggered events per second, provides the necessary scale for large-sized IoT solutions. For companies and enterprises that have already implemented an IoT solution or are beginning to create their IoT strategy, Event Hubs would be a great start to manage long-term growth and scale. Additionally, Stream Analytics integrates with Event Hubs and can provide real-time analysis of the data submitted to the Event Hubs by the devices. Stream Analytics can compare the data being sent to Event Hubs with historical data and proactively send an alert if the current patterns don't match the historical ones. Imagine the possibilities with this.

Organizations and even individuals who have large data pools—not necessarily collected by IoT devices—can use the HDInsight or Machine Learning services for the analysis of that data. HDInsight is a Hadoop solution that can scale to terabytes or petabytes of data on demand, and the Azure platform provides an almost infinite amount of storage and compute resources. Use HDInsight to find hidden business opportunities together with or independently from Machine Learning to mine data to help predict future trends and behaviors. These services introduce you to the new IoT era by exposing previously unseen information in innovative ways, opening up great possibilities, all of which can be presented in a user-friendly manner via Power BI.

## Wrapping Up

The goal of this article was to explain how to configure an IoT device using the Gadgeteer, connect it to the Internet and upload an image to an Azure Blob container in the cloud. Inserting data from a device into any Internet-accessible data source requires only a simple WebAPI call, and the configuration of the actual device hardware is as simple as dragging and dropping modules from a Toolbox menu to a design template. The only real complexity in this example was the creation of the PUT Blob REST API Authorization header, as it must be in a specific format and be encoded and hashed using the System.Security.Cryptography.HashAlgorithm.ComputeHash method via a WebAPI or using the .NET Micro Framework class.

I also summarized some Azure platform services and established their scaling capabilities for the storage and analysis of data captured from IoT devices. Once your devices begin to capture sufficient amounts of data, you can use Azure platform services like Event Hubs and Stream Analytics for real-time analysis, and HDInsight and Machine Learning for longer-term investigation. The discovery of the secrets hidden within the massive amount of generated, stored, processed and presented information can then be used for making business decisions, forecasting and helping your corporate strategy succeed. Let's do this! ■



Manage Access Keys

When you regenerate your storage access keys, you need to update any virtual machines, media services, or applications that access this storage account to use the new keys. Learn more.

STORAGE ACCOUNT NAME
contosox

PRIMARY ACCESS KEY
regenerate

SECONDARY ACCESS KEY
regenerate

Figure 8 **Acquiring the SharedKey Necessary for Azure Storage Authentication**

**Benjamin Perkins** *is a senior support escalation engineer at Microsoft and author of three books on IIS, NHibernate and Microsoft Azure. He is currently writing a C# book to be published in parallel with the release of C# 6.0. Reach him at benperk@microsoft.com.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Martin Grasruck and Colin Miller*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

vslive.com/redmond

**Redmond** AUGUST 10 - 14

MICROSOFT HEADQUARTERS, REDMOND, WA

## CODE HOME

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

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

EVENT SPONSOR Microsoft

SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA INC

## NAVIGATE THE .NET HIGHWAY

## DEVELOPMENT TRACKS INCLUDE:

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

## REGISTER NOW AND SAVE $400!

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

Use promo code REDMAR2

vslive.com/redmond

## *Join us on the Ultimate Code Trip in 2015!*

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

CONNECT WITH VISUAL STUDIO LIVE!

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

vslive.com/redmond

# Enhance Data Exploration with Azure Search

Bruno Terkaly

**The fields of data exploration,** real-time analytics and machine learning are being applied in many creative ways. Companies are building interesting architectures around a variety of open source software packages. Azure Search is one such piece of a larger architecture. Azure Search is a powerful new search experience for your Web sites and applications—even if you're not a search expert.

Azure Search is a fully managed, cloud-based service that uses a simple REST API. It includes type-ahead suggestions, suggested results based on near matches, multi-faceted navigation and the ability to adjust capacity based on need. Azure Search provides full-text search with weighting, ranking and your own search behaviors based on a schema defined by field-attribute combinations. Data is immediately indexed, which minimizes search delays.

The need for effective search is growing along with the massive size of data stores. On Facebook alone, users spend hundreds of billions of minutes searching every month. To efficiently search Wikipedia, you'd have to index 17 million entries. Twitter boasts more than 600 million users, generating= more than 50,000 tweets per day. Performing full-text search at this scale requires some creative engineering. To index and curate all this information is not for the faint of heart.

Many companies for which search is an integral component of their business are starting to work with Azure G-Series Virtual Machines with 32 cores, 448GB RAM and 6.5TB of solid-state drive (SSD). Some engineers are writing custom assembly and C code to optimize cache coherency, for both data and instruction caches. They use the caches to reduce the amount of time the CPU sits there waiting for a memory request to be fulfilled, and to reduce the overall amount of data that needs to be transferred. One challenge within these huge, multi-core machines is there are many threads competing for the memory bus. The performance boost of leveraging the L2 and L3 cache is orders of magnitude more significant. All this is important because getting lots of data quickly into your full-text engine is critical.

This article discusses:

- Configuring rich search modes within larger architectures
- Provisioning full-text search
- Develop a Node.js front end to execute searches

Technologies discussed:

Microsoft Azure, Azure Search, Node.js

Code download available at:

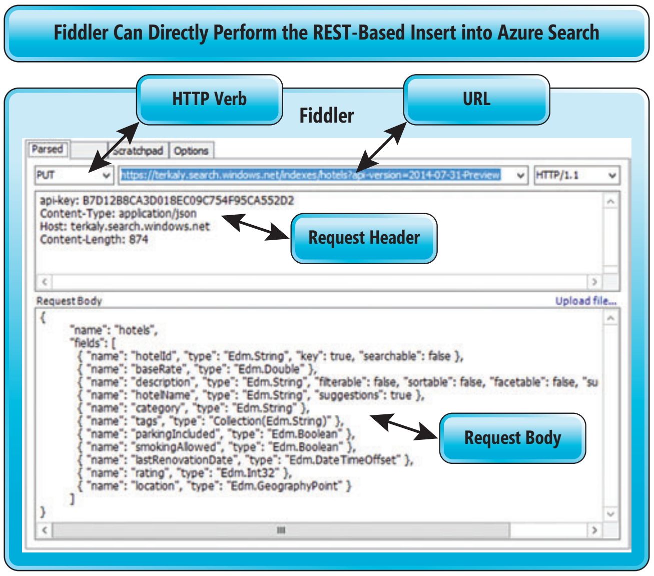
github.com/brunoterkaly/msdnnode

## Full-Text Search Done Correctly

Azure Search offers many advantages. It reduces the complexity of setting up and managing your own search index. It's a fully managed service that helps you avoid the hassle of dealing with index corruption, service availability, scaling and service updates. One of the big advantages is that Azure Search supports rich, fine-tuned ranking models. This lets you tie search results to business goals. It also offers multi-language search, spelling-mistake correction and type-ahead suggestions. If search results are weak, Azure Search can suggest queries based on near matches. You can follow a quick tutorial on provisioning a new Azure Search instance at bit.ly/1wYb8L8.

The tutorial guides you through getting started and the steps you need to perform at the portal. You'll provision Azure Search in the Azure Management Portal (portal.azure.com), which requires two things: the URL provided by the portal itself and the API-KEY. The URL represents the endpoint in the cloud running your Azure Search service to which your client app will talk. The API-KEY will need to be protected carefully, because it provides all access to your service. After all, you can't allow any client to access your Azure Search service without being authenticated with this key.

There are limits and constraints of which you should be aware in terms of the number of indexes, maximum fields per index, maximum document counts and so on. One important limitation is there are no quotas or maximum limits associated with queries. Queries-per-second (QPS) are variable, depending on available bandwidth and competition for system resources.



```
PUT  https://terkaly.search.windows.net/indexes/hotels?api-version=2014-07-31-Preview  HTTP/1.1
```

```
api-key: B7D12B8CA3D018EC09C754F95CA552D2
Content-Type: application/json
Host: terkaly.search.windows.net
Content-Length: 874
```

Request Body

```
{
    "name": "hotels",
    "fields": [
      { "name": "hotelId", "type": "Edm.String", "key": true, "searchable": false },
      { "name": "baseRate", "type": "Edm.Double" },
      { "name": "description", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false, "su
      { "name": "hotelName", "type": "Edm.String", "suggestions": true },
      { "name": "category", "type": "Edm.String" },
      { "name": "tags", "type": "Collection(Edm.String)" },
      { "name": "parkingIncluded", "type": "Edm.Boolean" },
      { "name": "smokingAllowed", "type": "Edm.Boolean" },
      { "name": "lastRenovationDate", "type": "Edm.DateTimeOffset" },
      { "name": "rating", "type": "Edm.Int32" },
      { "name": "location", "type": "Edm.GeographyPoint" }
    ]
}
```

Figure 1 **Execute an Insert into Azure Search Using Fiddler**

With the free Azure Search service, the Azure compute and storage resources backing your shared service are shared by multiple subscribers, so QPS for your solution will vary depending on how many workloads are running at the same time. For dedicated (STANDARD SKU) services, resources are all dedicated to the customer and not shared.

Once you have your URL and API-KEY, you're ready to use the service. The easiest way to do that is Fiddler because it lets you compose your own HTTP requests. (Get a free copy of Fiddler at bit.ly/1jKA1UJ.) Azure Search uses simple HTTP, so it's trivial to insert and query data with Fiddler (see **Figure 1**). Later in this article, you'll see how to use Node.js to talk to Azure Search.

As you can see, there are four things with which to concern yourself:

- HTTP Verb
- URL
- Request Header
- Request Footer

The HTTP verb (PUT, POST, GET or DELETE) maps to different operations, whether the schema definition operation, data insertion and so on. For example, PUT maps to a schema definition and POST maps to data insertion. The URL is available from the Azure Portal and may change depending on your query parameters. The API-KEY is sent in the request header. The request body is always a JSON representation of a schema or data being inserted.

Azure Search can play a key role in a larger architecture. **Figure 2** demonstrates an architecture that leverages Azure Search. Begin with the essential authentication layer, where you have some options. For example, you can use the Azure Active Directory Graph API using OAuth2 with Node.js. Speaking of Node.js, you can use it as a proxy to the URL endpoint of Azure Search, providing some structure and control to your service.

One of the great features of Azure Search is you can index and search almost any structured data—except photos, images and videos. **Figure 2** illustrates some potential data sources for Azure Search. Because relational databases aren't well suited to performing full-text searches, many startups are taking their relational data and exporting key data to Azure Search. This is also beneficial because it offloads the burden of full-text search in relational databases.

This doesn't mean that Azure Search is an end all or be all. You still need to mine data using map/reduce technologies like Hadoop or HDInsight to apply machine learning algorithms such as clustering, where text documents are grouped into topically related documents.

Imagine analyzing a tweet and giving it a score for how likely it belongs in some other category. For example, you may have a category called Rants for emotionally charged criticism or another called Raves for positive opinions. Linear classifier algorithms are often used for such insight. Azure Search isn't capable of this. But imagine the way you index documents in Azure Search is based on the way you categorize and analyze data.

## Node.js

Now I'll turn my attention to writing a Node.js front end, which I'll use as a proxy layer in front of Azure Search.

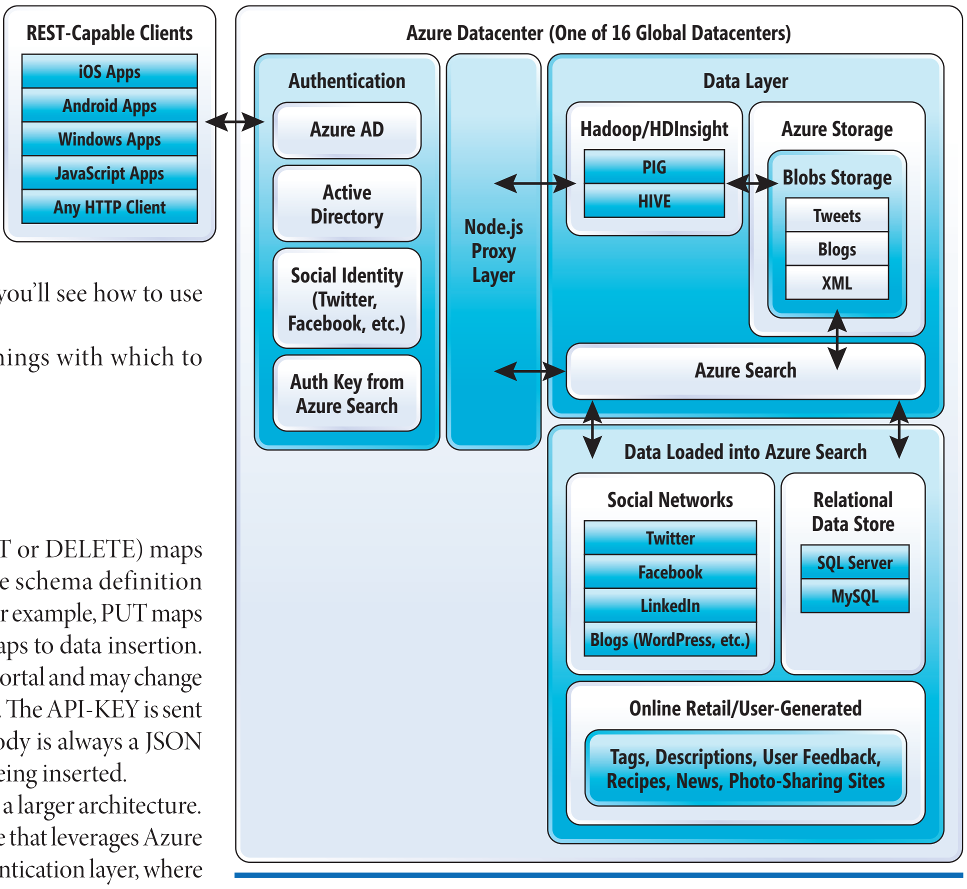


Figure 2 **Azure Search in Larger Architectures**

Figure 3 **Node.js Code That Shows How to Create an Index, Insert Data and Query Data**

```
var request = require('request');

/////////////////////////////////////////////////
// OPTIONS FOR HTTP PUT
// Purpose:    Used to create an index called hotels
/////////////////////////////////////////////////
var optionsPUT = {
  url: 'https://terkaly.search.windows.net/indexes/hotels?api-version=2014-
07-31-Preview',
  method: 'PUT',
  json: true,
  headers: {
    'api-key': 'B7D12B8CA3D018EC09C754F95CA552D2',
    'Content-Type': 'application/json'
  },
  body: {
    "name": "hotels",
    "fields": [
      { "name": "hotelId", "type": "Edm.String", "key": true, "searchable": false },
      { "name": "baseRate", "type": "Edm.Double" },
      { "name": "description", "type": "Edm.String", "filterable": false,
        "sortable": false,
        "facetable": false, "suggestions": true },
      { "name": "hotelName", "type": "Edm.String", "suggestions": true },
      { "name": "category", "type": "Edm.String" },
      { "name": "tags", "type": "Collection(Edm.String)" },
      { "name": "parkingIncluded", "type": "Edm.Boolean" },
      { "name": "smokingAllowed", "type": "Edm.Boolean" },
      { "name": "lastRenovationDate", "type": "Edm.DateTimeOffset" },
      { "name": "rating", "type": "Edm.Int32" },
      { "name": "location", "type": "Edm.GeographyPoint" }
    ]
  }
};

/////////////////////////////////////////////////
// OPTIONS FOR HTTP POST
// Purpose: Used to insert data
/////////////////////////////////////////////////
var optionsPOST = {
  url: 'https://terkaly.search.windows.net/indexes/hotels/docs/
    index?api-version=2014-07-31-Preview',
  method: 'POST',
  json: true,
  headers: {
    'api-key': 'B7D12B8CA3D018EC09C754F95CA552D2',
    'Content-Type': 'application/json'
  },
  body: {
    "value": [
    {
      "@search.action": "upload",
      "hotelId": "1",
      "baseRate": 199.0,
      "description": "Best hotel in town",
      "hotelName": "Fancy Stay",
      "category": "Luxury",
      "tags": ["pool", "view", "wifi", "concierge"],
      "parkingIncluded": false,
      "smokingAllowed": false,
      "lastRenovationDate": "2010-06-27T00:00:00Z",
      "rating": 5,
      "location": { "type": "Point", "coordinates": [-122.131577, 47.678581] }
    },
    {
      "@search.action": "upload",
      "hotelId": "2",
      "baseRate": 79.99,
      "description": "Cheapest hotel in town",
      "hotelName": "Roach Motel",
      "category": "Budget",
      "tags": ["motel", "budget"],
      "parkingIncluded": true,
      "smokingAllowed": true,
      "lastRenovationDate": "1982-04-28T00:00:00Z",
      "rating": 1,
      "location": { "type": "Point", "coordinates": [-122.131577, 49.678581] }
    },
    {
      "@search.action": "upload",
      "hotelId": "3",
      "baseRate": 279.99,
      "description": "Surprisingly expensive",
      "hotelName": "Dew Drop Inn",
      "category": "Bed and Breakfast",
      "tags": ["charming", "quaint"],
      "parkingIncluded": true,
      "smokingAllowed": false,
      "lastRenovationDate": null,
      "rating": 4,
      "location": { "type": "Point", "coordinates": [-122.33207, 47.60621] }
    },
    {
      "@search.action": "upload",
      "hotelId": "4",
      "baseRate": 220.00,
      "description": "This could be the one",
      "hotelName": "A Hotel for Everyone",
      "category": "Basic hotel",
      "tags": ["pool", "wifi"],
      "parkingIncluded": true,
      "smokingAllowed": false,
      "lastRenovationDate": null,
      "rating": 4,
      "location": { "type": "Point", "coordinates": [-122.12151, 47.67399] }
    }
```

- Step 1 requires you to complete the getting started tutorial noted earlier in this article so that Azure Search is provisioned at the portal. Recall that you'll need the URL and the API-KEY.
- Step 2 requires you to download and install the Node.js runtime locally on your development computer. You can find this at nodejs.org/download. My install ended up in the folder c:\program files\nodejs. It's recommended you get a basic "hello world" running in Node.js before proceeding further.
- Step 3 requires you to confirm the Node Package Manager (NPM) is properly installed and configured. NPM lets you install Node.js applications (JavaScript) available on the NPM registry.
- Step 4 involves installing the elasticsearch package, which simplifies writing code to communicate with Azure Search.

Once you've completed these steps, you're ready to return to the command prompt, navigate to whatever directory you like and start writing code. If you encounter some errors with NPM, you may need to validate some environment variables:

```
C:\node>set nodejs=C:\Program Files\nodejs\
C:\node>set node_path=C:\Program Files\nodejs\node_modules\*
C:\node>set npm=C:\Program Files\nodejs\
```

## Build Out the Node.js Solution

Now you're ready to develop some Node.js code to execute on your local system to illustrate communicating with Azure Search. Node.js makes it easy to insert and query data in Azure Search. Assume you have an Azure Search URL of terkaly.search.windows.net. You'd get this from the Azure Management Portal. You'll also need your API-KEY, which for this example is B7D12B8CA3D018EC09C754F95CA552D2.

There's more than one way to develop Node.js applications on your local computer. If you love the debugger in Visual Studio, then you'll want to use the Node.js Tools for Visual Studio plug-in (nodejstools.codeplex.com). If you like the command line, check out Nodejs.org. Once you install Node.js, it's important to integrate the NPM. This lets you install Node.js applications available on the NPM registry. The core package used here is called request.

```
      ]
    }
  };

  ////////////////////////////////////////////////////
  // OPTIONS FOR HTTP GET
  // Purpose:    Used to do a perform a query
  ////////////////////////////////////////////////////
  var optionsGET = {
    url: 'https://terkaly.search.windows.net/indexes/hotels/
      docs?search=motel&facet=category&facet=rating,
      values:1|2|3|4|5&api-version=2014-07-31-Preview',
    method: 'GET',
    json: true,
    headers: {
      'api-key': 'B7D12B8CA3D018EC09C754F95CA552D2',
      'Content-Type': 'application/json'
    },
    body: {
    }
  };
  request(optionsPUT, callbackPUT);

  ////////////////////////////////////////////////////
  // Purpose:    Used to create an index
  // Http Verb:  PUT
  // End Result: Defines an index using the fields
  // that make up the index definition.
  ////////////////////////////////////////////////////
  function callbackPUT(error, response, body) {
    if (!error) {

      try {
        if (response.statusCode === 204) {
            console.log('***success in callbackPUT***');
            request(optionsPOST, callbackPOST);
        }
      } catch (error2) {
        console.log('***Error encountered***');
        console.log(error2);
      }

    } else {
      console.log('error');
      console.log(error);
    }

  }


  ////////////////////////////////////////////////////
  // Purpose:    Used to insert data
  // End Result: Inserts a document
  ////////////////////////////////////////////////////
  function callbackPOST(error, response, body) {
    if (!error) {

      try {
        var result = response.request.response.statusCode;
        if (result === 200) {
            console.log('***success in callbackPOST***');
            console.log("The statusCode = " + result);
          // Perform a query
          request(optionsGET, callbackGET);
        }
      } catch (error2) {
        console.log('***Error encountered***');
        console.log(error2);
      }

    } else {
      console.log('error');
      console.log(error);
    }
  }
  ///////////////////////////////////////////////////////////////
  // Purpose:    Used to retrieve information
  // Http Verb:  GET
  // End Result: Query searches on the term "motel" and retrieves
  // facet categories for ratings.
  ///////////////////////////////////////////////////////////////
  function callbackGET(error, response, body) {
    if (!error) {
      try {
        var result = response.request.response.statusCode;
        if (result === 200) {
            result = body.value[0];
            console.log('description = ' + result.description);
            console.log('hotel name = ' + result.hotelName);
            console.log('hotel rate = ' + result.baseRate);
        }
        console.log('***success***');
      } catch (error2) {
        console.log('***Error encountered***');
        console.log(error2);
      }

    } else {
      console.log('error');
      console.log(error);
    }

  }
```

The code in **Figure 3** is straightforward. It does the same thing as described in the tutorial at bit.ly/1Ilh6vB, the only difference being you've implemented this code and Node.js using the request package. The code covers some of the more general use cases, such as creating an index, inserting data and, of course, performing queries. There are a number of callbacks here that define a schema, insert data and query data.

The callback chain is straightforward, as well. It starts with a simple GET, then moves to a PUT, POST and a second GET (with the query). It demonstrates the core operations you'd use with Azure Search. First, create a schema for the documents you'll add later. Use a PUT HTTP verb to define a schema. Next, use a POST to insert data. Finally, use a GET to query the data.

## Wrapping Up

The goal in this article was to expose some of the exciting things happening out there in the world of startups, specifically in the area of social networking analytics. You can use Azure Search as one piece in a larger solution where you need a sophisticated and powerful search experience to integrate with your Web site and applications. It lets you use fine-tuned ranking models to tie search results to business goals, as well as reliable throughput and storage. ■

**Bruno Terkaly** *is a principal software engineer at Microsoft with the objective of enabling development of industry-leading applications and services across devices. He's responsible for driving the top cloud and mobile opportunities across the United States and beyond from a technology-enablement perspective. He helps partners bring their applications to market by providing architectural guidance and deep technical engagement during the ISV's evaluation, development and deployment. He also works closely with the cloud and mobile engineering groups, providing feedback and influencing the roadmap.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Liam Cavanagh, Simon Gurevich, Govind Kanshi, Raj Krishnan, Venugopal Latchupatula, Eugene Shvets*

# Gradient Descent Training Using C#

My informal definition of machine learning (ML) is a system that uses data to make predictions. Anyone who starts investigating ML quickly encounters the somewhat mysterious phrase "gradient descent." In this article, I'll explain what gradient descent is and demonstrate how to use it to train a logistic regression classification system.

To get an idea of where this article is headed, take a look at the demo program in **Figure 1**. The demo begins by generating 10,000 synthetic data items. Using artificial data rather than real data is often useful when investigating ML because you can control the characteristics of the data. Each data item has eight predictor variable values (often called features in ML terminology) followed by a single dependent variable, which can be either 0 or 1. The data was generated by using eight random weight values (-7.78, -0.65, ... -7.97) plus an additional constant (-5.02).

You can imagine that the synthetic data corresponds to a problem where you're trying to predict the sex (male = 0, female = 1) of a person based on eight features such as age, annual income, credit score, and so on, where the feature values have all been scaled so they fall between -10.0 and +10.0.

After generating the 10,000 data items, the demo randomly splits that data into an 8,000-item set to be used to train a classifier, and a 2,000-item set to be used to estimate the predictive accuracy of the resulting model. Next, the demo creates a logistic regression binary classifier and then prepares for gradient descent training by setting values for variables maxEpochs (1,000) and learning rate (0.01). Gradient descent is an iterative process and variable maxEpochs sets a limit on the number of iterations. I'll explain the learning rate parameter later, but for now you can think of the learning rate as a value that controls how much change occurs in the logistic regression classifier model in each training iteration.

The demo trains the classifier and displays the error of the model on the training data, every 100 iterations. Gradient descent can be used in two different ways to train a logistic regression classifier. The first, more common, approach is called "stochastic" or "online" or "incremental." (ML vocabulary is chaotic.) The second approach is called "batch" or "offline." I'll describe both approaches later, but the demo program uses the stochastic gradient descent training approach.

After training completes, the demo displays the best weight values found (-9.84, -14.88, ... -15.09). Notice the model weights are all about twice as large as the weights used to generate the random data. The demo program computes the accuracy of the resulting model on the training data (99.88 percent, or 7,990 out of 8,000 correct) and the accuracy on the test data (99.80 percent, or 1,996 out of 2,000 correct).

**Code download available at msdn.microsoft.com/magazine/msdnmag0315.**

## Logistic Regression Classification

Logistic regression classification is best explained using a concrete example. Suppose you want to predict the sex of a person (male = 0, female = 1) based on age (x1), annual income (x2) and education level (x3). If Y is the predicted value, a logistic regression model for this problem would take the form:

$$Z = b0 + b1(x1) + b2(x2) + b3(x3)$$
$$Y = 1.0 / (1.0 + e\text{^}{-Z})$$

Here, b0, b1, b2 and b3 are weights, which are just numeric values that must be determined. In words, you compute an intermediate value Z that is the sum of input values times b-weights, add a b0 constant, then pass the Z value to the equation that uses math constant e. The equation is called the logistic sigmoid function. Notice that each input variable (xi) has an associated weight (bi), and that there's an additional weight (b0) not associated with any input.

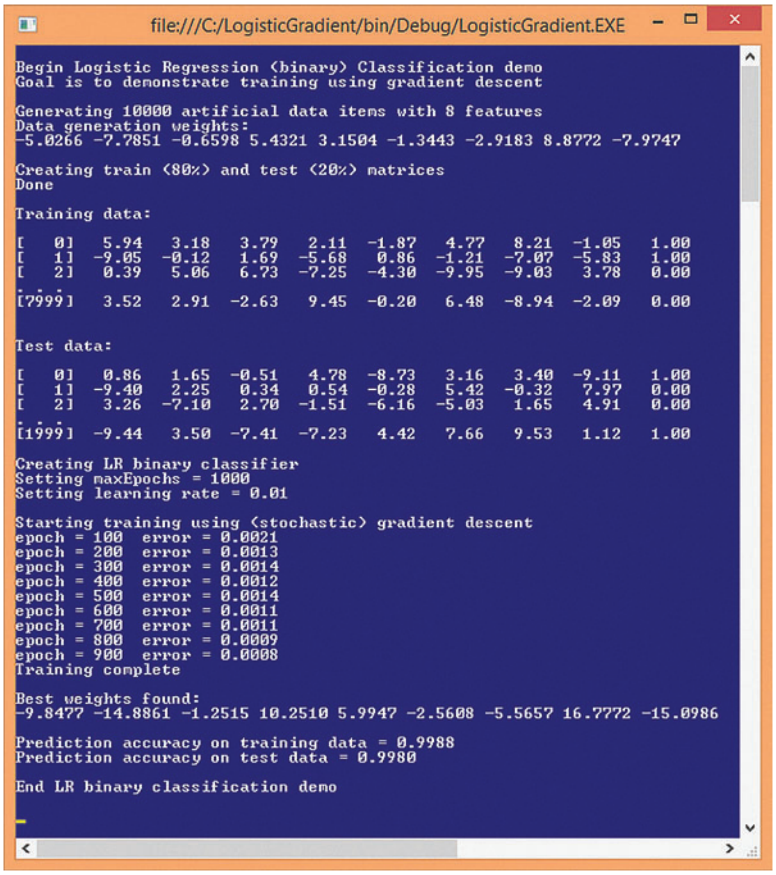
It turns out Y will always be between 0 and 1. If Y is less than 0.5 (closer to 0), you conclude the predicted output is 0 and if Y is greater than 0.5, you conclude the output is 1. If there are n features, there will be n+1 b-weights. Not all data can be modeled using logistic regression, but because it's one of the simplest classification techniques, logistic regression is a good place to start.

Suppose a person has a scaled age of x1 = +0.80 (older than average), annual income of x2 = -0.50 (slightly less than average) and education level x3 = -1.00 (less than average). And suppose that b0 = 3.0, b1 = -2.0, b2 = 2.0 and b3 = 1.5. Then Z = 3.0 + (-2.0)(0.80) + (2.0)(-0.50) + (1.5)(-1.00) = -1.10 and so Y = 1.0 / (1.0 + e^-(-1.10)) = 0.25. Because Y is closer to 0 (less than 0.5) than to 1, you'd predict the person is male.

Here's the demo code that implements computing logistic regression output:

```
public double ComputeOutput(double[] dataItem, double[] weights)
{
  double z = 0.0;
  z += weights[0]; // Add b0 constant
  for (int i = 0; i < weights.Length - 1; ++i)
    z += (weights[i + 1] * dataItem[i]); // Skip b0
  return 1.0 / (1.0 + Math.Exp(-z)); // Logistic sigmoid
}
```

The question is, where do the b-weights come from? The process of determining the values of the b-weights is called training the model.


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

```
Begin Logistic Regression (binary) Classification demo
Goal is to demonstrate training using gradient descent

Generating 10000 artificial data items with 8 features
Data generation weights:
-5.0266 -7.7851 -0.6598 5.4321 3.1504 -1.3443 -2.9183 8.8772 -7.9747

Creating train (80%) and test (20%) matrices
Done

Training data:

[   0]   5.94   3.18   3.79   2.11  -1.87   4.77   8.21  -1.05   1.00
[   1]  -9.05  -0.12   1.69  -5.68   0.86  -1.21  -7.07  -5.83   1.00
[   2]   0.39   5.06   6.73  -7.25  -4.30  -9.95  -9.03   3.78   0.00
. . .
[7999]   3.52   2.91  -2.63   9.45  -0.20   6.48  -8.94  -2.09   0.00


Test data:

[   0]   0.86   1.65  -0.51   4.78  -8.73   3.16   3.40  -9.11   1.00
[   1]  -9.40   2.25   0.34   0.54  -0.28   5.42  -0.32   7.97   0.00
[   2]   3.26  -7.10   2.70  -1.51  -6.16  -5.03   1.65   4.91   0.00
. . .
[1999]  -9.44   3.50  -7.41  -7.23   4.42   7.66   9.53   1.12   1.00

Creating LR binary classifier
Setting maxEpochs = 1000
Setting learning rate = 0.01

Starting training using (stochastic) gradient descent
epoch = 100  error = 0.0021
epoch = 200  error = 0.0013
epoch = 300  error = 0.0014
epoch = 400  error = 0.0012
epoch = 500  error = 0.0014
epoch = 600  error = 0.0011
epoch = 700  error = 0.0011
epoch = 800  error = 0.0009
epoch = 900  error = 0.0008
Training complete

Best weights found:
-9.8477 -14.8861 -1.2515 10.2510 5.9947 -2.5608 -5.5657 16.7772 -15.0986

Prediction accuracy on training data = 0.9988
Prediction accuracy on test data = 0.9980

End LR binary classification demo
```

Figure 1 **Training a Logistic Regression Classifier Using Gradient Descent**

The idea is to use a set of training data that has known input and output values, and then try different values for the b-weights until you find a set of values that minimizes the error between computed outputs and the known, correct output values (often called the target values, or the desired values).

Finding the weight values that minimize error is difficult and there are many numerical optimization algorithms you can use. Each algorithm has different strengths and weaknesses. The most common optimization algorithms include simplex optimization, L-BFGS optimization, particle swarm optimization, iterated Newton-Raphson, plus about a dozen others. The most fundamental optimization algorithm is called gradient descent.

## Understanding Gradient Descent

Let me try to explain gradient descent from a software developer's point of view. I'll take liberties with my explanation and terminology in order to make the ideas as clear as possible. Take a look at the graph in **Figure 2**. The graph plots error as a function of the value of some weight. As the value of a weight changes, the resulting logistic regression classifier's error will change. The goal is to find the weight value where error is at a minimum. For **Figure 2**, this would be w = 5.0. Here, I use w to indicate any of the b-weights. Note that there'd be one graph like the one in **Figure 2** for each weight.

If you knew the shape of all the error graphs, determining each weight would be easy. But, unfortunately, you don't know the shape of any error graph. You might think you could just try every possible weight value and then compute the resulting error, but there are an infinite number of possible weight values.

A calculus derivative of a function at some point is the slope of the tangent line at the point. A slope has a sign (+ or -) that indicates the direction of the tangent line, and a magnitude that indicates the steepness of the tangent. For example, in **Figure 2**, when w = 7.0, the slope of the tangent line (in other words, the derivative) is +2.15 (from upper right to lower left, and steep). When w = -5.0, the derivative is -0.90 (from upper left to lower right, and not too steep).

Each individual derivative is called a partial derivative (or just a "partial" for brevity) because there are derivatives for each weight. A gradient is the collection of all the partial derivatives. In casual usage, the terms gradient and partial derivative are often used interchangeably, mostly because a phrase like, "the partial derivative of the error function with respect to weight b2," is a lot harder to say or write than, "the gradient." Partial derivatives are often indicated using a special math symbol that resembles a backward 6.

So, what's the point? If you look closely at **Figure 2**, you'll see that a partial derivative can be used to move from a given weight value toward the weight value where error is minimized. The sign of the partial indicates the direction to move, and the magnitude of the partial gives a hint of how far to move; a larger magnitude means you can consider moving farther than a smaller magnitude. This technique is called gradient descent because you're going down the error function toward a minimum value.

OK, so far so good, but how does this idea translate into usable code? Or, put another way, what is the pseudo-code to update a logistic regression weight? There are many resources on the Internet that show some fairly sophisticated calculus to derive the weight-update rule. The end result is:

```
wj = wj + a * (target - computed) * xj
```

Finding the weight values that minimize error is difficult and there are many numerical optimization algorithms you can use

In words, "for the jth weight, the new weight value is the old weight plus the product of a constant 'a,' times the difference between the target value in the training data and the computed output value, times the feature (input) value associated with the jth weight." The update rule is often written using Greek letters with theta (θ) for the weight, and alpha (α) for the constant. The constant "a" is often called the learning rate.

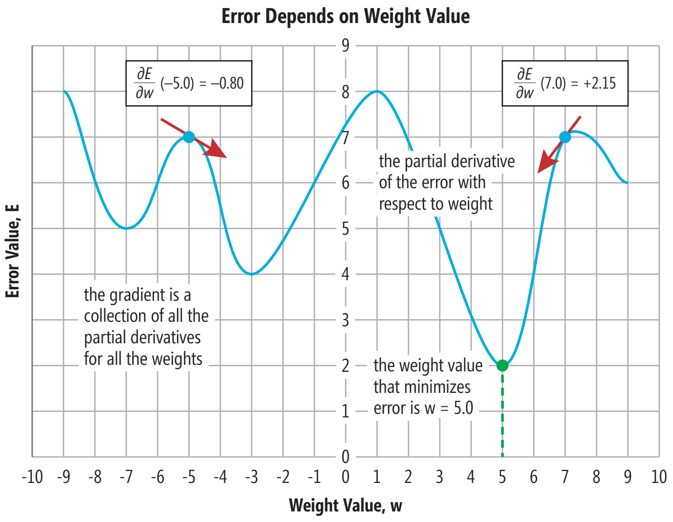


Figure 2 **Partial Derivatives and Gradient Descent**

Suppose you're working with a weight where j = 2, b2. And suppose the current value of b2 is 0.50. If, for some training data item, the known target value is 1 and the computed output value (using all x values) is 0.80, and x2 (the single input value that corresponds to weight b2) has value 3.0, and the learning rate is 0.10, then the new b2 weight is:

b2 = 0.50 + 0.10 * (1 - 0.80) * 3.0
b2 = 0.50 + 0.06
b2 = 0.56

The update rule is applied iteratively until a stopping condition is met. It's almost too simple. Notice that the computed output (0.80) was too small compared to the target output (1.0), so the weight update rule increased the value of the weight. This will have the effect of increasing the computed output value on the next iteration of training. If the computed output had been too large compared to the target output, the weight update rule would've reduced the weight. Very neat!

There are two main ways to derive the weight update rule. The most common approach you'll see in references on the Internet starts by defining the probability of getting a set of training data for a given set of weight values, and then uses a fairly complicated calculus technique called maximum likelihood expectation to find the values of the parameters that maximize the probability of the observed data.

An alternate approach starts by defining what's meant by error, using either two common error definitions—the sum of squared deviations error or the cross entropy error. This approach then uses calculus to find the set of weight values that minimizes error. When starting with cross entropy error, the resulting weight update rule is identical to the rule generated by maximizing probability. When starting with sum of squared deviations error, the resulting update rule has two additional terms:

wj = wj + a * (target - computed) * xj * computed * (1 - computed)

In the alternate update rule, because the computed term is always between 0 and 1, the product of computed and (1 - computed) will always be between 0 and 0.25, which means that updating weights using the alternate update rule just takes smaller steps than the simpler form. In practice, both update rules give similar results so the simpler form is almost always used. Luckily, you don't need to know how to derive the weight update rule in order to train a logistic regression classifier.

The probability-derivation approach maximizes a probability by going up a gradient, so it's called gradient ascent. The error derivation approach minimizes an error by going down a gradient and is called gradient descent. The point is that you'll see training a logistic regression classifier using a gradient referred to as both the gradient descent technique and the gradient ascent technique. Both terms refer to the same weight update rule.

## Demo Program Structure

The structure of the demo program, with some minor edits to save space, is presented in **Figure 3**. To create the demo, I launched Visual Studio and selected the C# Console Application template. I named the project LogisticGradient. The demo has no significant .NET dependencies so any version of Visual Studio will work. The demo is too long to present in its entirety, but all the source code is available in the download that accompanies this article. I removed all normal error checking to keep the main ideas as clear as possible.

After the template code loaded, in the Solution Explorer window, I right-clicked file Program.cs and renamed it to the more descriptive LogisticGradientProgram.cs and Visual Studio automatically renamed class Program for me. In the editor window, at the top of the source code, I deleted all unneeded using statements, leaving just the one referencing the top level System namespace.

The LogisticGradientProgram class contains helper methods MakeAllData, MakeTrainTest, ShowData, and ShowVector, which create and display the synthetic data. All the classification logic is contained in program-defined class LogisticClassifier. The Main method creates the synthetic data with these statements:

```
int numFeatures = 8;
int numRows = 10000;
int seed = 1; // Arbitrary
double[][] allData = MakeAllData(numFeatures, numRows, seed);
```

Method MakeAllData is essentially logistic regression classification in reverse. The method generates random weights and then iteratively generates random input values, combines the weights and input values using the logistic sigmoid function, and calculates the corresponding output value. The method doesn't add any random noise to the data, which means that, in theory, 100 percent prediction accuracy is possible. The synthetic data is split into training and test sets, like so:

```
double[][] trainData;
double[][] testData;
MakeTrainTest(allData, 0, out trainData, out testData);
ShowData(trainData, 3, 2, true);
ShowData(testData, 3, 2, true);
```

Method MakeTrainTest uses a hardcoded 80 percent to 20 percent train-test split. You might want to pass the training percentage as a parameter. The logistic regression classifier is created and trained with:

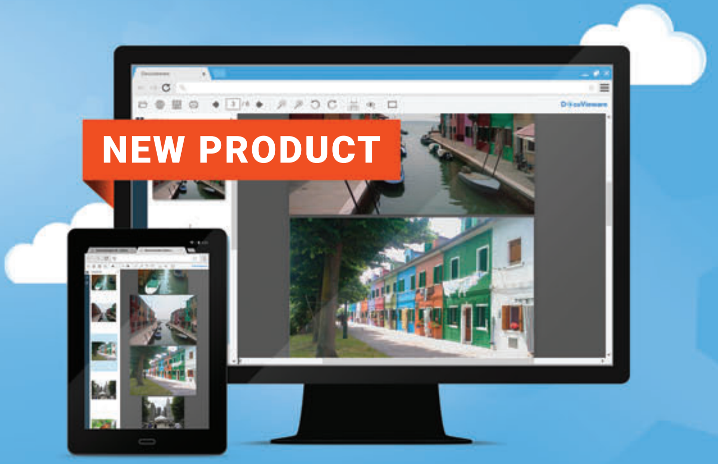
```
LogisticClassifier lc = new LogisticClassifier(numFeatures);
int maxEpochs = 1000;
double alpha = 0.01; // Learning rate
double[] weights = lc.Train(trainData, maxEpochs, alpha);
ShowVector(weights, 4, true);
```

The values for training parameters maxEpochs and alpha (the learning rate) were determined by trial and error. Tuning most

# DocuVieware

## Universal HTML5 Viewer and Document Management Kit

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



## Why **DocuVieware**?

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

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

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

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

**Mobile** devices optimization & **responsive** design.

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

## Works in all modern browsers ...

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

## ... so it works on all devices.

PC, Mac, tablets and smartphones.

Powered by **GdPicture**.NET (11)

www.docuvieware.com

ML training methods typically requires some experimentation to get good prediction accuracy. The quality of the trained model is evaluated like so:

```
double trainAcc = lc.Accuracy(trainData, weights);
Console.WriteLine(trainAcc.ToString("F4"));
double testAcc = lc.Accuracy(testData, weights);
Console.WriteLine(testAcc.ToString("F4"));
```

The accuracy of the model on the test data is the more relevant of the two accuracy values. It provides you with a rough estimate of how accurate the model would be when presented with new data with unknown output values.

Figure 3 **Demo Program Structure**

```
using System;
namespace LogisticGradient
{
  class LogisticGradientProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Begin classification demo");
      Console.WriteLine("Demonstrating gradient descent");
      ...
      Console.WriteLine("End demo");
      Console.ReadLine();
    }

    static double[][] MakeAllData(int numFeatures,
      int numRows, int seed) { . . }

    static void MakeTrainTest(double[][] allData, int seed,
      out double[][] trainData, out double[][] testData) { . . }

    static void ShowData(double[][] data, int numRows,
      int decimals, bool indices) { . . }

    static void ShowVector(double[] vector,
      int decimals, bool newLine) { . . }
  }

  public class LogisticClassifier
  {
    private int numFeatures;
    private double[] weights;
    private Random rnd;

    public LogisticClassifier(int numFeatures) { . . }
    public double[] Train(double[][] trainData,
      int maxEpochs, double alpha) { . . }
    private void Shuffle(int[] sequence) { . . }
    private double Error(double[][] trainData,
      double[] weights) { . . }
    private double ComputeOutput(double[] dataItem,
      double[] weights) { . . }
    private int ComputeDependent(double[] dataItem,
      double[] weights) { . . }
    public double Accuracy(double[][] trainData,
      double[] weights) { . . }
  }
}
```

Figure 4 **The Train Method When Using Batch Training**

```
double[] accumulatedGradients = new double[weights.Length];
for (int i = 0; i < trainData.Length; ++i)  // Accumulate
{
  double computed = ComputeOutput(trainData[i], weights);
  int targetIndex = trainData[i].Length - 1;
  double target = trainData[i][targetIndex];
  accumulatedGradients[0] += (target - computed) * 1; // For b0
  for (int j = 1; j < weights.Length; ++j)
    accumulatedGradients[j] += (target - computed) * trainData[i][j - 1];
}
for (int j = 0; j < weights.Length; ++j) // Update all wts
  weights[j] += alpha * accumulatedGradients[j];
```

## Implementing Gradient Descent Training

The definition of method Train begins with:

```
public double[] Train(double[][] trainData, int maxEpochs, double alpha)
{
  int epoch = 0;
  int[] sequence = new int[trainData.Length];
  for (int i = 0; i < sequence.Length; ++i)
    sequence[i] = i;
...
```

Variable epoch is the training loop counter variable. The array named sequence is initialized to the indices into the training data. The idea here is that the training data will be processed in a different, random order on every iteration. Next, the main weight-update loop begins:

```
while (epoch < maxEpochs)
{
  ++epoch;
  if (epoch % 100 == 0 && epoch != maxEpochs)
  {
    double mse = Error(trainData, weights);
    Console.Write("epoch = " + epoch);
    Console.WriteLine(" error = " + mse.ToString("F4"));
  }
  Shuffle(sequence); // Process data in random order
...
```

A measure of error is calculated and displayed every 100 epochs. Method Error returns the mean squared error, which is the average of the sum of squared differences between computed and target output values. Note that this is slightly different from the definition of error that's the basis of the gradient descent weight update rule. When using gradient descent training, the error is implicitly used, but isn't used directly. Other training techniques, in particular particle swarm optimization, use the error explicitly. Method Shuffle scrambles the training data indices contained in the sequence array using the Fisher-Yates algorithm.

The heart of gradient descent training is short:

```
for (int ti = 0; ti < trainData.Length; ++ti)
{
  int i = sequence[ti];
  double computed = ComputeOutput(trainData[i], weights);
  int targetIndex = trainData[i].Length - 1;
  double target = trainData[i][targetIndex];
  weights[0] += alpha * (target - computed) * 1;
  for (int j = 1; j < weights.Length; ++j)
    weights[j] += alpha * (target - computed) * trainData[i][j - 1];
}
```

First, the next random-order training item is identified using the scrambled sequence array. Method ComputeOutput uses the current weights to compute the output for the current set of weight values, which will be a value between 0.0 and 1.0. The target value, 0 or 1, is extracted from the current training item. The b0 weight is updated first. Recall that the weight update rule uses the input value associated with the weight being modified. However, the b0 weight isn't associated with any actual input. To deal with this, logistic regression b0 weights are said to have a dummy input value always equal to 1.0. The demo code multiplies by the 1.0 value, which obviously has no effect, to illustrate the similarity between updating b0 and updating any other b-weight. Updating the regular b-weights is pretty straightforward and just requires some attention to the indexing details. Method Train concludes with:

```
  ...
} // While loop
  return this.weights;
} // Train
```

The method returns a reference to the actual weights in the LogisticClassifier weights array. For safety, you might want to consider creating a results array, then copying the weights into that array and returning a reference to the array.

As noted earlier, the demo uses stochastic gradient descent. As each training item is encountered, the gradient for that one training item is calculated and used to update all weights. In batch gradient descent, in contrast, in each iteration gradients are accumulated over all training items first, and then the weights are updated. To use batch training, the heart of method Train would become the code shown in **Figure 4**.

With batch training, because all training items are processed before any weights are updated, there's no advantage to processing the training data in random order.

With batch training, because all training items are processed before any weights are updated, there's no advantage to processing the training data in random order.

When gradient descent training was first conceived, the batch approach was considered theoretically preferable because that technique uses all available information to find the weight gradient. However, ML practitioners quickly realized that training speed could be increased by using the gradient for just a single training item as an estimate for the overall gradient. In other words, stochastic (which means randomly determined) gradient descent uses one gradient to estimate the overall gradient.

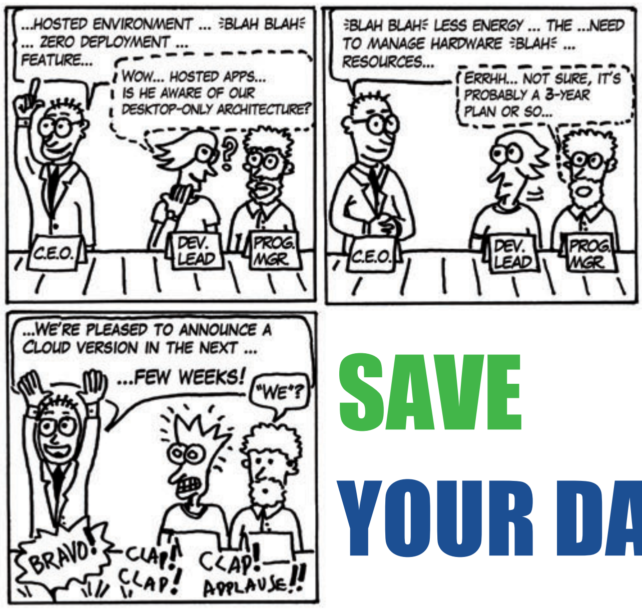
## Wrapping Up

Two related variations of basic gradient descent that are often used with logistic regression classifiers are called BFGS and L-BFGS. These two algorithms are an attempt to improve on basic gradient descent, at the expense of significantly increased complexity.

In addition to logistic regression classification, gradient descent can be used with several other ML techniques. In particular, gradient descent can be used to train a neural network. When gradient descent is used with neural networks, the technique is called back-propagation. ■

**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 technical expert at Microsoft Research for reviewing this article: *Richard Hughes*


...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...
...WE'RE PLEASED TO ANNOUNCE A CLOUD VERSION IN THE NEXT ...
...FEW WEEKS!
"WE"?
BRAVO!
CLAP! CLAP! CLAP! APPLAUSE!!

**SAVE YOUR DAY!**

SUPPORT FOR VISUAL STUDIO 2015 PREVIEW

**Use CodeFluent Entities**

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

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

Renato Xavier - Colombaroli - Brazil

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

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

CodeFluent Entities
tools for developers, by developers

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

# Design and Develop Accessible Modern Apps

What value is there in technology—the Internet, apps and various forms of media—unless there's a human benefit? Unfortunately, there's a lot of software out there that doesn't benefit all people. Designers, management and developers often pay more attention to security and performance than accessibility. Accessibility is usually a low priority in software, if it's even a priority at all.

Accessibility should be a higher priority. WebAIM estimates 20 percent of Web users have accessibility needs or rely on assistive technologies. That's more than 60 million people in the United States alone who may have difficulty using your Web site or app or consuming your content. For most Web sites and apps, lost customers equates directly to lost revenue. My PDF, "The Importance of Accessibility," illustrates the many other reasons for an accessible design (download it at bit.ly/1Clx4k4).

## The ABCs of Accessibility

When designing and developing with accessibility in mind, consider the broad categories of disability:

- **Visual:** People with visual impairments range from low vision to blindness, including a spectrum of color blindness.
- **Hearing:** Hearing impaired people might be hard of hearing or fully deaf.
- **Motor:** There are many people with motor disabilities. Some have suffered complete loss of or use of limb. Others may have neuropathy from an accident or illness. People with a motor impairment might need specialized input devices altogether.
- **Cognitive:** People with learning disabilities, including ADHD and dyslexia, often have difficulty consuming information, depending on its presentation.

Many people require some form of assistive technology. Assistive technology is any tool or technology used to assist in daily activities to make them easier or possible. People rarely consider eyeglasses to be an assistive technology, but they are—albeit at the low end of the technology scale. Some technology users can't function at all without their glasses. Common assistive technologies include braille readers, mouth sticks, head wands, adaptive keyboards, voice-recongnition software and so on.

## Accessible Content and Design

You'll find the most accessible design is often considered great design. Too many Web sites have too many ads jammed into the flow of the content, which greatly disrupts the reader's flow. Others have hard-to-use menus and navigation aspects. The layout and navigation of a Web site or app are important considerations when considering accessibility needs.

When organizing content, separate it into distinct sections with clear headings. Movies, music and animations should include captions or transcripts as part of their content. Most of the movie-making software available today lets you enter transcripts for closed captioning.

Good design also means a consistent and clear navigation scheme. JavaScript cascading menus are often difficult for users with no disability. They're worse for those with a motor impairment. Links across the top or down the side of a Web page work best. In phone apps, you'll want to go along with the navigation scheme of the target device. For more on navigation in Windows 8.x, see my August 2013 column, "Navigation Essentials in Windows Store Apps" (msdn.microsoft.com/magazine/dn342878).

Fonts should be large and clear. Script-like fonts are more difficult to read. There are many things that can cause some letters from a font to be unclear. One is the learning disorder dyslexia, which causes an inability to distinguish between letters that look alike. An estimated 5 percent of the population has dyslexia. Letters that face one way to non-dyslexics look flipped and backward to dyslexics.

Using this knowledge about dyslexics, the folks over at dyslexiefont.com created a font that changes letters slightly so they're easier for dyslexics to read. So far, font testers report they love it. You might choose not to use the Dyslexie font and that's OK. It doesn't mean you're snubbing dyslexics. However, be sure to choose a font that is as easy to read.

You might want to rethink changing your font if your app targets entrepreneurs. Font colors should also have high contrast. A good example of high contrast is a light or white background with dark or black text. The complete opposite is true, too. Using light text on dark backgrounds also works well. Usually it's a matter of preference and style as to which to use, as long as there's sufficient contrast.

You can't go wrong with a mobile-first design strategy. Mobile-first designs generally force you to make the best possible use of space. Most phones and small tablets only have a few inches of space for content. This means the app usually only presents the most-critical information. In this scenario, there simply is no room for vertical ads.

There are other good design practices you can follow that also help ensure accessibility. Think of the most popular news and social sites on the Web. Many of them aren't accessible, or are only accessible to a small degree. They often contain popups that get in the way and can completely stop screen readers and humans alike. These same popups often have tiny close buttons that are difficult to find, and impossible to click or tap—even with mainstream devices.

Others employ serial popups. When you finally get rid of one popup; another takes its place. This is frustrating. Many people with accessibility needs refuse to visit these sites, as they just aren't worth the hassle.

Avoid strobe effects, flashing effects, or images flickering without warning. This poses a seizure risk. Plus, these are usually annoying unless they're specifically supposed to be an optical illusion or eye trick.

Don't user color alone to convey your message to users. For example, many forms use a red font to denote a required field. Colorblind people can't see that difference. Don't just use the phrase "click here" as link text. That doesn't help screen readers at all. Use something descriptive instead.

## Program Accessible Code

There are programming techniques you can use to develop accessible Web sites and apps. As a developer, you need to interact with both input and output. This means you should keep in mind that different people need different ways to interact with your software instead of just the mouse and keyboard.

For those who mostly or exclusively use the keyboard, ensure field tab orders are straightforward and in order. You should also label fields and elements such as buttons using the HTML <label> element. This ensures greater clarity. Images should have alt attributes set to something descriptive yet succinct. Screen readers can't magically read images, so they use the alt tag to describe the image to the listener.

HTML5 contains a set of elements called semantic elements. The point of these is so both machines and humans can easily read and understand HTML elements. Semantic elements describe their content much like XML. For example, anyone can understand the following semantic elements just by reading them: caption, figure, article, footer, header, summary, time, nav, mark and main, to name a few.

For those who rely on a screen reader, skip links are life savers. A skip link lets a reader pass over navigation elements and ads on a Web site and go directly to the desired content. Having to sit through multiple, excruciating and unnecessary iterations of options and ads is no fun. Don't force your users to sit through such a painful experience. Immediately following the <body> element is where the primary skip navigation link goes, as the following code shows:

```
<body>
<a href="#maincontent">Skip to main content</a>
<nav><!-- navigation here --></nav>
<main id="maincontent">
<p>The quick brown fox jumps over the lazy dog.</p>
```

When a screen reader sees the skip link, it focuses on the element the skip link refers to by its id. That means in the earlier example, the skip link points to the <main> element with an id of "maincontent." Because skip links must be the first element and many designers prefer the skip link go somewhere else, you can hide it, yet make it visible to assistive technology with some CSS, as shown in this code:

```
.skiplink-offscreen {
  position:absolute;
  left:-10000px;
  top:auto;
  width:1px;
  height:1px;
  overflow:hidden;
}
```

As you can see, it's an absolute positioned class selector with a value attribute set to -10000px. Its overflow is hidden. The code is positioning the element that uses this selector where the user will never see it, but a screen reader will. Screen readers skip over elements with the hidden or display attributes set to hidden or none. That's why you should use this small hack.

Figure 1 **Accessible HTML Form with ARIA Attributes**

```
<form>
  <div>
    <label for="name">* Name:</label>
    <input type="text" id="name" aria-required="true" />
  </div>
  <div>
    <label for="checkboxGroupLabel">* Language</span>
    <ul role="radiogroup" aria-labelledby="checkboxGLabel">
      <li role="checkbox"><input type="checkbox" value="C#"
        name="language" aria-checked="false" />C#</li>
      <li role="checkbox"><input type="checkbox" value="JavaScript"
        name="language" aria-checked="false" />JavaScript</li>
      <li role="checkbox"><input type="checkbox" value="Python"
        name="language" aria-checked="false" />Python</li>
    </ul>
  </div>
  <div>
    <span id="yearsLabel">* Years Experience</span>
    <select name="YearsExperience" aria-labelledby="yearsLabel" >
      <option value="1">1-5 Years</option>
      <option value="6">6-10 Years</option>
      <option value="10">10-20 Years</option>
      <option value="20">20+ Years</option>
    </select>
  </div>
  <div>
    <input type="submit" alt="Submit" />
  </div>
</form>
```

## Develop with ARIA

Accessible Rich Internet Applications (ARIA) are a set of standard attributes you can apply to markup such as HTML that help assistive technology work efficiently. With ARIA, you can define an element by its state, property or role. From that information, screen readers can determine what the software is doing.

For example, a checkbox may have an ARIA checked state. Or an element might assume the role of a menu. This bit of extra information about the state or role helps screen readers construct a better representation of the Web page's content. These attributes don't change the element, but they do make the element behave in a more semantic nature. Semantic markup is easier for both humans and machines to understand. For example, the following code shows an article with one section that's semantic, so a screen reader can better identify related content and convey it to the user:

```
<article>
<section aria-labelledby="ProgrammingBestPractices">
  <h2 id="ProgrammingBestPractices">Best Practices for Programmers</h2>
  <ol>
  <li>Take a few minutes a day to refactor small portions of code.</li>
  <li>Learn a new programming language every year.</li>
  </ol>
</section>
</article>
```

You can extend this semantic and accessible markup into HTML forms. Screen readers need to know what elements label or describe form fields, and which fields belong together as a group. The state of buttons and checkboxes are other things about which screen readers must know. Good form design means knowing how script affects assistive technology. You might want to reconsider code that automatically sets the focus to a control or dynamically changes the state of form controls.

Use the HTML <label> element to label individual form elements to single labels. However, there will be times when you need to label an element with multiple labels, such as in a table. That's no problem. The aria-labelledby attribute works in those cases. Set the aria-labelledby of each element in the group to the ids of element that will do the labelling. You can use both the <label> element and the aria-labelledby attribute in the same form. There's no need to label Submit or Reset buttons, unless they're image buttons. Then be sure to set the alt attribute.

Every form has required elements and elements with specific restrictions on the type of data they'll accept. Most forms run some JavaScript to perform validation on required and constrained fields. They do so before sending all the data back to the server for processing, as it's better to notify the user of errors sooner rather than later. Screen readers have a difficult time discerning what's happening in the page when a script is running. To work around this, use the aria-required and aria-invalid attributes, as shown in **Figure 1**.

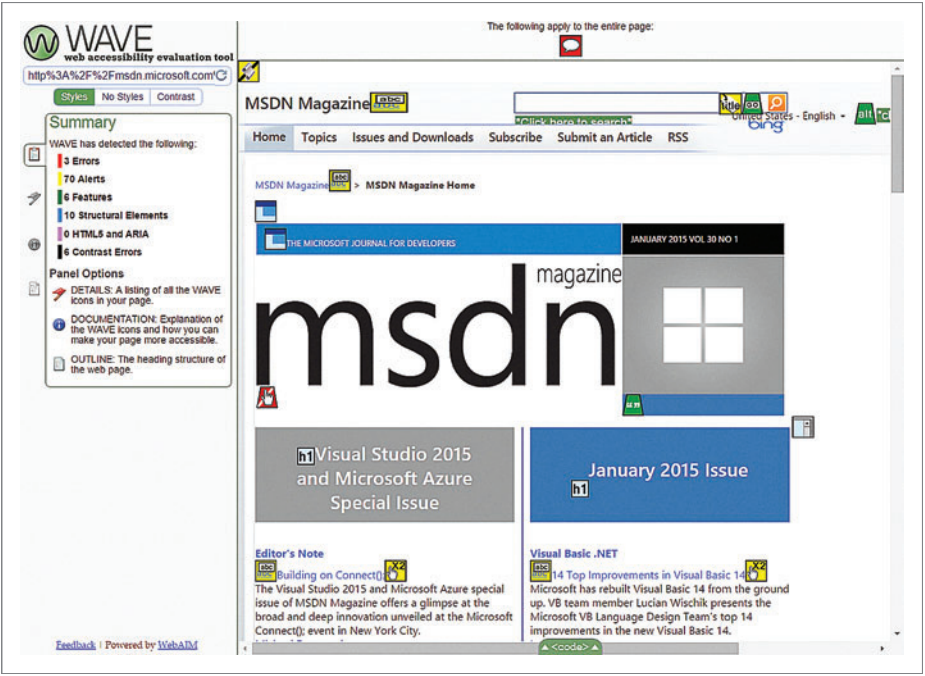
Also shown in **Figure 1** is an example of aria-required and aria-labelledby, as well as using roles. You're just adding some extra but unobtrusive HTML. It doesn't take much effort, and it provides a huge return.

Assistive technology must deal with more than static elements. JavaScript, AJAX calls and SPA-style apps all frequently change the contents of Web sites and apps. Sometimes dynamic script such as this gets directly in the way of screen readers. You must reset the state of some ARIA attributes, such as aria-invalid, after JavaScript validation runs.

## Test Accessible Code

Test your work by downloading screen-reading software and trying out assistive technologies for yourself. Close your eyes and use the screen reader as if you were blind. Getting feedback from your users who use assistive technology is also a great way to see how accessible you've built software. Some screen readers you can download and use include:

- NVDA: nvaccess.org
- JAWS for Windows: bit.ly/1yJkxuV
- Window-eyes: gwmicro.com
- VoiceOver for OS X: bit.ly/1JuA1q9


WAVE
web accessibility evaluation tool
http%3A%2F%2Fmsdn.microsoft.com'
Styles
No Styles
Contrast
MSDN Magazine
*Click here to search*
United States - English
Home
Topics
Issues and Downloads
Subscribe
Submit an Article
RSS
MSDN Magazine > MSDN Magazine Home
THE MICROSOFT JOURNAL FOR DEVELOPERS
JANUARY 2015 VOL 30 NO 1
msdn magazine
Visual Studio 2015 and Microsoft Azure Special Issue
January 2015 Issue
<code>
Feedback | Powered by WebAIM

Summary

WAVE has detected the following:

- 3 Errors
- 70 Alerts
- 6 Features
- 10 Structural Elements
- 0 HTML5 and ARIA
- 6 Contrast Errors

Panel Options

DETAILS: A listing of all the WAVE icons in your page.

DOCUMENTATION: Explanation of the WAVE icons and how you can make your page more accessible.

OUTLINE: The heading structure of the web page.

The following apply to the entire page:

Editor's Note

Building on Connect();

The Visual Studio 2015 and Microsoft Azure special issue of MSDN Magazine offers a glimpse at the broad and deep innovation unveiled at the Microsoft Connect(); event in New York City.

Visual Basic .NET

14 Top Improvements in Visual Basic 14

Microsoft has rebuilt Visual Basic 14 from the ground up. VB team member Lucian Wischik presents the Microsoft VB Language Design Team's top 14 improvements in the new Visual Basic 14.

Figure 2 ***MSDN Magazine* After Running Through the WAVE Scanner**

Once you've tested for readers, WebAIM has a comprehensive checklist and page scanner called the Web Accessibility Evaluation tool (WAVE for short), that will report errors, warnings and other information about the accessibility status of a page. It's simple to use. Go to wave.webaim.org and enter the URL you want to scan. WAVE displays the target page inline and annotates trouble spots and elements you can change to improve accessibility. **Figure 2** shows the scanner in action as it flags some items on the *MSDN Magazine* January 2015 issue page.

Click on any of the annotated elements and WAVE will display error information or metadata about the element. The scanner looks at everything from missing labels and alt attributes to contrast problems. It flags them as errors or warnings so you can change the more important problems first. You can run a standalone version of WAVE, or you can add the API to your automatic QA processes.

## Wrapping Up

Accessible design is usually a better design approach for everyone. A non-accessible design misses out on a substantial percentage of the population, so it's simply a smart business decision to put in the effort to make your software accessible. UIs should present clearly labeled, easy-to-find options, and make steps to complete a task straightforward and simple. When you design for accessibility, you automatically get an easy-to-use and straightforward system that works well for everybody. ■

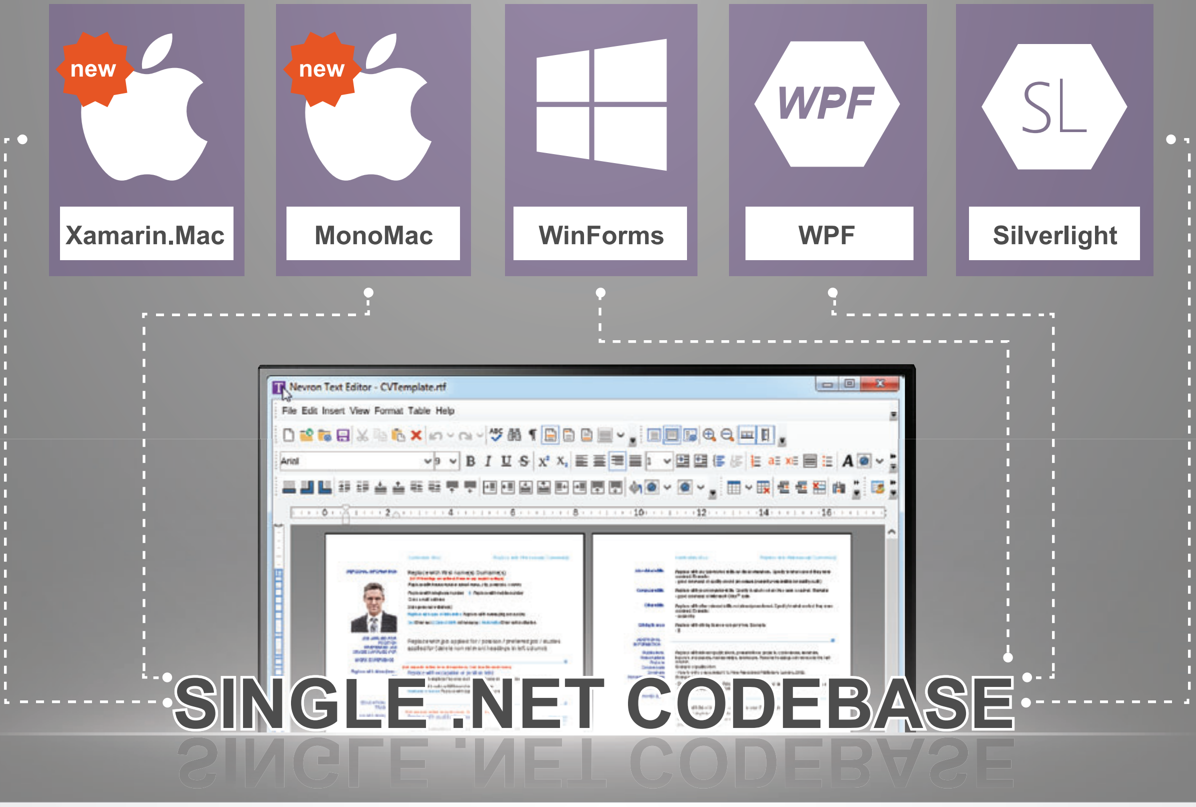
**Rachel Appel** *is a consultant, author, mentor and former Microsoft employee with more than 20 years of experience in the IT industry. She speaks at top industry conferences such as Visual Studio Live!, DevConnections, Mix and more. Her expertise lies within developing solutions that align business 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*

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

# TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

MICROSOFT HEADQUARTERS, REDMOND, WA

AUGUST 3 - 7, 2015

## ENGINEERED FOR YOU @ THE SOURCE

TechMentor is once again bringing its unique brand of independent, real-world training to the heart of it all – **Microsoft Headquarters in Redmond, WA.**

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

With educational sessions and workshops taught by 3rd party instructors, leading independent IT analysts and Microsoft team members, the topics covered are carefully vetted by long-time conference co-chairs Don Jones and Greg Shields to ensure their relevance and usability to IT Pros just like you!

EVENT SPONSOR: Microsoft

PLATINUM SPONSOR: GRIDSTORE

You owe it to yourself, your company and your career to be at TechMentor Redmond 2015!

## THE AGENDA WILL FEATURE:

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

## CONTENT AREAS INCLUDE:

- A Dedicated PowerShell Track
- Infrastructure Foundations
- Runbook Automation
- Configuration and Service Management
- Datacenter Provisioning and Deployment

## SAVE $300!

REGISTER TODAY

USE PROMO CODE **TMMAR2**

**TECHMENTOREVENTS.COM/REDMOND**

## CONNECT WITH TECHMENTOR!

Twitter / @TechMentorEvent

Facebook / Search “TechMentor”

LinkedIn / Search “TechMentor”

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

GOLD SPONSORS: GOVERLAN pluralsight

SUPPORTED BY: Redmond Magazine, Redmond Channel Partner, Virtualization Review

PRODUCED BY: 1105 MEDIA

# Hitting a New Wall

I really hate talking to a doctor's back. But that's what I did as this doctor took my history. "Any allergies?" he asked, back firmly turned toward me. "Have you had this before?" (back turned)—on and on, through his entire questionnaire. It would have been ludicrous if it weren't so sad.

The reason the doctor did this, as you've probably guessed, is that he had to enter all of my responses into his computer, to become part of my electronic medical record. U.S. federal incentives call for hospitals to be doing *something* with electronic medical records ("meaningful use"), though they don't specify exactly what. This priority inversion, valuing data input over patient contact, is one of the results.

Even the best usage of PCs by primary care physicians is highly intrusive. My daughters' pediatrician uses a laptop and faces her patients. But the device still creates a barrier, consumes the physician's time, diverts her clinical attention, forces her to repeatedly break eye and hand contact; not to mention transferring germs between patients via impossible-to-sterilize keyboards.

Computerizing operations was supposed to make users more efficient. In primary care medicine, we've accomplished exactly the opposite. As Katie Haffner wrote in *The New York Times*: "For decades, physicians pinned their hopes on computers to help them manage the overwhelming demands of office visits. Instead, electronic health records have become a disease in need of a cure, as physicians do their best to diagnose and treat patients while continuously feeding the data-hungry computer" (nyti.ms/1BMcdgl).

These demands rapidly consume the physician-user's hassle budget. As I wrote in April 2013 (msdn.microsoft.com/magazine/dn166939) and July 2011 (msdn.microsoft.com/magazine/hh288087), the hassle budget is the amount of extraneous effort that users will tolerate in order to get their computing jobs done. If an operation exceeds a user's hassle budget, he'll either toss that program or figure out a workaround. Writing down ever-changing passwords on sticky notes is the classic example.

Some physicians have created a workaround by dumping the data entry task onto another person, the *medical scribe*. The doctor faces and interacts directly with the patient (what a concept!). The scribe sits in the same room but apart, listens to the conversation, and enters the resultant data into the computer (see bit.ly/1y3URnV).

The scribe dresses in black to signal her invisibility, like the *kuroko* stagehands in traditional Japanese theater (bit.ly/1zA0scf). She does not speak, except to answer a doctor's question or request a clarification. Her sole function is to enter data into a computer so the doctor doesn't have to. I'd call her a liveware analog-to-digital converter.

Kathleen Myers is an emergency physician and founder of Scribes STAT, a company that provides medical scribe services to hospitals. Meyers says of working with a scribe: "I get to sit down and look at [my patient], and really focus on what they're saying. I feel like I miss less information—the patients have a greater bond with me and are able to share more information." (See a video about the service at bit.ly/1t2Ue25.)

Scribes only earn $10 to $15 per hour. Considering the physician's time they save, scribes would seem to quickly pay for themselves. Although, as I recall, that's what computer systems were supposed to do, and we're seeing exactly the opposite.

Computerizing operations was supposed to make users more efficient. In primary care medicine, we've accomplished exactly the opposite.

We could certainly build better medical apps. Involving the users from the project start would go a long way; instead of writing what we geeks think they need and cramming it down their protesting throats, as usually happens.

But I wonder: Have we reached the limits of what computer programs can accomplish? Quantum mechanics reached its limit with Heisenberg's uncertainty principle, and mathematical logic with Gödel's incompleteness theorem. Have we hit such a wall with medical software?

As my primary care physician (not the back-turner in the opening paragraph) Peter Zuromskis, M.D., likes to say, "Medicine is an analog process. Those bean counters are trying to make it digital, but at its core, it simply isn't." ■

**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 | Data Visualization | Spreadsheet |
|---|---|---|
| **ActiveReports** | **Studio Enterprise** | **Spread Studio** |
| 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. | 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. | 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®