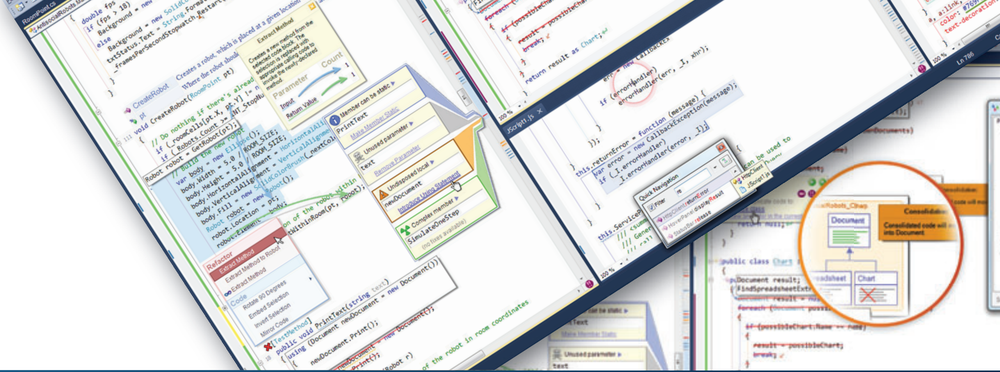
DevExpress



DXv2 Duplicate Code Detection & Consolidation with **CodeRush**

CodeRush helps you create and maintain source code with optimum efficiency. Consume-first declaration, powerful templates, smart selection tools, intelligent code analysis, innovative navigation and an unrivalled collection of refactorings all work together to increase your productivity dramatically.

Download the 30-day free trial to experience the next generation of developer productivity tools at **www.DevExpress.com**

Microsoft® GOLD CERTIFIED *Partner*

THE MICROSOFT JOURNAL FOR DEVELOPERS

MAY 2012 VOL 27 NO 5

msdn magazine

Microsoft

Start a Revolution

Refuse to choose between desktop and mobile.

With the brand new NetAdvantage for .NET, you can create awesome apps with killer data visualization today, on any platform or device.

Get your free, fully supported trial today!

www.infragistics.com/NET

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

Compatible with Microsoft® Visual Studio® 11 Beta dtSearch®

Instantly Search Terabytes of Text

- 25+ fielded and full-text federated search options
- dtSearch's own file parsers highlight hits in popular file and email types
- Spider supports static and dynamic data
- APIs for .NET, Java, C++, SQL, etc.
- Win / Linux (64-bit and 32-bit)

"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

- Desktop with Spider
- Network with Spider
- Publish (portable media)
- Web with Spider
- Engine for Win & .NET
- Engine for Linux

Ask about fully-functional evaluations!

The Smart Choice for Text Retrieval® since 1991

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

msdn magazine

MAY 2012 VOLUME 27 NUMBER 5

MITCH RATCLIFFE Director
LUCINDA ROWLEY Editorial Director/mmeditor@microsoft.com
PATRICK O'NEILL Site Manager

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

SCOTT SHULTZ Creative Director
JOSHUA GOULD Art Director

CONTRIBUTING EDITORS Dino Esposito, Joseph Fultz, Kenny Kerr, Julie Lerman, Dr. James McCaffrey, Ted Neward, John Papa, Charles Petzold, David S. Platt

RedmondMediaGroup

Henry Allain President, Redmond Media Group
Doug Barney Vice President, New Content Initiatives
Michele Imgrund Sr. Director of Marketing & Audience Engagement
Tracy Cook Director of Online Marketing

ADVERTISING SALES: 508-532-1418/mmorollo@1105media.com

Matt Morollo VP/Group Publisher
Chris Kourtoglou Regional Sales Manager
William Smith National Accounts Director
Danna Vedder Microsoft Account Manager
Jenny Hernandez-Asandas Director, Print Production
Serena Barnes Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA INC

Neal Vitale President & Chief Executive Officer
Richard Vitale Senior Vice President & Chief Financial Officer
Michael J. Valenti Executive Vice President

Christopher M. Coates Vice President, Finance & Administration
Erik A. Lindgren Vice President, Information Technology & Application Development
David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

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

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

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

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

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

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

List Rental: This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Merit Direct. Phone: 914-368-1000; E-mail: 1105media@meritdirect.com; Web: www.meritdirect.com/1105

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

Microsoft

BPA WORLDWIDE

Printed in the USA

LEADTOOLS®
DOCUMENT SDKs

LEADTOOLS Lorem ipsum dolor sit
amet, consectetur adipiscing elit. Ut
eget ultrices dui. Aliquam sodales nisl
sit amet ligula rutrum pretium. Ut nulla
nisi, bibendum vel lobortis vel, hendre-
rit quis turpis. Aliquam sed leo sit
amet purus porta euismod a in ipsum.
Aliquam viverra cursus lacus, vitae
tempor lorem pharetra id. Nulla in velit
sed nisl dignissim auctor sed dictum
dui. Suspendisse vestibulum
non felis dictum consectetur.
porta nulla non ante bibendu
quam enim placerat. Aliquam
lutpat. Fusce sit amet interdum augue.
Quisque lectus arcu, accumsan ut
mollis et, dapibus in felis. Cras vestibu-

OCR • BARCODE
PDF READ, WRITE, EXTRACT, VIEW & EDIT

DOCUMENT READERS & WRITERS • PREPROCESSING
FORMS RECOGNITION & PROCESSING • ANNOTATIONS

IMAGE FORMATS & COMPRESSION • SCANNING
VIRTUAL PRINTER

ZERO FOOTPRINT AND RICH CLIENT CONTROLS
FOR WEB, MOBILE & DESKTOP

C++, .NET, HTML5, SILVERLIGHT,
WEB SERVICES, WPF, ASP.NET, COM & CLOUD

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM

LEAD TECHNOLOGIES INCORPORATED

SALES@LEADTOOLS.COM
800.637.1840

MSDN Magazine Welcomes Script Junkie

Unless you've been living under a rock the past two years, you've noticed that Microsoft has gotten behind HTML5-based application development in a big way. And it's no surprise—we've been busy producing a steady diet of features, columns and Web articles focused on HTML5, JavaScript and CSS3.

This month includes another installment in our ongoing HTML5 feature series by Brandon Satrom, "Using HTML5 to Create Mobile Experiences." He explores two pillars of responsive Web design—fluid grids and flexible images. Also in this issue you'll find Colin Eberhardt's feature on developing native HTML5 applications for Windows Phone, using the open source Apache Cordova mobile development framework.

In February we launched the Client Insight column specifically to address growing interest in rich client frameworks and technologies, including HTML5 and XAML-based Silverlight and Windows Presentation Foundation (WPF). Written by John Papa, the column this month digs deeper into the JsRender JavaScript library and explores scenarios such as rendering external templates, changing context with the {{for}} tag and using complex expressions.

Yes, there's plenty of HTML5 and related coverage in each issue of *MSDN Magazine*. And there's more of that to be found on the *MSDN Magazine* Web site, including Rachel Appel's popular Web Dev Report column. Her latest piece offers helpful tips for working with CSS, and comes after a pair of articles focused on HTML5 forms.

Welcome Script Junkie

We've taken our efforts a big step further, welcoming the popular Microsoft Script Junkie site into the *MSDN Magazine* fold earlier this spring. Previously hosted on msdn.microsoft.com, Script Junkie has been helping Web developers work with HTML, CSS and JavaScript since 2009. Now Script Junkie's how-to and tutorial content will appear in its own section on the *MSDN Magazine* Web site. You can find the Script Junkie homepage at msdn.microsoft.com/magazine/ee729207. Or just look for the link on the *MSDN Magazine* homepage.

What does the transition mean for you? More than anything, it ensures that developers will continue to enjoy access to timely how-to articles from respected and established Script Junkie authors like Tim Kulp, Emily Lewis and Addy Osmani. What's more, the move helps introduce Script Junkie to the larger community of *MSDN Magazine* subscribers and site visitors. Our Web metrics have shown strong reader interest in HTML5-themed features and columns, and we expect Script Junkie will be valued by *MSDN Magazine* readers who are anxious to explore new topics and challenges in script-based Web development.

Are there specific topics or issues you'd like to see covered at Script Junkie? E-mail me at mmeditor@microsoft.com.

So Long and Thanks for All the Fish

No, the dolphins aren't leaving the planet (at least, not yet). But I'm sad to say that *MSDN Magazine* Editorial Director Kit George has moved on, assuming a new role with the Bing team that he described as "too good to refuse."

Kit's had a hand in a lot of the changes you've seen in the magazine—and especially the Web site—over the past year and a half. He campaigned for the addition of unique online editorial content, which you see today in the form of monthly features and online columns such as Web Dev Report and Bruno Terkaly's Windows Azure Insider. And it was Kit who jumped at the chance to bring Script Junkie under the *MSDN Magazine* banner.

Kit has moved *MSDN Magazine* forward in important ways, but the thing we'll miss most is his tenacity. He worked tirelessly to win the participation of key Microsoft product teams in the pages of *MSDN Magazine* and pushed hard to expand the boundaries of the publication. Good luck with the Bing team, Kit.

[illegible]

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

Brilliantly intuitive tools for dev and support teams.

Powered by HTML

OnTime Scrum

Agile project management & bug tracking

$7 per user per month

Small team? $10 per month for 10 users

- product backlogs, releases, and sprints
- powerful user story and bug tracking
- automated burndown charts
- visual planning board

OnTime Help Desk

Customer support for software apps

$7 per user per month

Small team? $10 per month for 10 users

- auto-generate tickets for support emails
- auto-responses & canned responses
- organize customers and contacts
- fully customizable Customer Portal

OnTime Team Wiki

Project wiki for collaborative dev teams

$7 per user per month

Small team? $10 per month for 10 users

- hierarchical document organization
- configurable security controls
- HTML and WYSIWYG editor
- table of contents and search support

Develop in Visual Studio?

Our new **OnTime Visual Studio Extension** brings the power of OnTime Scrum to your Visual Studio environment. Learn more at **ontimenow.com/addons**.

Visit OnTimeNow.com and...

Get started free with an OnTime OnDemand trial.

Download our pretty sweet **Scrum Diagram.**

axosoft axosoft.com • @axosoft • 800.653.0024

Understanding the Power of WebSockets

The current World Wide Web isn't designed to be a real-time medium. Web applications give the *impression* of continuous feel through classic polling solutions implemented via AJAX or perhaps via long-polling requests when effectively implemented by ad hoc libraries such as SignalR and Comet. For the needs of most applications, polling is a good solution, even though it might suffer from client-to-server and server-to-client latency. In this article I'll explore a new alternative called WebSockets.

The growing integration between Web and mobile applications with social media is lowering the threshold of tolerable delay in client/server interaction. When you update your Facebook status, you want that information to be immediately available to your friends. Similarly, when someone likes one of your posts, you want to be notified instantly. Today, all of these features are real, and this is just one of the reasons for the worldwide adoption of Facebook and for the explosion of the social network phenomenon. So in the end, there's a significant demand from developers of solutions and tools for implementing real-time communication over the Web.

Achieving zero-lag connectivity between Web clients and servers requires going beyond the HTTP protocol. This is just what the WebSocket Protocol provides. Currently an Internet Engineering Task Force standard exists for the WebSocket Protocol; you can read about it at bit.ly/va6qSS. A standard API for implementing the protocol is being formalized by the World Wide Web Consortium (W3C) for browsers to support it (see bit.ly/h1IsjB). The specification is in "Candidate Recommendation" status.

The WebSocket Protocol

The new WebSocket Protocol aims to overcome a structural limitation of the HTTP protocol that makes it inefficient for Web applications hosted in browsers to stay connected to the server over a persistent connection. The WebSocket Protocol enables bidirectional communication between Web applications and Web servers over a single TCP socket. Put another way, the protocol makes it possible for a Web application hosted in a browser to stay connected with a Web endpoint all the time while incurring minimal costs such as pressure on the server, memory and resource consumption. The net effect is that data and notifications can come and go between browsers and Web servers with no delay and no need to arrange for additional requests. As emphatic as it may sound, the WebSocket Protocol opens up a whole new world of possibilities to developers and makes polling-based tricks and frameworks a thing of the past. Well, not exactly.

Using WebSockets Today

Browser support for the WebSocket Protocol will improve quickly, but only the latest versions of browsers will support WebSockets, of course. Users who don't usually upgrade their browsers regularly (or aren't allowed to upgrade by strict corporate policies) will be left behind.

This means developers can't just abandon code based on AJAX polling or long-polling solutions. In this regard, it's relevant to note that SignalR—the upcoming Microsoft framework for zero-lag messaging between browsers and Web servers—does a fantastic job of abstracting a persistent connection, automatically switching to WebSockets where supported and using long polling in any other cases. I covered SignalR in recent columns, and once again I invite you to try it out as soon as possible if you haven't done so yet. SignalR has everything to be a winning library and a tool for every developer and any Web application.

Who Supports WebSockets Today?

Figure 1 provides a brief summary of the support for WebSockets that most popular browsers provide at present.

With the exception of Firefox, you can programmatically check for WebSockets support by looking at the window.WebSocket object. For Firefox, you should currently check the MozWebSocket object. It should be noted that most HTML5-related capabilities can be checked in browsers by means of a specialized library such as Modernizr (modernizr.com). In particular, here's the JavaScript code you need to write if you linked the Modernizr library to your page:

```
if (Modernizr.websockets)
{
  ...
}
```

Figure 1 **Browser Support for WebSockets**

Browser	WebSockets Support
Internet Explorer	WebSockets will be supported in Internet Explorer 10. Metro applications written using JavaScript and HTML5 will support WebSockets as well.
Firefox	WebSockets are supported starting with version 6 of the browser released in mid-2011. Some very early support was offered in version 4 and then dropped in version 5.
Chrome	WebSockets are supported starting with version 14, which was released in September 2011.
Opera	Support for WebSockets has been removed in version 11.
Safari	Supports an earlier version of the WebSocket Protocol.

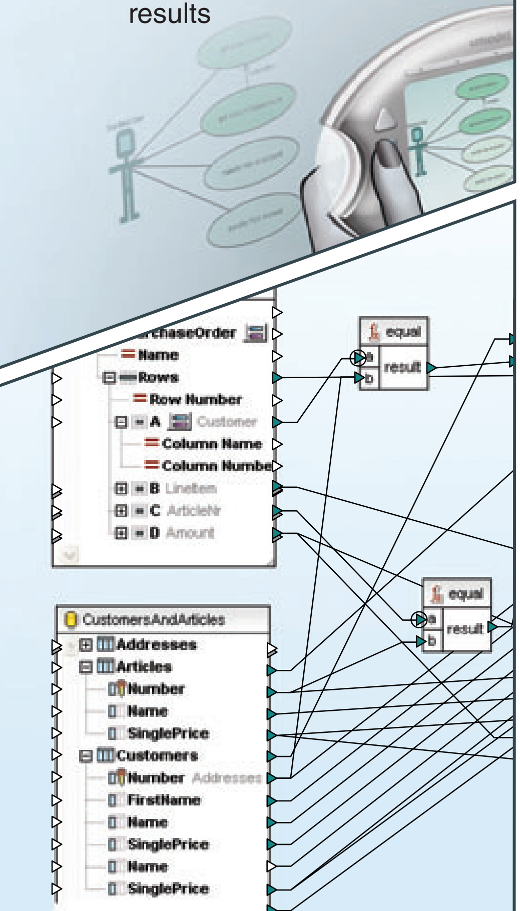
Visualize software works of art with the complete set of tools from Altova®

ALTOVA® missionkit®

The MissionKit® is an integrated suite of UML, XML, and data integration tools for today's software architect.

NEW in Version 2012:

- Support for Model Driven Architecture: transform UML models between C#, VB, Java, databases, & XSD
- Intelligent HTML5 editing
- Code generation from UML sequence diagrams
- Sorting of data mapping results



The Altova MissionKit includes multiple tools for software architects:

UModel® – UML tool for software modeling

- Support for all UML diagram types, MDA, SQL database diagrams, BPMN, SysML
- Reverse engineering and code generation in Java, C#, VB.NET

XMLSpy® – XML editor & development environment

- Support for all XML-based technologies
- Royalty-free Java, C#, C++ code generation

MapForce® – graphical data mapping tool

- Mapping between XML, databases, EDI, flat files, XBRL, Excel 2007+, Web services
- Royalty-free Java, C#, C++ code generation

Download a 30 day free trial!

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

Scan to learn more about these Altova MissionKit tools.

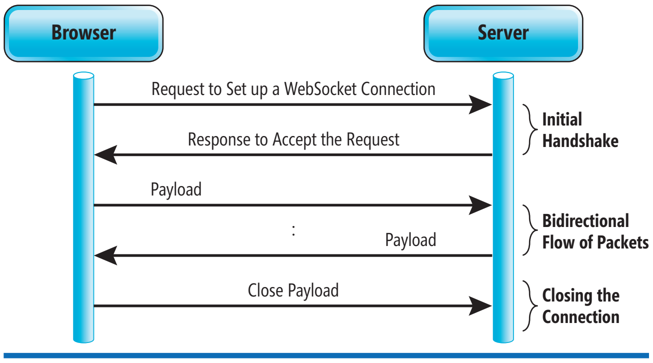

Browser
Server
Request to Set up a WebSocket Connection
Response to Accept the Request
Payload
:
Payload
Close Payload
Initial Handshake
Bidirectional Flow of Packets
Closing the Connection

Figure 2 **The WebSocket Protocol Schema**

Modernizr is probably an excellent choice today if you want to start using a WebSocket implementation because it provides you with polyfills—code that kicks in automatically if a given feature isn't supported on the current browser.

In the end, WebSockets are an extremely compelling feature with non-uniform support today across vendors. Microsoft, however, broadly supports WebSockets through the upcoming Internet Explorer 10 and also IIS, ASP.NET, Windows Communication Foundation (WCF) and Windows Runtime (WinRT). Note, though, that no official standard API exists yet, so the early support is a great sign of interest. The best you can do today is use WebSockets through some abstraction layer. Modernizr is a possible option if you want to stay close to the metal and write your own code that opens and closes WebSockets. SignalR is a better option if you're looking for a framework that transparently connects a browser and a Web endpoint in a persistent manner, with no frills and no need to know many underlying details.

A Look at the WebSocket Protocol

The WebSocket Protocol for bidirectional communication requires that both the client and server application are aware of the protocol details. This means you need a WebSocket-compliant Web page that calls into a WebSocket-compliant endpoint.

A WebSocket interaction begins with a handshake in which the two parties (browser and server) mutually confirm their intention to communicate over a persistent connection. Next, a bunch of message packets are sent over TCP in both directions. **Figure 2** outlines how the WebSocket Protocol works.

Note that in addition to what's in **Figure 2**, when the connection is closed, both endpoints exchange a close frame to cleanly close the connection. The initial handshake consists of a plain HTTP request that the client sends to the Web server. The request is an HTTP GET configured as an upgrade request:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
```

In HTTP, a client request with the Upgrade header indicates the intention of the client to request that the server switch to another protocol. With the WebSocket Protocol, the upgrade request to the server contains a unique key that the server will return mangled as the proof that it has accepted the upgrade request. This is a practical demonstration to show the server understands the WebSocket Protocol. Here's a sample response to a handshake request:

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

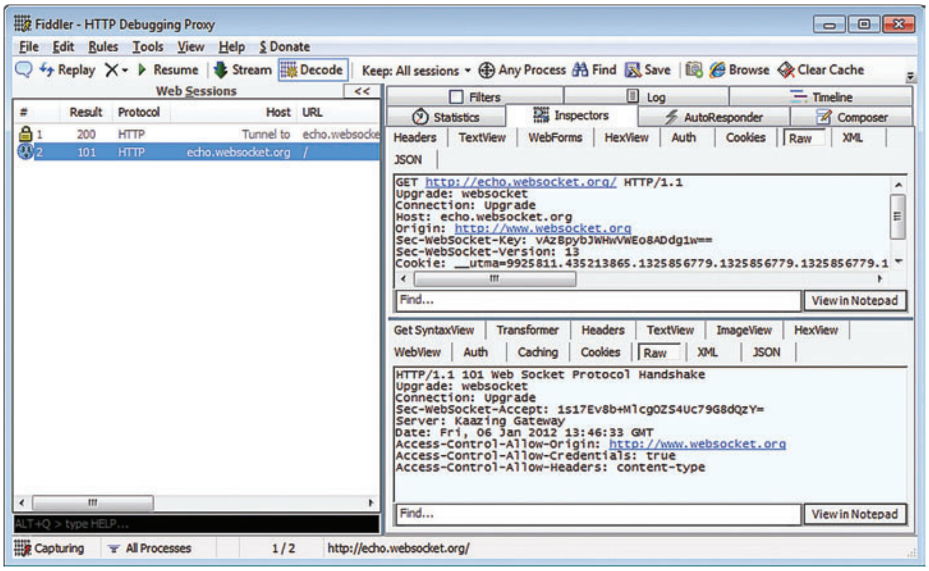
A successful status code is always 101, and any other status code will be interpreted as a refusal to upgrade to the WebSocket Protocol. The server concatenates the received key with a fixed GUID string and calculates a hash out of the resulting string. The hash value is then encoded to Base64 and returned to the client via the Sec-WebSocket-Accept header.

The client can also send other headers such as Sec-WebSocket-Protocol in order to indicate which subprotocols it's happy to employ. A subprotocol is an application-level protocol built on top of the basic WebSocket Protocol. If the server understands some of the suggested subprotocols, it will pick one up and send its name back to the client via the same header.

Following the handshake, client and server can freely send messages over the WebSocket Protocol. The payload begins with an opcode that indicates the operation being performed. One of these opcodes—specifically 0x8—indicates a request to close the session. Note that WebSocket messages go asynchronously so a send request won't necessarily receive an immediate response, as in HTTP. With the WebSocket Protocol, you're better off thinking in terms of general messages going from client to server or vice versa, and forgetting about the classic request/response pattern of HTTP.

The typical URL for a WebSocket endpoint takes the following form:

```
var myWebSocket =
  new WebSocket("ws://www.websocket.org");
```


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help $ Donate
Replay Resume Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache
Web Sessions
Filters Log Timeline
Statistics Inspectors AutoResponder Composer
Headers TextView WebForms HexView Auth Cookies Raw XML JSON
Get SyntaxView Transformer Headers TextView ImageView HexView
WebView Auth Caching Cookies Raw XML JSON
Find... View in Notepad
ALT+Q > type HELP...
Capturing All Processes 1 / 2 http://echo.websocket.org/

#	Result	Protocol	Host	URL
1	200	HTTP	Tunnel to	echo.websocke
2	101	HTTP	echo.websocket.org	/

```
GET http://echo.websocket.org/ HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Host: echo.websocket.org
Origin: http://www.websocket.org
Sec-WebSocket-Key: vAzBpybJWHwVWEo8ADdg1w==
Sec-WebSocket-Version: 13
Cookie: __utma=9925811.435213865.1325856779.1325856779.1325856779.1
```

```
HTTP/1.1 101 Web Socket Protocol Handshake
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: 1s17Ev8b+Mlcg0ZS4Uc79G8dQzY=
Server: Kaazing Gateway
Date: Fri, 06 Jan 2012 13:46:33 GMT
Access-Control-Allow-Origin: http://www.websocket.org
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: content-type
```

Figure 3 **Real Handshaking Between Browser and Server**

All-in-One with dotConnect

ADO.NET Providers for Oracle, MySQL, PostgreSQL, SQLite and **Salesforce Now!**

Entity Framework

Fast, configurable EF provider with support for newest EF features

SQL Server BIS

BIS Data source for data export/import

Entity Developer

Powerful ORM model designer for Entity Framework and LinqConnect

Database Specific Extensions

Script, Loader, Alerter, Dump, Change Notification

dbMonitor

Monitor all the database interaction with free dbMonitor tool

NHibernate

Support for the well-known open-source ORM

ASP.NET DataSource component

SqlDataSource analog for different databases

Enterprise Library

Provider for Enterprise Library Data Access Block

ADO.NET Provider

Fast and advanced ADO.NET provider with direct data access, rich design-time

Workflow Foundation

Workflow Instance Store and Workflow Tracking support

LinqConnect

Fast and lightweight ORM solution, compatible with LINQ to SQL

ASP.NET Providers

ASP.NET Provider for storing state in different databases

devart

www.devart.com

You use the wss protocol prefix if you want to go on a secure socket connection (secure connections will generally be more successful when intermediaries are present). Finally, the WebSocket Protocol acknowledges and addresses the problem of cross-origin communication. A WebSocket client generally—but not always—permits sending requests to endpoints located on any domain. But it's the WebSocket server that will decide whether to accept or refuse the handshake request.

A Look at the WebSocket API

As mentioned, the W3C is currently standardizing an API for the WebSocket Protocol, and browsers are catching up with the various drafts as they become available. You should be aware any code that works today might not work across all browsers and, more importantly, is not even guaranteed to work on the same browser when a new release hits the market. In any case, once you have some working WebSocket code you're pretty much done, as any changes that might be required in the future will likely be just minor changes.

If you want to experiment with the WebSocket Protocol you can visit websocket.org with a browser that supports the protocol. For example, you can use a preview of Internet Explorer 10 or a recent version of Google Chrome. **Figure 3** shows the handshake as it's being tracked by Fiddler.

Not surprisingly, the current version of Fiddler (version 2.3.x) will only capture HTTP traffic. However, a new version of Fiddler that deals with WebSocket traffic is currently in beta.

The WebSocket API is fairly simple. On the browser side, you need to create an instance of the WebSocket browser class. This class exposes a bunch of interesting events for which you want to have proper handlers:

```
var wsUri = " ws://echo.websocket.org/";
websocket = new WebSocket(wsUri);
websocket.onopen = function(evt) { onOpen(evt) };
websocket.onmessage = function(evt) { onMessage(evt) };
websocket.onclose = function(evt) { onClose(evt) };
websocket.onerror = function(evt) { onError(evt) };
```

The onopen event is fired when the connection is established. The onmessage event fires whenever the client receives a message from the server. The onclose is triggered when the connection has been closed. Finally, onerror is fired whenever an error occurs.

To send a message to the server, all you need to do is place a call to the method send, as shown here:

```
var message = "Cutting Edge test: " +
  new Date().toString();
websocket.send(message);
```

Figure 4 shows a sample page that's an adaptation of the echo example found on the websocket.org Web site. In this example the server just echoes the received message back to the client.

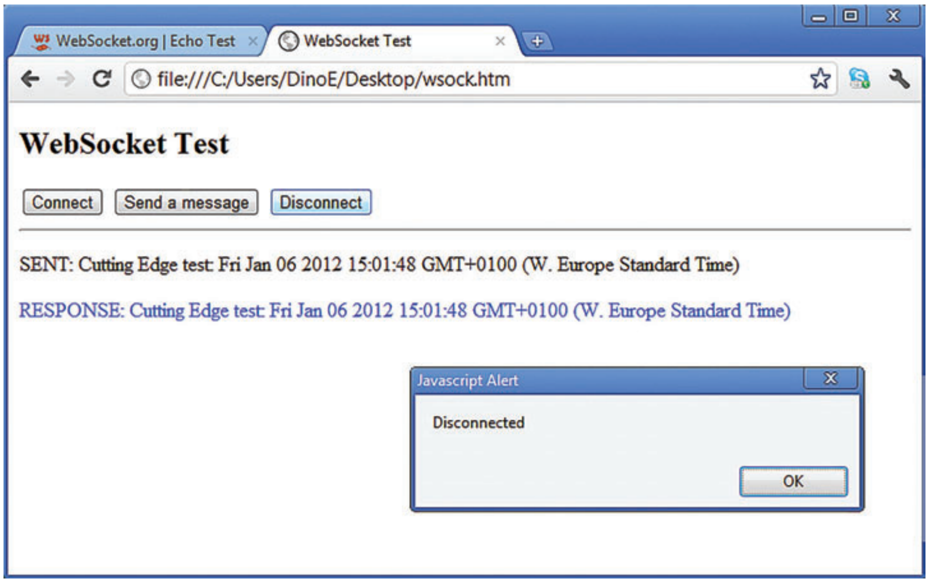
If you're interested in WebSocket programming for Internet Explorer 10, see bit.ly/GNYWFh.

The Server Side of WebSockets

In this article I mostly focused on the client side of the WebSocket Protocol. It should be clear that in order to use a WebSocket client, you need a proper WebSocket-compliant server that understands the requests and can reply appropriately. Frameworks for building a WebSocket server have started to appear. For example, you can try out Socket.IO for Java and Node.js (socket.io). If you're looking for some Microsoft .NET Framework stuff, have a look at "Web Socket Server" from The Code Project at bit.ly/lc0rjt. In addition, Microsoft server support for WebSockets is available in IIS, ASP.NET and WCF. You can refer to the Channel 9 video, "Building Real-Time Web Apps with WebSockets Using IIS, ASP.NET and WCF," for more details (bit.ly/rnYaw5).

Sliced Bread, Hot Water and WebSockets

As many have said, WebSockets are the most useful invention since sliced bread and hot water. After you've made sense of WebSockets you just wonder how the software world could've thrived without them. WebSockets are helpful in a number of applications, though not for just any applications. Any application where instant messaging is key is a potential scenario where you can seriously consider building a WebSocket server and a bunch of clients—Web, mobile or even desktop. Gaming and live-feed applications are other industry fields that will benefit immensely from the WebSocket Protocol. Yes, WebSockets are definitely the best after hot water! ■


WebSocket.org | Echo Test
WebSocket Test
file:///C:/Users/DinoE/Desktop/wsock.htm
Connect
Send a message
Disconnect
Javascript Alert
OK

WebSocket Test

SENT: Cutting Edge test: Fri Jan 06 2012 15:01:48 GMT+0100 (W. Europe Standard Time)

RESPONSE: Cutting Edge test: Fri Jan 06 2012 15:01:48 GMT+0100 (W. Europe Standard Time)

Disconnected

Figure 4 **The WebSocket Protocol in Action**

DINO ESPOSITO *is the author of "Programming ASP.NET 4" (Microsoft Press, 2011) and "Programming ASP.NET MVC 3" (Microsoft Press, 2010), and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. Follow him on Twitter at Twitter.com/despos.*

THANKS to the following technical experts for reviewing this article:
Levi Broderick and Brian Raymor

Powerful Tools for Developers

v4.5!

High-Performance PDF Printer Driver

- Create accurate PDF documents in a fraction of the time needed with other tools
- WHQL tested for all Windows 32 and 64-bit platforms
- Produce fully compliant PDF/A documents
- Standard PDF features included with a number of unique features
- Interface with any .NET or ActiveX programming language

v4.5!

PDF Editor for .NET, now Webform Enabled

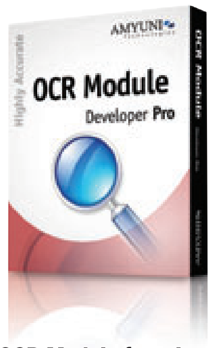
- Edit, process and print PDF 1.7 documents programmatically
- Fast and lightweight 32 and 64-bit managed code assemblies for Windows, WPF and Web applications
- Support for dynamic objects such as edit-fields and sticky-notes
- Save image files directly to PDF, with optional OCR
- Multiple image compression formats such as PNG, JBIG2 and TIFF

New!

PDF Integration into Silverlight Applications

- Server-side PDF component based on the robust Amyuni PDF Creator ActiveX or .NET components
- Client-side C# Silverlight 3 control provided with source-code
- Optimization of PDF documents prior to converting them into XAML
- Conversion of PDF edit-boxes into Silverlight TextBox objects
- Support for other document formats such as TIFF and XPS

OCR Module available with royalty-free licensing!



The new OCR Module from Amyuni enables developers to:

- Convert non-searchable PDF files into searchable PDFs
- Create searchable PDF documents out of various image formats such as multi-page TIFF, JPEG or PNG while applying text recognition
- Compress image based PDF documents using high compression JBIG2 or more standard CCITT, JPEG and PNG compression formats

The Amyuni OCR module is based on the Tesseract Library with the Amyuni PDF technology being used to process and create the PDF documents.

Learn more at www.amyuni.com

Windows 7

CERTIFIED FOR Windows Server 2008

More Development Tools Available at:

www.amyuni.com

USA and Canada
Toll Free: 1 866 926 9864
Support: (514) 868 9227

Info: sales@amyuni.com

Europe
Sales: (+33) 1 30 61 07 97
Support: (+33) 1 30 61 07 98

Customizations: management@amyuni.com

AMYUNI Technologies

windows forms · wpf · silverlight · windows phone 7 · webforms · mvc · html5

<we know controls>

<xaml>

<windows>

<web>

FREE TRIAL @: *http://c1.ms/ultimate*

Build everything... everywhere with the ultimate collection of tools from ComponentOne. Create any type of application, reach every platform, and impress your end users with stunning UIs and boosted performance.

iphone · compact framework · entity framework · activex · lightswitch · sharepoint

<xaml>

Develop for the desktop and web at once, and deliver line-of-business apps that take advantage of the latest trends.

<windows>

Embrace the powerful desktop with next-generation controls complete with built-in features, smart designers, and hundreds of samples.

<web>

The best is standard in our tools for ASP.NET and HTML5 apps; this includes application-wide theming, full cross-browser support, unmatched performance, and built-in Web Standards.

ComponentOne®

5 Reasons to Start Working with Windows Azure

Everywhere you turn nowadays, you hear about the cloud—that it's a major step in the evolution of the Web and will change the way you develop, deploy and manage applications. But not everyone has figured out how the cloud really applies to them. This is especially true for those with medium-to-large infrastructures and relatively flat usage consumption—where the capitalized cost is beneficial compared to the operational cost of the cloud. However, if your infrastructure is on the small side or you have a dynamic consumption model, the cloud—Windows Azure—is a no-brainer. Moreover, for shops heavy in process, where standing up a development environment is like sending a bill to Capitol Hill, Windows Azure can provide a great platform for rapid prototyping.

It's with those thoughts in mind that I want to point out some things about Windows Azure that I hope might spur you into putting the magazine down and putting some Windows Azure up.

Great Tools Integration

For Windows Azure development, the tooling and the integration with Visual Studio has been pretty good—and is quickly evolving into great. You can find the latest set of tools in the Windows Azure Developer Center at bit.ly/xh1CAE.

For Windows Azure development, the tooling and the integration with Visual Studio has been pretty good—and is quickly evolving into great.

As **Figure 1** shows, you can select the type of roles and language you want for a new project. No matter what you choose, you can immediately take advantage of the tools integration. In my experience, the three features you'll find most helpful are the development emulators, runtime debugging and integrated deployment.

The Windows Azure development environment consists of two emulators that allow you to easily run and debug your applications on your development machine before deployment (see **Figure 2**). The Windows Azure Compute Emulator is what lets you run your service locally for testing and debugging. With the Storage Emulator, you can test the storage locally.

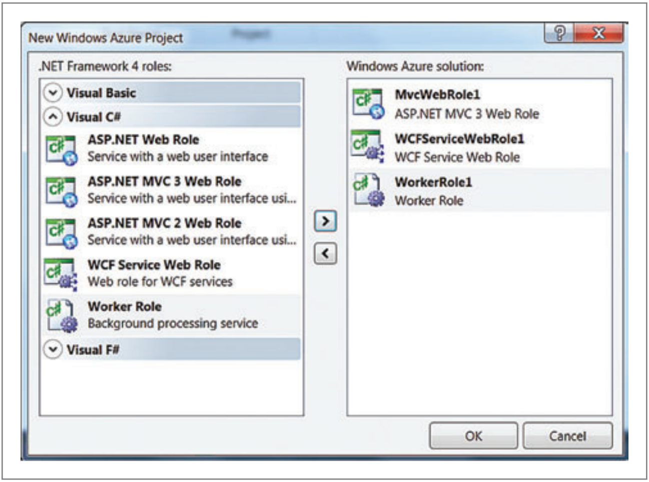


Figure 1 **Creating a New Project**

When the time is right, deployment to the Staging or Production environment is just a right-click away. The tools take care of packaging, moving and deploying the roles within your solution, and progress is reported back via Visual Studio, as **Figure 3** shows.

Early on, a big problem with Windows Azure was that you could've developed some code that worked perfectly locally, but failed or had terrible performance once deployed. Luckily, the introduction of IntelliTrace and profiling helped alleviate these issues. You can enable these features when you publish your solution, as shown in **Figure 4**.

For debugging hard-to-reproduce errors, especially those that seem to show up only in the production environment, there's nothing quite as good as IntelliTrace. IntelliTrace essentially records the execution of your application, which you can then play back. For example, once you deploy the roles with IntelliTrace enabled, you can view the IntelliTrace logs and step through exactly what happened at what time (see **Figure 5**).

Once you've stepped into the thread, you can walk though any existing code to see what was changing during execution. When your site is bug-free (or as bug-free as it's going to get) and you're ready to try to identify performance issues, you can turn off IntelliTrace and turn on profiling. As you saw in **Figure 4**, you can select the type of profiling to do. For example, if you're wondering what the call time is on individual methods, you might select Instrumentation.

Telerik RadControls for

ASP.NET AJAX

The industry's leading AJAX components

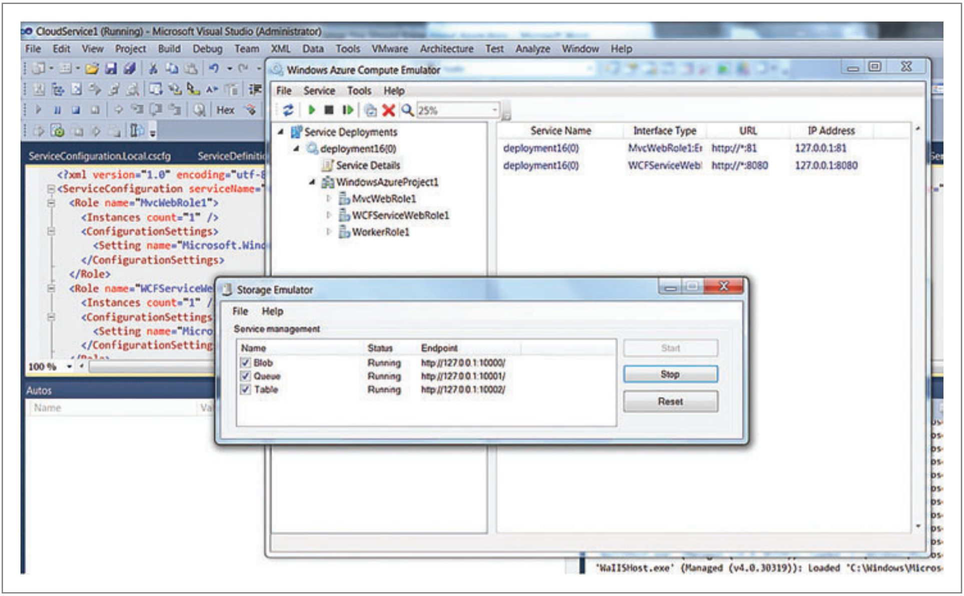
All Telerik Developer tools support Visual Studio 2011

We created RadControls for ASP.NET AJAX with your productivity in mind.
We offer a suite of feature-rich controls for every need that will truly save you time, while helping you deliver awesome-looking apps with the performance you desire.
You be the judge.

TRY IT FOR YOURSELF at www.telerik.com/ajax

Best of tech·ed 2011 Finalist | Visual Studio Magazine Readers Choice 2011 Winner | SD Times 2011 100 | Members Choice Winner The Code Project 2011 | Microsoft Partner Network 2011 PARTNER OF THE YEAR Mobility Finalist

telerik
deliver more than expected


CloudService1 (Running) - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team XML Data Tools VMware Architecture Test Analyze Window Help
ServiceConfiguration.Local.cscfg
Windows Azure Compute Emulator
File Service Tools Help
25%
Service Deployments
deployment16(0)
Service Details
WindowsAzureProject1
MvcWebRole1
WCFServiceWebRole1
WorkerRole1
Storage Emulator
File Help
Service management
Start
Stop
Reset
Autos
Name

```
<?xml version="1.0" encoding="utf-8
<ServiceConfiguration serviceName="
  <Role name="MvcWebRole1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.Wind
    </ConfigurationSettings>
  </Role>
  <Role name="WCFServiceWe
    <Instances count="1" /
    <ConfigurationSettings
      <Setting name="Micro
    </ConfigurationSetting
```

Service Name	Interface Type	URL	IP Address
deployment16(0)	MvcWebRole1:E	http://*:81	127.0.0.1:81
deployment16(0)	WCFServiceWeb	http://*:8080	127.0.0.1:8080

Name	Status	Endpoint
✓ Blob	Running	http://127.0.0.1:10000/
✓ Queue	Running	http://127.0.0.1:10001/
✓ Table	Running	http://127.0.0.1:10002/

```
'WaIISHost.exe' (Managed (v4.0.30319)): Loaded 'C:\Windows\Micros
```

Figure 2 **The Windows Azure Compute Emulator and Storage Emulator Running**

This method collects detailed timing data that's useful for focused analysis and for analyzing input/output performance issues. It's also useful for gathering detailed timing information about a section of your code and for understanding the impact of input and output operations on application performance. You can then walk through the site to execute the code base until you're satisfied, at which point you'll choose to "View Profiling Report" to see an instance in Server Explorer. Visual Studio will fetch the information and put together a report like the one depicted in **Figure 6**.

The report shows CPU usage over time, as well as the "Hot Path," which alone might help you focus your efforts. If you'd like to dig a little further, however, a direct link in the Hot Path section lets you see the individual timing for each function. The main page also displays a nice graph indicating the functions that do the most individual work. Clearly, having IntelliTrace and profiling available directly from Visual Studio is a huge benefit, not only for productivity but also for product quality.

Note that if you're working in JavaScript, PHP or Java, you're not left out in the cold, having to write your own access to the Windows Azure platform. Microsoft provides SDKs and resources for all of these at bit.ly/uGqPNh.

Performance and Scale

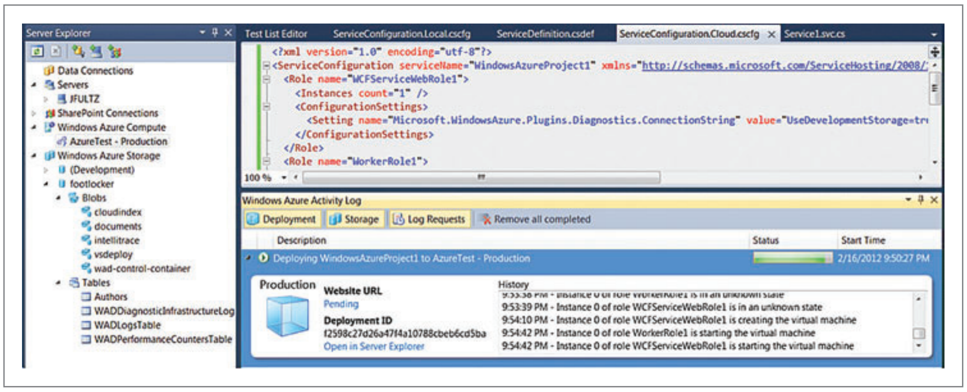
If you've been paying even marginal attention over the past few years, you know that one of the key promises of the cloud is the ability to scale on demand. For compute virtual machines (VMs), you can often just pay more for a larger role and get more resources. For Microsoft SQL Azure, though, the optimizations are a little more … well … manual.

It's great to know that deploying to the cloud gives you the ability to scale the farm, but a more immediate question is often, "What size role do I need?" The answer is that it depends on traffic and what you're doing. You can take an educated guess based on your past experience and on the specifications of the role sizes, as shown in **Figure 7**.

One of these configurations is likely to meet your needs, especially in combination with the rest of the role instances in the farm. Take note that all attributes increase, including the available network bandwidth, which is often a secondary consideration for folks. Note also that you don't really have to guess. Instead you can turn on profiling as discussed previously and collect actual metrics across the instances to assess performance. Based on profiling results, you can adjust the VM size and collect profiling information again until the sweet spot is reached. For edge conditions, you make a best-fit choice or find an alternative solution. For example, if your site serves a lot of content and isn't very dynamic, you might choose one of the higher role specs or move to the Windows Azure Content Delivery Network.

Clearly, having IntelliTrace and profiling available directly from Visual Studio is a huge benefit.

Now for some mixed news: SQL Azure does not always give the performance you might get with your own private instance. You will, however, get consistent performance. There are a few things you can do to get the best possible performance and runtime behavior:



```
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="WindowsAzureProject1" xmlns="http://schemas.microsoft.com/ServiceHosting/2008/
  <Role name="WCFServiceWebRole1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="UseDevelopmentStorage=tr
    </ConfigurationSettings>
  </Role>
  <Role name="WorkerRole1">
```

Description	Status	Start Time
Deploying WindowsAzureProject1 to AzureTest - Production		2/16/2012 9:50:27 PM

Production
Website URL: Pending
Deployment ID: f2598c27d26a47f4a10788cbeb6cd5ba

```
History
9:53:39 PM - Instance 0 of role WCFServiceWebRole1 is in an unknown state
9:54:10 PM - Instance 0 of role WCFServiceWebRole1 is creating the virtual machine
9:54:42 PM - Instance 0 of role WorkerRole1 is starting the virtual machine
9:54:42 PM - Instance 0 of role WCFServiceWebRole1 is starting the virtual machine
```

Figure 3 **Deployment Progress for Windows Azure as Reported Back Through Visual Studio**

The best ideas evolve.

Great ideas don’t just happen.
They evolve. Your own development
teams think and work fast.

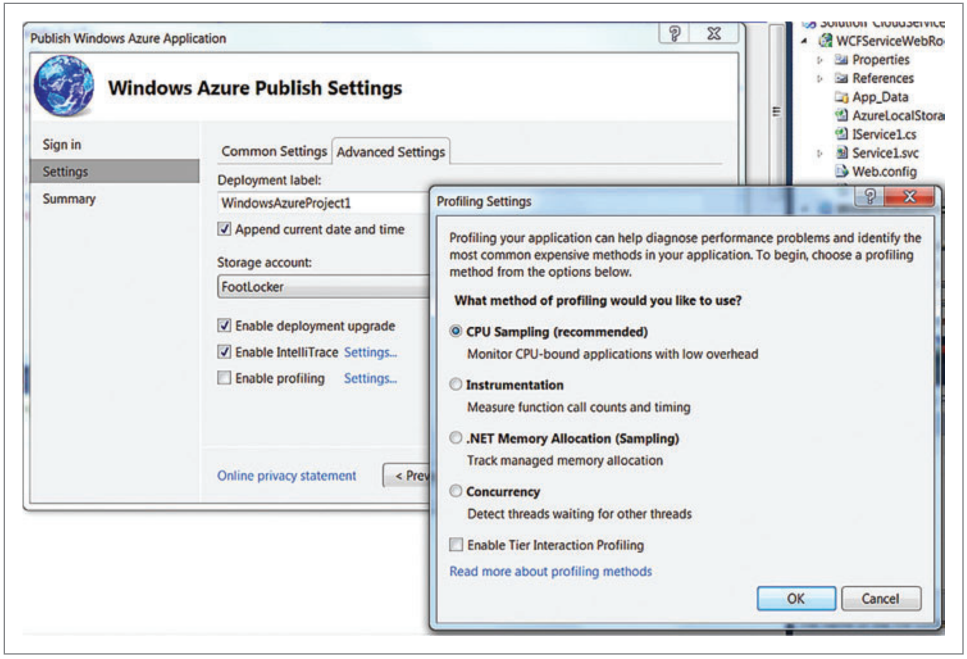
Don’t miss a breakthrough.
Version *everything* with Perforce.

Software and firmware. Digital assets
and games. Websites and documents.
More than 5,500 organizations and
400,000 users trust Perforce for
enterprise version management.

Try it now. Download the free
20-user, non-expiring Perforce Server
from perforce.com/free20

Or request an evaluation license
for any number of users.

PERFORCE
Version everything.


Publish Windows Azure Application

Windows Azure Publish Settings

Sign in
Settings
Summary

Common Settings | Advanced Settings

Deployment label:
WindowsAzureProject1
☑ Append current date and time

Storage account:
FootLocker

☑ Enable deployment upgrade
☑ Enable IntelliTrace Settings...
☐ Enable profiling Settings...

Online privacy statement

Profiling Settings

Profiling your application can help diagnose performance problems and identify the most common expensive methods in your application. To begin, choose a profiling method from the options below.

What method of profiling would you like to use?

● **CPU Sampling (recommended)**
Monitor CPU-bound applications with low overhead

○ **Instrumentation**
Measure function call counts and timing

○ **.NET Memory Allocation (Sampling)**
Track managed memory allocation

○ **Concurrency**
Detect threads waiting for other threads

☐ Enable Tier Interaction Profiling

Read more about profiling methods

OK Cancel

Solution 'CloudService
WCFServiceWebRo
Properties
References
App_Data
AzureLocalStora
IService1.cs
Service1.svc
Web.config

Figure 4 **IntelliTrace and Profiling Settings in Windows Azure**

1. Make sure SQL Azure is in the same datacenter as your compute.
2. Optimize your queries and data structures to suit your queries.
3. Don't cut corners on retry logic and make sure the retry logic is in your code and tested.
4. Repeat step 2.

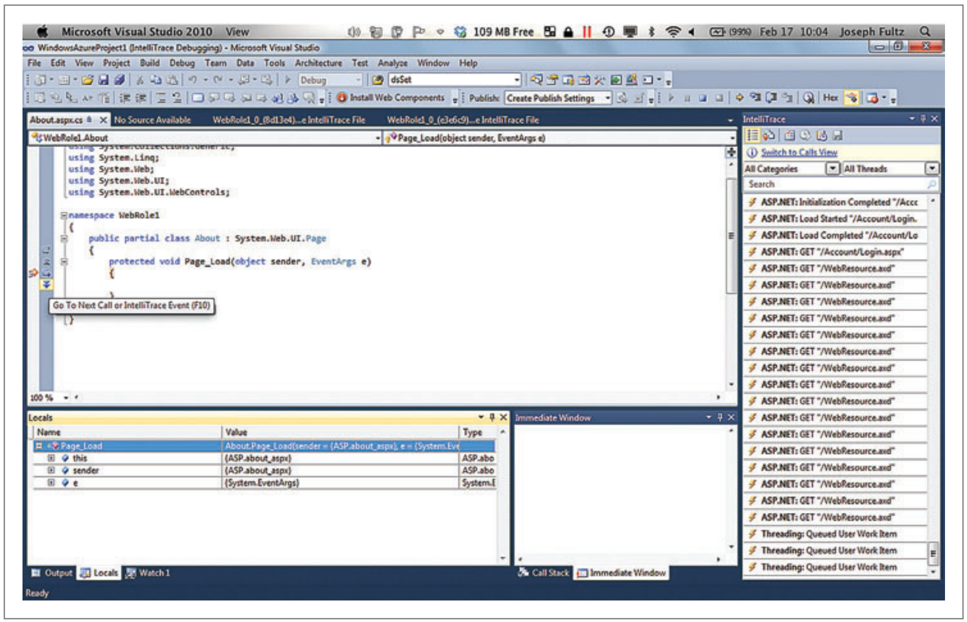
Over the years, one of the biggest mistakes I've seen people make when optimizing a site is to just increase the size of the hardware without doing anything else. Sometimes this helped a little, but as soon as load really spiked, the problem would come back with symptoms worse than ever, because the additional horsepower had the effect of making more things conflict more quickly and not actually resolving or mitigating the real issue. So, when I suggest repeating step 2, I'm not kidding. You can't just throw more hardware at the problem and hope it isn't a deadlocking issue. The SQL Azure Profiler tool can help you with this effort. I suggest you start with the optimization on your local instance prior to deploying to the cloud, and then use SQL Azure Profiler to help identify and make any adjustments needed once in the cloud.

As a final note, one strategy for increasing scale or size of a SQL Azure database is federation, commonly referred to as "data sharding," which is a technique of horizontally partitioning data across multiple physical servers to provide application scale-out. This reduces individual query times, but adds the complexity of scattering the queries to target instances and gathering the results together once they're complete. For example, you'll get the benefit of running Create, Read, Update, Delete (CRUD) operations and smaller datasets, and in parallel. The tax you'll pay is having to broker the access across the shards. That being said, some of the largest sites employ sharding, prefetching and caching to manage queries, and you use those sites every day without much complaint about performance.

Manageable Infrastructure

Early on it was not always easy to know what was going on in a Windows Azure deployment, but those days are long gone. Not only does Microsoft provide an ever-evolving management portal, but a management pack for System Center Operations Manager (SCOM) brings the management of your entire infrastructure into one place.

Windows Azure writes all of its diagnostics data out to a storage container. You can consume the logs directly and generate reports or take custom actions. However, you can also use SCOM to monitor Windows Azure applications. Those responsible for managing the infrastructure of an enterprise are inclined to be conservative and want full-featured tools for monitoring. Using a familiar solution like SCOM will help address the reservations that the infrastructure management team might have


Microsoft Visual Studio 2010 View
109 MB Free
(99%) Feb 17 10:04 Joseph Fultz
WindowsAzureProject1 (IntelliTrace Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
dsSet
Install Web Components
Publish: Create Publish Settings
Hex
About.aspx.cs
No Source Available
WebRole1_0_(8d13e4)...e IntelliTrace File
WebRole1_0_(c3e6c9)...e IntelliTrace File
WebRole1.About
Page_Load(object sender, EventArgs e)
Go To Next Call or IntelliTrace Event (F10)
100 %
Output
Locals
Watch 1
Call Stack
Immediate Window
Ready

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebRole1
{
    public partial class About : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Locals

Name	Value	Type
Page_Load	About.Page_Load(sender = {ASP.about_aspx}, e = {System.Eve	
this	{ASP.about_aspx}	ASP.abo
sender	{ASP.about_aspx}	ASP.abo
e	{System.EventArgs}	System.E

IntelliTrace

Switch to Calls View

All Categories | All Threads

Search

```
ASP.NET: Initialization Completed "/Accc
ASP.NET: Load Started "/Account/Login.
ASP.NET: Load Completed "/Account/Lo
ASP.NET: GET "/Account/Login.aspx"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
ASP.NET: GET "/WebResource.axd"
Threading: Queued User Work Item
Threading: Queued User Work Item
Threading: Queued User Work Item
```

Figure 5 **Debugging Windows Azure with IntelliTrace in Visual Studio**

Files *driving* you crazy?

Get on the right track with Aspose

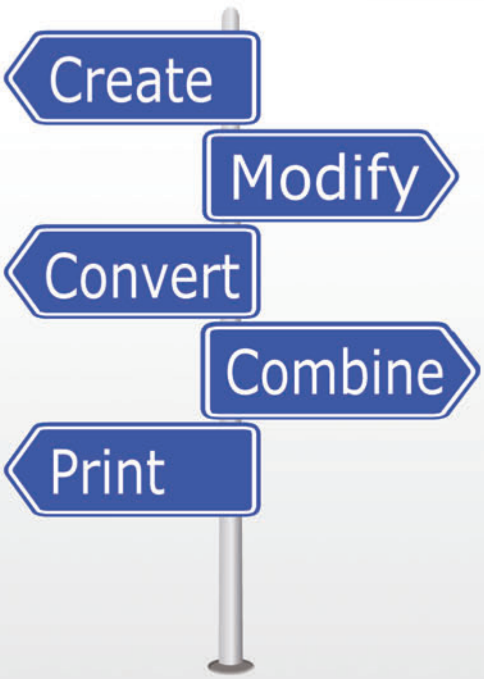
Aspose.Words
DOC, DOCX, RTF, HTML, PDF,
XPS & other document formats.

Aspose.Cells
XLS, XLSX, XLSM, XLTX, CSV,
SpreadsheetML & image formats.

Aspose.Pdf
PDF, XML, XLS-FO, HTML, BMP,
JPG, PNG & other image formats.

Aspose.Email
MSG, EML, PST, EMLX &
other formats.

and many more!



Aspose.BarCode Aspose.Tasks
Aspose.Slides Aspose.Diagram
Aspose.OCR Aspose.Imaging

Scan our QR Code
for an exclusive
20% coupon code.

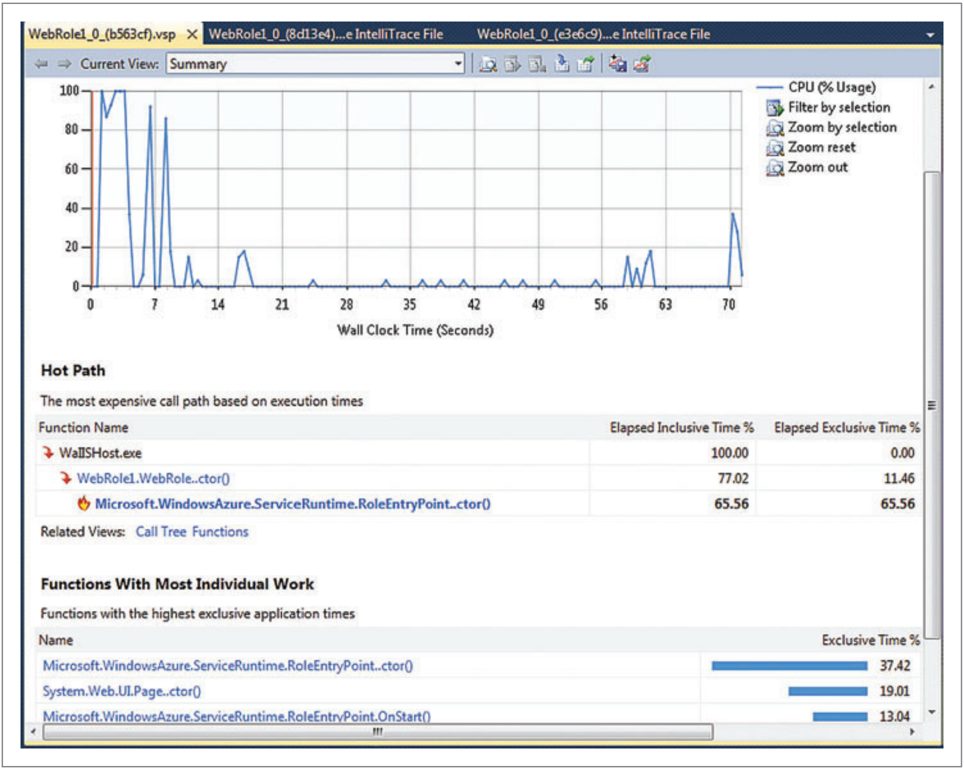
Follow us on
Facebook & Twitter

ASPOSE
Your File Format Experts

Get your FREE evaluation copy at **http://www.aspose.com**

US Sales: 1.888.277.6734
sales@aspose.com

EU Sales: +44 (0)800 098 8425
sales.europe@aspose.com



Hot Path

The most expensive call path based on execution times

Function Name	Elapsed Inclusive Time %	Elapsed Exclusive Time %
WaIISHost.exe	100.00	0.00
WebRole1.WebRole..ctor()	77.02	11.46
Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint..ctor()	**65.56**	**65.56**

Related Views: Call Tree Functions

Functions With Most Individual Work

Functions with the highest exclusive application times

Name	Exclusive Time %
Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint..ctor()	37.42
System.Web.UI.Page..ctor()	19.01
Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint.OnStart()	13.04

Figure 6 **A Profiling Report**

about deploying a cloud solution. SCOM lets you monitor the health of all Windows Azure deployments and enables you to drill down into Hosted Services, Roles and Role Instances. Built into the pack are alerts for services and performance, but a key benefit is that you can create your own rules and alerts relating to your deployments and the data being collected. An additional nicety is that rules for grooming the logs are built-in. As usual, if the logs aren't pruned along the way, they can grow to be unmanageable. To help with that, the management pack comes with predefined rules:

- .NET Trace Grooming
- Performance Counter Grooming
- Event Log Grooming

These can be enabled to make sure your space usage doesn't get out of hand, but you'll need to balance that against the number of transactions to perform the tasks. You can download the System Center Monitoring Pack for Windows Azure Applications at bit.ly/o5MW4a.

You're Writing the Code Already

Very often when a new technology comes along, you have to go through a fair amount of education and experience to become proficient: think about moving to Windows Presentation Foundation/Silverlight from Windows Forms, choosing whether to use ASP.NET or SharePoint, or even something more foundational such as deciding between procedural and object-oriented development. This is the very thing about the cloud, especially with the tooling available: If you're already writing sites and services, you can continue to make the most of your .NET skills and investments and move straight to the cloud.

This doesn't mean there aren't some best practices to learn, but not much more than you'd already be doing to be thorough in your design and development. And, when you're ready, the platform provides many additional features that can be learned and leveraged to make your solution secure and robust, and have the best performance without having to write the features or frameworks yourself.

It's the Future

Start now. That's my advice. Go to azure.com, get the tools and get started. Use Windows Azure in your projects to prototype. Use it in your projects to provide otherwise hard-to-requisition resources. Use it for whatever you want, but use it. The cloud is the future we will all live in, and it will be as ubiquitous as running water and electricity. Cloud technologies are evolving to expand computing beyond the conventional to a model that will deliver computing power where it's needed, when it's needed and the way it's needed. That's something you want to be a part of. ■

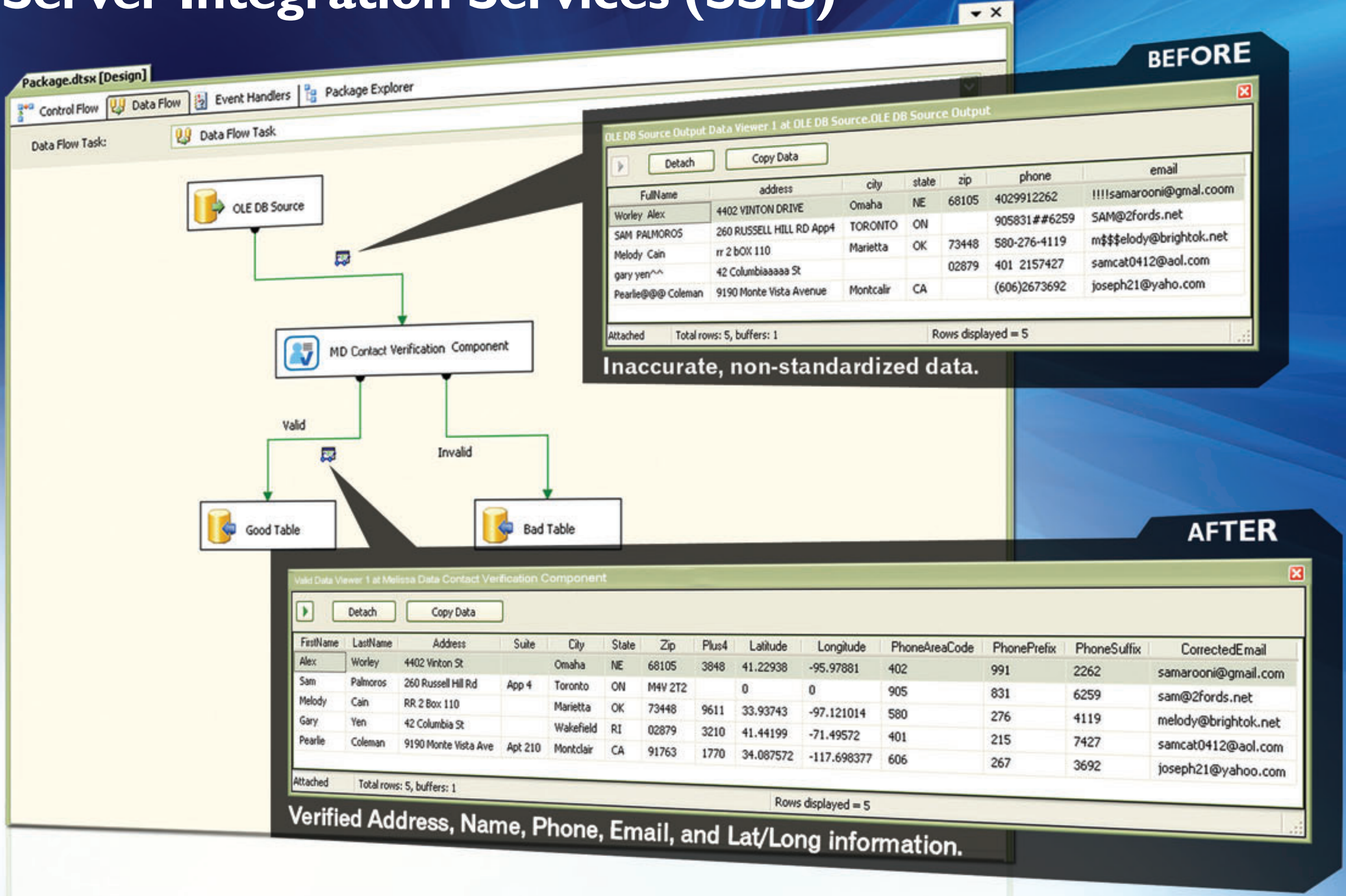
Figure 7 **Virtual Machine Size Specifications**

Virtual Machine Size	CPU Cores	Memory	Disk Space for Local Storage Resources in Web and Worker Roles	Disk Space for Local Storage Resources in a VM Role	Allocated Bandwidth (Mbps)
Extra Small	Shared	768MB	19,480MB (6,144MB is reserved for system files)	20GB	5
Small	1	1.75GB	229,400MB (6,144MB is reserved for system files)	165GB	100
Medium	2	3.5GB	500,760MB (6,144MB is reserved for system files)	340GB	200
Large	4	7GB	1,023,000MB (6,144MB is reserved for system files)	850GB	400
Extra Large	8	14GB	2,087,960MB (6,144MB is reserved for system files)	1890GB	800

JOSEPH FULTZ *is a software architect at Hewlett-Packard Co., working as part of the HP.com Global IT group. Previously he was a software architect for Microsoft, working with its top-tier enterprise and ISV customers defining architecture and designing solutions.*

THANKS to the following technical expert for reviewing this article: *Bruno Terkaly*

Data Cleansing Tools for SQL Server Integration Services (SSIS)



OLE DB Source Output Data Viewer 1 at OLE DB Source.OLE DB Source Output

FullName	address	city	state	zip	phone	email
Worley Alex	4402 VINTON DRIVE	Omaha	NE	68105	4029912262	!!!!samarooni@gmal.coom
SAM PALMOROS	260 RUSSELL HILL RD App4	TORONTO	ON		905831##6259	SAM@2fords.net
Melody Cain	rr 2 bOX 110	Marietta	OK	73448	580-276-4119	m$$$elody@brightok.net
gary yen^^	42 Columbiaaaaa St			02879	401 2157427	samcat0412@aol.com
Pearlie@@@ Coleman	9190 Monte Vista Avenue	Montcalir	CA		(606)2673692	joseph21@yaho.com

Attached Total rows: 5, buffers: 1 Rows displayed = 5

Inaccurate, non-standardized data.

Valid Data Viewer 1 at Melissa Data Contact Verification Component

FirstName	LastName	Address	Suite	City	State	Zip	Plus4	Latitude	Longitude	PhoneAreaCode	PhonePrefix	PhoneSuffix	CorrectedEmail
Alex	Worley	4402 Vinton St		Omaha	NE	68105	3848	41.22938	-95.97881	402	991	2262	samarooni@gmail.com
Sam	Palmoros	260 Russell Hill Rd	App 4	Toronto	ON	M4V 2T2		0	0	905	831	6259	sam@2fords.net
Melody	Cain	RR 2 Box 110		Marietta	OK	73448	9611	33.93743	-97.121014	580	276	4119	melody@brightok.net
Gary	Yen	42 Columbia St		Wakefield	RI	02879	3210	41.44199	-71.49572	401	215	7427	samcat0412@aol.com
Pearlie	Coleman	9190 Monte Vista Ave	Apt 210	Montclair	CA	91763	1770	34.087572	-117.698377	606	267	3692	joseph21@yahoo.com

Attached Total rows: 5, buffers: 1 Rows displayed = 5

Verified Address, Name, Phone, Email, and Lat/Long information.

Contact Verification Component

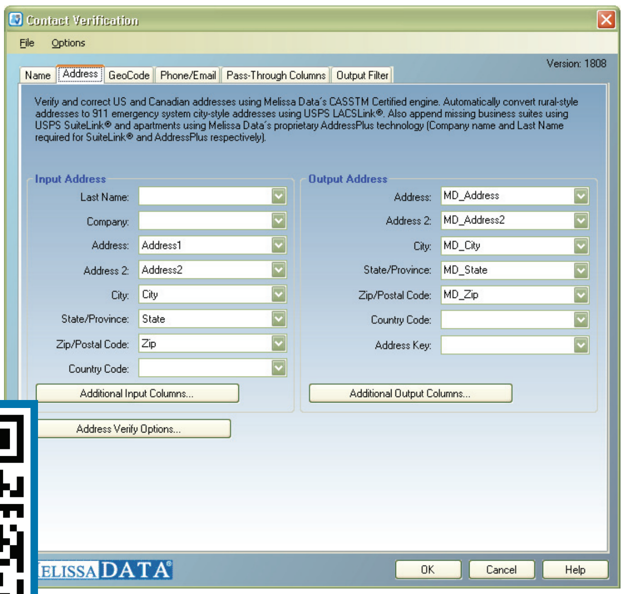
Parse, validate, correct and geocode addresses, phone numbers, email addresses, and full names.

SmartMover Component

Identify and update the addresses of people or businesses that have moved in the last 48 months.

MatchUp Component

Use advanced fuzzy matching algorithms and a custom lexicon to accurately dedupe contact data.



Verify and correct US and Canadian addresses using Melissa Data´s CASSTM Certified engine. Automatically convert rural-style addresses to 911 emergency system city-style addresses using USPS LACSLink®. Also append missing business suites using USPS SuiteLink® and apartments using Melissa Data´s proprietary AddressPlus technology (Company name and Last Name required for SuiteLink® and AddressPlus respectively).

Input Address

Last Name:	
Company:	
Address:	Address1
Address 2:	Address2
City:	City
State/Province:	State
Zip/Postal Code:	Zip
Country Code:	

Output Address

Address:	MD_Address
Address 2:	MD_Address2
City:	MD_City
State/Province:	MD_State
Zip/Postal Code:	MD_Zip
Country Code:	
Address Key:	

An easy-to-use interface enables you to easily select the fields from your input data and direct them to the correct fields in your output data.

Data Quality Components for SSIS

SCAN TO WATCH A SHORT VIDEO

For a Free Trial, Go to MelissaData.com/msdn-dqc
Call 1-800-MELISSA (635-4772)

MELISSA DATA®
Your Partner in Data Quality

Creating Media Apps for Windows Phone

Mark Hopkins

Windows Phone is a great platform for consuming media. The built-in media player and the Windows Phone Marketplace are my main sources of listening material for both music and my favorite podcasts. I listen at the gym, in the car, on the bus and even while I'm getting ready for my day. I love media apps that help me discover material I might not otherwise know about. For example, I learn about a lot of new music through the built-in Smart DJ feature, as well as various other streaming music apps. Several apps allow me to access video content provided by YouTube and Vimeo.

If you're interested in creating media apps for Windows Phone, you need to put a number of pieces in place to integrate well with the OS. In particular, you'll want to make sure your app participates and surfaces in the Music + Videos Hub.

This article discusses:

- Integrating your app with the Music + Videos Hub
- Populating the Now Playing tile
- Adding History and New items
- Determining how your app was launched
- Integrating with the Apps page
- Required graphics files
- Playing media
- Marketplace certification

Technologies discussed:

Windows Phone

Integrating with the Music + Videos Hub

"Hubs" provide a convenient place for users to find their content. The Music + Videos Hub is the go-to place for media on a Windows Phone-based device. You go there not only to see your content, but also because it provides instant access to the media that's most relevant to you. Based on research done by the Windows Phone team, we've learned that the most common things users want to do are:

1. Resume something they were playing.
2. Get to the content they play most frequently.
3. Find content that they've just added to their phone.

The Music + Videos Hub puts content in these three categories front and center while also aggregating media apps and the local media library in a single experience. **Figure 1** shows the full panorama view of the Music + Videos Hub on Windows Phone.

Besides being great for end users, the Music + Videos Hub provides a great deal of value to your applications. It requires no proactive user customization, so your content is easily discoverable. If you have an application that provides playback of music or video content, you should integrate with the Music + Videos Hub. It's an important part of what the Window Phone experience is all about.

There are four key integration points that your application can take advantage of in the music and videos experience:

1. Populating the Now Playing tile.
2. Adding items to the New and History lists.
3. Determining if your app was launched from a New or History item.
4. Integrating into the Apps list.

Populating the Now Playing Tile

This tile is the most prominent real estate in the Music + Videos Hub, displaying a relatively large image of whatever content is currently paused or was last played. Tapping on this tile will either resume or initiate playback of the content displayed. By populating this image you both promote your content and create an additional entry point into your app's playback experience. Update the Now Playing item by setting the MediaHistory.Instance.NowPlaying property, as shown in the following code snippet (I'm using C# in this article):

```
MediaHistoryItem mediaHistoryItem = new MediaHistoryItem();

// <hubTileImageStream> must be a valid ImageStream.
mediaHistoryItem.ImageStream = <hubTileImageStream>;
mediaHistoryItem.Source = "";
mediaHistoryItem.Title = "NowPlaying";
mediaHistoryItem.PlayerContext.Add("keyString", "Song Name");
MediaHistory.Instance.NowPlaying = mediaHistoryItem;
```

Adding History Items

Content that has been played most recently on the phone can be displayed in a History tile by calling the MediaHistory.Instance.WriteRecentPlay method. Just like the Now Playing tile, History tiles display images and can be tapped to launch playback of the content they contain. Here's how to update the History list:

```
MediaHistoryItem mediaHistoryItem = new MediaHistoryItem();

// <hubTileImageStream> must be a valid ImageStream.
mediaHistoryItem.ImageStream = <hubTileImageStream>;
mediaHistoryItem.Source = "";
mediaHistoryItem.Title = "RecentPlay";
mediaHistoryItem.PlayerContext.Add("keyString", "Song Name");
MediaHistory.Instance.WriteRecentPlay(mediaHistoryItem);
```

Adding New Items

"New" tiles function the same as History tiles, but display recently added phone content. For example, a New tile might promote a music file that's just been downloaded to the phone. However, these tiles can also be used for things such as newly created radio stations or playlists. The same action of initiating playback on tile-touch applies here. The following code demonstrates how to update the New list:

```
MediaHistoryItem mediaHistoryItem = new MediaHistoryItem();

// <hubTileImageStream> must be a valid ImageStream.
mediaHistoryItem.ImageStream = <hubTileImageStream>;
mediaHistoryItem.Source = "";
mediaHistoryItem.Title = "MediaHistoryNew";
mediaHistoryItem.PlayerContext.Add("keyString", "Song Name");
MediaHistory.Instance.WriteAcquiredItem(mediaHistoryItem);
```

Determining How Your App Was Launched

History and New tiles should only be used to start playback and shouldn't be used as generic launch points into your application. It's OK if your app is launched and your UI is displayed. The important thing is that the music or video should start with a single tap, so the experience is efficient and consistent.

To determine if your app was launched from a History or New tile, start by overriding the OnNavigatedTo virtual method. The information in the NavigationContext is used to determine the media associated with the item, in this case a song from the media library on the device. The song starts playing in the Loaded event handler for the PhoneApplicationPage after the page has finished loading. For more information about this, download the Music + Videos Hub Sample in the MSDN Library at bit.ly/y0tEiX.

The code in **Figure 2** shows how to determine whether the application was launched from an item in the History or New list.

Integrating with the Apps Page

If your application calls one or more of the aforementioned APIs, it will automatically appear in the Apps list inside the Hub, in addition to appearing in the main applications list. This makes it easier to find and access your application and makes it a part of the core media experience. This automatic integration happens when the App Hub submission and certification process detects that your application calls the MediaHistory and MediaHistoryItem APIs and changes the application manifest to reflect that it's a media app by setting the HubType attribute. You can set this yourself prior to submission—for testing purposes only—by updating the <App> element in the WMAppManifest.xml file in your Windows Phone project:

```
<App xmlns="" ... HubType="1">
```

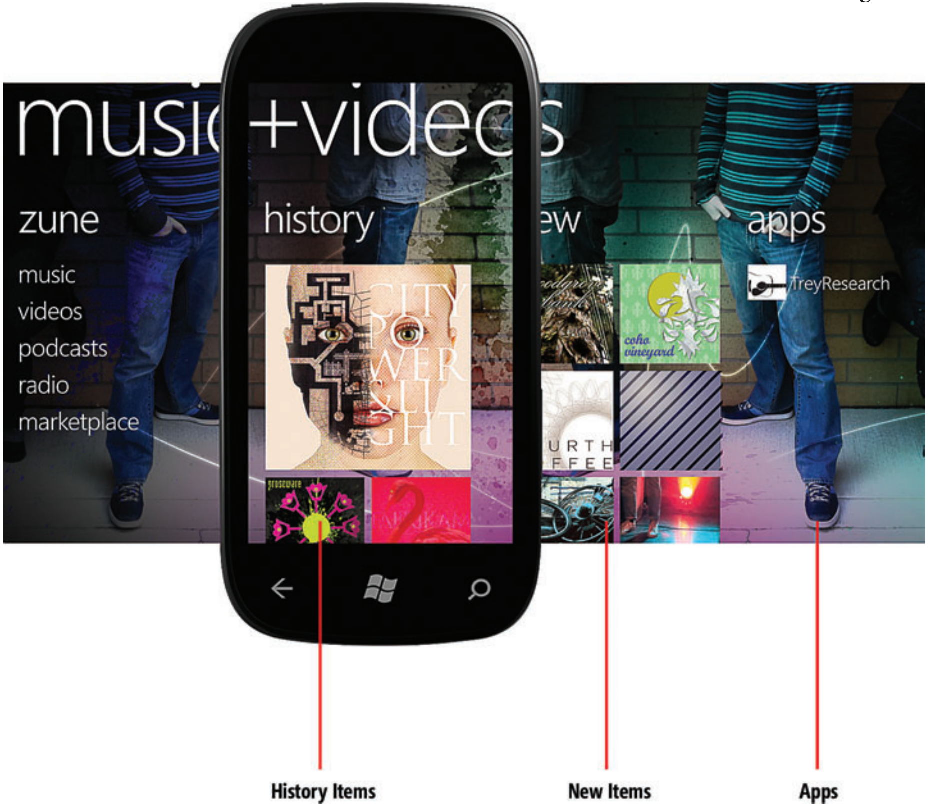


Figure 1 **The Music + Videos Hub on Windows Phone**

This will allow your program to show up in the Apps list while testing. Note that the AppHub submission process will overwrite this file before publishing based on the APIs it detects that you're calling in your application.

Required Graphics Files

Because the Music + Videos Hub is content-centric, the amount of real estate that each application gets is based on usage. This means the more your app gets used, the more prominent your content will be. For customers who use your application regularly, the Music + Videos Hub will become filled with your content, giving you an excellent promotional opportunity.

Because the Music + Videos Hub aggregates content from many different sources, it's important that you think about visual tile design, which will distinguish your app from others. What this means will vary based on your app. Keep in mind that, for example, the user will have generic album art shown in that same list for his actual albums. If you have a radio station that just happens to be playing a song from an album, your radio station app should display a tile that is visually distinct from the album cover artwork you'd see when playing the song from the media library. Consider using tiles that show your brand and artwork for the content.

Tiles that are displayed in the Music + Videos Hub must conform to the following iconography rules:

- You must include your application title or logo on each tile.
- The Now Playing tile must be 358 x 358 pixels. The file size must be 75KB or less.
- Other tiles must be 173 x 173 pixels.
- The Title property of the MediaHistoryItem class must be set to text that represents the content, such as a station name or video title.

To help avoid confusion for users, tiles shouldn't contain album art unless the album plays when the tile is pressed. If your application plays a stream, the tile graphic should describe the stream that's being played. While this isn't a certification requirement, it is a best practice.

Figure 2 **Determining Whether an App Was Launched from an Item in the History or New List**

```
// Indicates whether the app was launched from a MediaHistoryItem.
bool _historyItemLaunch = false;

// Key for MediaHistoryItem key-value pair.
const String _playSongKey = "keyString";

// The song to play.
Song _playingSong = null;

protected override void OnNavigatedTo(System.Windows.Navigation.
NavigationEventArgs e)
{
  MediaLibrary library = new MediaLibrary();

  if (NavigationContext.QueryString.ContainsKey(_playSongKey))
  {
    // The app launched from a history item.
    // Change _playingSong even if something was already playing
    // because the user directly chose a song history item.

    // Use the navigation context to find the song by name.
    String songToPlay = NavigationContext.QueryString[_playSongKey];

    foreach (Song song in library.Songs)
    {
      if (0 == String.Compare(songToPlay, song.Name))
      {
        _playingSong = song;
        break;
      }
    }

    // Set a flag to indicate that the app started from a
    // history item and that it should immediately start
    // playing the song after the UI has finished loading.
    _historyItemLaunch = true;
  }
}

private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
{
  if (_historyItemLaunch)
  {
    // Launched from a history item, start playing the song.
    if (_playingSong != null)
    {
      MediaPlayer.Play(_playingSong);
    }
  }
}
```

Playing Media

When playing video, the experience is similar whether you create an XNA Framework or a Silverlight app. In an XNA Framework app, use the MediaPlayerLauncher class, which launches the Windows Phone media player when you call MediaPlayerLauncher.Show. In Silverlight, use the MediaElement API, which allows more customization of the user experience—such as the look and placement of controls—but ultimately uses the Windows Phone media player to actually display the video. For audio, it gets a bit more interesting.

MediaElement vs. SoundEffect

In an XNA Framework application, use the SoundEffect class (or the related SoundEffectInstance or DynamicSoundEffectInstance classes) to play audio. These classes support playback of WAV audio sources only. However, you have greater control over playback and can use some nice features, such as the ability to play up to 16 sounds at once and have them mixed together on output.

In a Silverlight application, you can use the MediaElement class for playing audio in addition to video. MediaElement supports playback

Figure 3 **My Silverlight StartGameLoop Simulates an XNA Framework Game Loop**

```
// Constructor
public MainPage()
{
  InitializeComponent();
   StartGameLoop();
}

private void StartGameLoop()
{
  // Timer to simulate the XNA game loop (SoundEffect
  // class is from the XNA Framework).
  GameTimer gameTimer = new GameTimer();
  gameTimer.UpdateInterval = TimeSpan.FromMilliseconds(33);

  // Call FrameworkDispatcher.Update to update the XNA Framework internals.
  gameTimer.Update += delegate { try { FrameworkDispatcher.Update(); } catch { } };

  // Start the GameTimer running.
  gameTimer.Start();

  // Prime the pump or you'll get an exception
  // on the first XNA Framework call.
  FrameworkDispatcher.Update();
}
```

Windows Phone
Put people first.

Pick your own challenge start date*

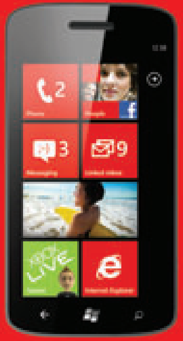
30

to Launch

YOUR IDEA | YOUR APP | 30 DAYS

Build a Windows Phone app in 30 days - with a weekly schedule, daily tips and great rewards including a chance to get a Nokia Lumia 800*

Get started at www.30tolaunch.com



*See details and Official Rules at www.30tolaunch.com

Microsoft®

Figure 4 **Overriding OnNavigatedTo and OnNavigatedFrom to Pause Audio**

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  // If the MediaPlayer is already playing music,
  // pause it upon entering the app.
  PauseMediaPlayer();
}

protected override void OnNavigatedFrom(NavigationEventArgs e)
{
  // If the MediaPlayer was already playing music,
  // resume playback as the user leaves the app.
  ResumeMediaPlayer();
}

private void PauseMediaPlayer()
{
  // See the MainPage Constructor earlier in the
  // article where the GameTimer object is created.
  // This enables the use of the XNA Framework MediaPlayer
  // class by pumping the XNA FrameworkDispatcher.

  // Pause the media player if it's already playing music.
  if (!MediaPlayer.GameHasControl)
  {
    MediaPlayer.Pause();
    resumeMediaPlayerAfterDone = true;
  }
}

private void ResumeMediaPlayer()
{
  // If music was playing, resume playback.
  if (resumeMediaPlayerAfterDone)
  {
    MediaPlayer.Resume();
  }
}
```

of WAV, WMA, MP3 and other audio sources. See the Supported Media Codecs for Windows Phone (bit.ly/aflZrb) for a complete list of audio formats playable on Windows Phone. MediaElement only allows one sound to play at a time. If another sound is already playing, MediaElement will stop it when you initiate playback of a new sound.

In a Silverlight application, you can also use the SoundEffect class to play audio. In order to use an XNA Framework API in a Silverlight application, you'll need to simulate the Game loop found in XNA Framework applications. To do this, I usually create a method called StartGameLoop to set up a GameTimer. I call the StartGameLoop method from the constructor of my PhoneApplicationPage derived class (see **Figure 3**). GameTimer is a new class available in the Windows Phone 7.1 SDK to facilitate Silverlight and XNA Framework integration.

After implementing this timer loop, you can call XNA Framework APIs elsewhere in your PhoneApplicationPage derived class.

Marketplace Certification

When an application calls the MediaHistory or MediaHistoryItem classes, it's considered to be a Music + Videos Hub application and will appear in the Apps list when installed on the phone. The submission process detects that the application uses these classes and automatically updates the hub type to Music + Videos in the Windows Phone application manifest file.

Several certification requirements regarding media playback must be followed in order for your app to be accepted into the Marketplace. At the time this article was written, these were documented in sections 6.4 and 6.5 of the Application Certification Requirements for Windows Phone at bit.ly/kN6N7Z. Certification requirements are always subject to change and are updated on a regular basis, so for any of the following guidance, make sure you're up to speed on the most recent version.

The certification requirements dictate that you can't interrupt the music a user has playing when your app launches. This makes sense because a user might enjoy listening to her favorite music while she interacts with a game, for example. Sections 6.5.1 and 6.5.2 address this particular scenario.

Of additional interest is section 6.5.3, which states:

> *An application may interrupt currently playing music on the phone to play a non-interactive full motion video, or a non-interactive audio segment (e.g. cut-scene or media clip) without asking for user consent.*
>
> *An application must resume the music that was previously playing, once the application is closed.*

So, how do you pause and restart the user's music? If you're creating a Silverlight app, you'll need to set up the simulated Game loop, as detailed earlier.

I experimented with pausing the currently playing audio, playing my sound effect and immediately restarting the audio that was playing. This created a jarring user experience and really diminished the impact of the sound effect I wanted to play. So I ended up pausing the currently playing audio when the user navigates to my page, then restarting it when she navigates away. I did this by overriding the OnNavigatedTo and OnNavigatedFrom methods. Inside those overridden methods, I call helper functions to pause and restart the audio, as shown in **Figure 4**.

Be a Good Citizen

To recap, when creating media applications for the Windows Phone OS there are several things to consider beyond the low-level details of handling actual media.

Make sure you integrate with the Music + Videos Hub so your application contributes to the user experience in the ways users have come to expect from all media apps. This means populating the History and New tiles with your content and making sure your program shows up in the Apps list.

Read and understand the certification requirements surrounding media apps, so your application is more likely to pass through the Windows Phone Marketplace submission process.

By following these guidelines you ensure that users will have a head start on knowing how to interact with your app. You'll also raise the visibility of your app and your content by making sure they surface in the proper areas in the Music + Videos Hub. And your app will be a "good citizen" on Windows Phone. ■

MARK HOPKINS *is a senior programming writer on the Windows Phone Developer Documentation Team. He has been employed at Microsoft since 1992, working on developer-focused products including developer support, Visual C++, MFC, Windows Platform SDK, Internet Explorer SDK, Tablet PC SDK, Surface SDK and Windows Phone SDK. He's also a musician and dedicated Seattle Sounders FC fan.*

THANKS to the following technical experts for reviewing this article: *Andrew Byrne, Kim Cameron, Robert Lyon, Nitya Ravi, Cheryl Simmons and Matt Stroshane*

What do you get when you build smartphones with world class hardware and software?

You get the newest Windows Phones.

You get Nokia Lumia!

See how you can create the Amazing Everyday at http://developer.nokia.com/Lumia

NOKIA Developer

Develop HTML5 Windows Phone Apps with Apache Cordova

Colin Eberhardt

This article introduces Apache Cordova, a framework for creating cross-platform mobile applications using HTML5 and JavaScript, and shows how it can be used to develop applications for Windows Phone.

Windows Phone and its native development platform allow you to create beautiful Metro-style applications with ease. With the recent Nokia partnership, Windows Phone is starting to find its way into more and more pockets.

This article discusses:

- Problems solved by Apache Cordova
- How to get prerequisite tools
- The anatomy of a Windows Phone Cordova application
- Developing Apache Cordova applications
- Architecture choices
- The Twitter Search demo application
- Using JavaScript in Visual Studio
- The MVVM application structure
- Creating a Metro UI with CSS
- Managing the back stack
- State persistence

Technologies discussed:

Windows Phone, Apache Cordova, HTML5, JavaScript

Code download available at:

code.msdn.microsoft.com/mag201205Cordova

Recent data published by research firm Gartner Inc. predicts a promising future for the Microsoft OS (bit.ly/h5lc32), with significant market share in a fragmented market. If you're developing a smartphone application, this market fragmentation means you either have to choose which OS to target or write the same application multiple times using the diverse range of languages these phones require (C#, Java and Objective-C).

Smartphones have a highly capable browser, in many ways more capable than their desktop counterparts.

However, there *is* another way. All these smartphones have a highly capable browser, in many ways more capable than their desktop counterparts, where some people still use archaic browsers! Modern smartphones allow you to create applications that run within the browser using a combination of HTML5, JavaScript and CSS. With these technologies you can potentially write a single browser-based application that runs across a diverse range of smartphone devices.

Introducing Apache Cordova

You can create an HTML5-based mobile application by creating a public Web page with JavaScript and HTML5 content and directing

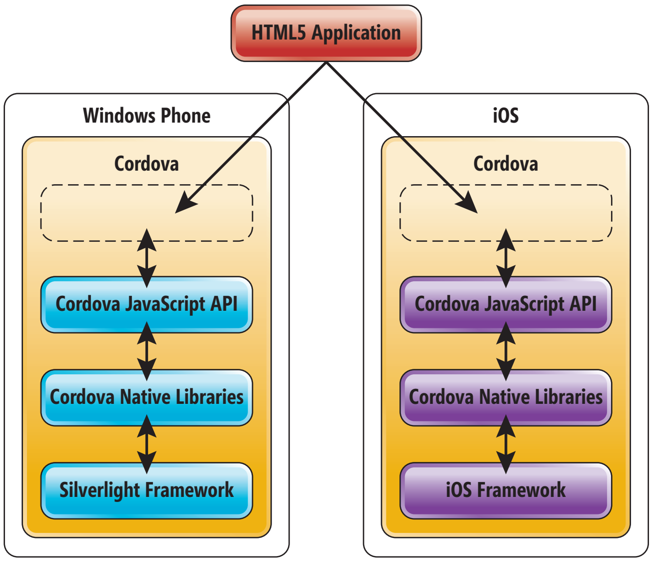


Figure 1 **Cordova Allows the Same HTML5 Application to Run Across a Range of Mobile OSes**

people to the hosting URL. However, there are a couple of problems with this approach. The first is the distribution model through online marketplaces and stores. You can't submit the URL that hosts your Web app to a marketplace, so how can you monetize it? The second problem is how to access the phone's hardware. There are no widely supported browser APIs for accessing phone contacts, notifications, cameras, sensors and so on. Apache Cordova (just Cordova hereafter for brevity) is a free and open source framework that solves both of these problems.

Cordova started life as PhoneGap, which was developed by Nitobi. In October 2011 Nitobi was acquired by Adobe Systems Inc., with the PhoneGap framework being open-sourced under the Apache Software Foundation and rebranded as Cordova. This transition is still underway.

Cordova provides an environment for hosting your HTML5/JavaScript content within a thin native wrapper. For each smartphone OS, it uses a native browser control to render your application content, with the application assets being bundled into the distributable. With Windows Phone, your HTML5 assets are packaged within the XAP file and loaded into isolated storage when your Cordova application starts up. At run time, a WebBrowser control renders your content and executes your JavaScript code.

Cordova also provides a set of standard APIs for accessing the functionality that's common across different smartphones. Some of these functionalities include:

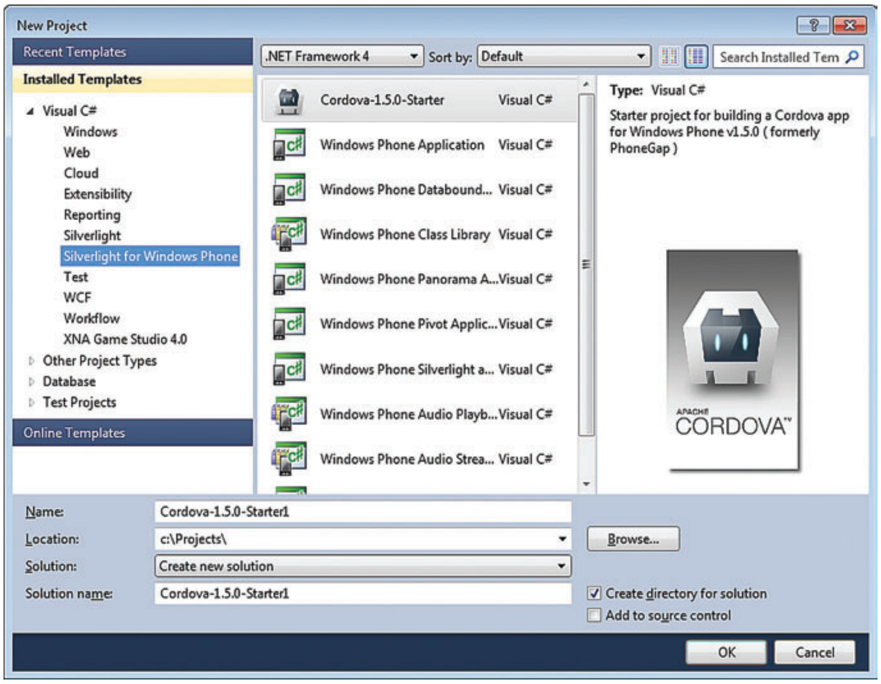
- Application lifecycle events
- Storage (HTML5 local storage and databases)
- Contacts
- Camera
- Geolocation
- Accelerometer

Cordova provides an environment for hosting your HTML5/JavaScript content within a thin native wrapper.

Each one of the preceding functionalities is exposed as a JavaScript API, which you use from your JavaScript code. Cordova does all the hard work involved in providing the required native implementation, ensuring that you work against the same JavaScript APIs, regardless of the phone OS your code is running on, as illustrated in **Figure 1**.

The bulk of this article discusses Cordova from the perspective of a Windows Phone developer, where development takes place within Visual Studio and you test your application on the emulator or a physical device. While Cordova is a cross-platform technology, you typically develop using your editor or IDE of choice, so an iOS developer would develop a Cordova application in Xcode and an Android developer would most likely use Eclipse.

Cordova also has a cloud-based build service called Build (build.phonegap.com), where you can submit your HTML5/JavaScript content. After a short time it returns distributions for most of the Cordova-supported platforms. This means you don't need to have copies of the various


New Project
Recent Templates
Installed Templates
Visual C#
Windows
Web
Cloud
Extensibility
Reporting
Silverlight
Silverlight for Windows Phone
Test
WCF
Workflow
XNA Game Studio 4.0
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by:
Default
Search Installed Tem
Cordova-1.5.0-Starter Visual C#
Windows Phone Application Visual C#
Windows Phone Databound... Visual C#
Windows Phone Class Library Visual C#
Windows Phone Panorama A... Visual C#
Windows Phone Pivot Applic... Visual C#
Windows Phone Silverlight a... Visual C#
Windows Phone Audio Playb... Visual C#
Windows Phone Audio Strea... Visual C#
APACHE CORDOVA™
Browse...
OK
Cancel

Type: Visual C#

Starter project for building a Cordova app for Windows Phone v1.5.0 (formerly PhoneGap)

Name: Cordova-1.5.0-Starter1

Location: c:\Projects\

Solution: Create new solution

Solution name: Cordova-1.5.0-Starter1

☑ Create directory for solution

☐ Add to source control

Figure 2 **Cordova for Windows Phone Includes a Visual Studio Template**

platform-specific IDEs (or a Mac computer) in order to build your application for a range of platforms. The Build service is the property of Adobe and is currently in beta and free to use. It will remain free for open source projects.

Acquiring the Tools

It's assumed you already have Visual Studio, the Windows Phone SDK and (optionally) Zune set up for Windows Phone development. If not, you can obtain the tools for free by downloading Visual Studio 2010 Express for Windows Phone (bit.ly/dTsCH2).

You can use any of the standard JavaScript/HTML5 libraries or frameworks in your Cordova application, as long as they're compatible with the phone's browser.

You can obtain the latest Cordova developer tools from the PhoneGap Web site (phonegap.com), although future releases will be distributed via Apache (incubator.apache.org/cordova). The download includes the templates, libraries and scripts required to develop Cordova applications across all the supported platforms. You'll want to use the Windows Phone version, of course.

Once you've downloaded the Cordova tools, follow the Windows Phone Get Started Guide (phonegap.com/start#wp) and install the Visual Studio template. Creating a "Hello World"-style application is as simple as creating a new project based on the supplied template, as shown in **Figure 2**.

If you build and deploy the project created by the template to your emulator, you should be greeted with the message "Hello Cordova," as shown in **Figure 3**.

The Anatomy of a Windows Phone Cordova Application

Although you can develop a Cordova application without much knowledge of how it works under the hood, it's worthwhile understanding what the various files generated by the template are, as shown in **Figure 4**.

Focusing only on the Cordova files in **Figure 4** (from top to bottom), note the following:

- **GapLib/WP7CordovaClassLib.dll** is the Cordova assembly. This contains the Windows Phone native implementation of the Cordova APIs.
- **www** is the folder where you place your application assets, HTML5, JavaScript, CSS and images. The template generates a basic index.html file and the master.css stylesheet.
- **www/cordova-1.5.0.js** provides the Windows Phone implementation of the Cordova JavaScript APIs. This interfaces with the native code contained within WP7CordovaClassLib.
- **BuildManifestProcessor.js** is a JavaScript file that's invoked by a post-build step. This file generates the CordovaSourceDictionary.xml file, ensuring that anything you add to the www folder will be loaded into isolated storage.
- **CordovaSourceDictionary.xml** is a generated XML file that lists all of your application assets. When your application first launches, this XML file indicates the files to be loaded into isolated storage.

The MainPage.xaml file contains an instance of the CordovaView control, a user control that contains a WebBrowser control:

```
<Grid x:Name="LayoutRoot">
  <my:CordovaView Name="PGView" />
</Grid>
```

When the app starts, the CordovaView control takes care of loading your application assets into local storage and navigating to the www/index.html file, thus launching your application. You can, of course, place other Silverlight controls in the page by editing this XAML, although I wouldn't recommend it. If you're writing an HTML5 application, your intention is probably to make this work cross-platform. Any controls you add to MainPage.xaml will of course be specific to your Windows Phone build.

Developing Cordova Applications

You can add your HTML, JavaScript and CSS files to the www folder and—as long as you mark them with a Build Action of Content—they'll be included in your project and accessible via the browser control when your application executes. You can use any of the standard JavaScript/HTML5 libraries or frameworks in your Cordova application, as long as they're compatible with the phone's browser.

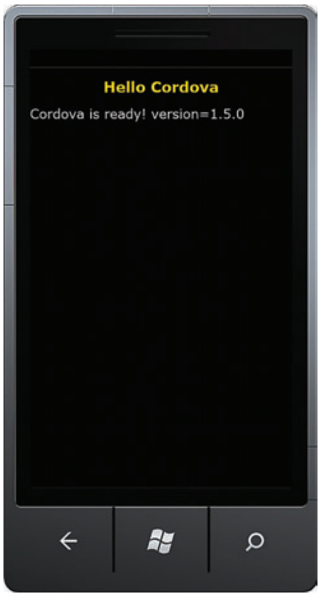


Figure 3 **The Cordova Template Application Running on an Emulator**

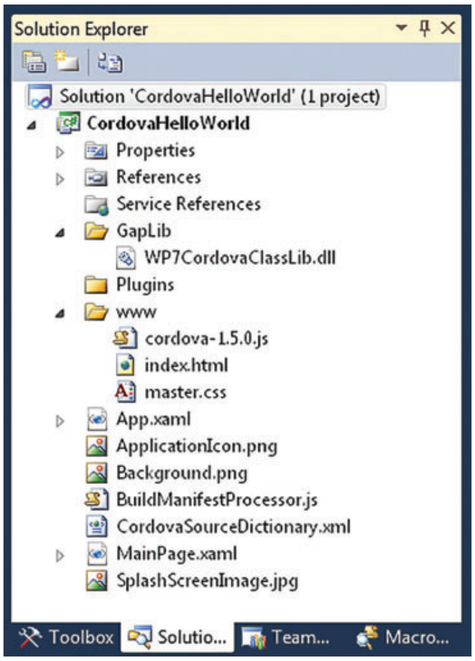


Figure 4 **The Cordova Template Folder Structure**

With ScaleOut GeoServer, the sky's no limit.

Global Data Integration

ScaleOut StateServer's industry-leading in-memory data grid technology is the key to fast data access and scalable performance. With ScaleOut GeoServer, now your applications can seamlessly access memory-based data worldwide.

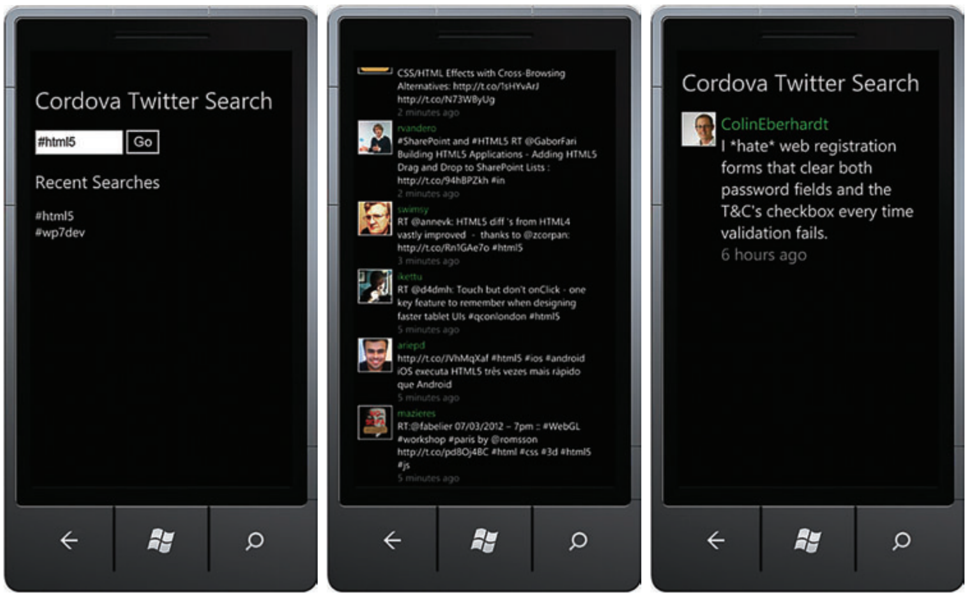
Give Your Apps...

- **Blazing Speed**
- **Global Data Access**
- **Transparent Data Migration**

Take your application's scalability to new heights. Download your FREE trial copy of ScaleOut StateServer® and ScaleOut GeoServer® today!

SCALEOUT SOFTWARE
In-Memory Data Grids for the Enterprise

www.scaleoutsoftware.com | 503.643.3422



Cordova Twitter Search

#html5 Go

Recent Searches

#html5
#wp7dev

CSS/HTML Effects with Cross-Browsing Alternatives: http://t.co/1sHYvArJ http://t.co/N73WByUg
2 minutes ago

rvandero
#SharePoint and #HTML5 RT @GaborFari Building HTML5 Applications - Adding HTML5 Drag and Drop to SharePoint Lists : http://t.co/94hBPZkh #in
2 minutes ago

swimsy
RT @annevk: HTML5 diff 's from HTML4 vastly improved - thanks to @zcorpan: http://t.co/Rn1GAe7o #html5
3 minutes ago

ikettu
RT @d4dmh: Touch but don't onClick - one key feature to remember when designing faster tablet UIs #qconlondon #html5
5 minutes ago

ariepd
http://t.co/JVhMqXaf #html5 #ios #android iOS executa HTML5 três vezes mais rápido que Android
5 minutes ago

mazieres
RT:@fabelier 07/03/2012 – 7pm :: #WebGL #workshop #paris by @romsson http://t.co/pd8Oj4BC #html #css #3d #html5 #js
5 minutes ago

Cordova Twitter Search

ColinEberhardt
I *hate* web registration forms that clear both password fields and the T&C's checkbox every time validation fails.
6 hours ago

Figure 5 **The Cordova Twitter Search Demo Application**

The Cordova APIs are documented on the Cordova Web site; I won't describe them in detail here. One important thing to note is that you must wait for the deviceready event before making use of any of the other API methods. If you inspect the index.html file generated from the template, you can see that it waits until the device is ready before updating the UI:

```
<script type="text/javascript">
  document.addEventListener("deviceready",onDeviceReady,false);

  function onDeviceReady()
  {
    document.getElementById("welcomeMsg").innerHTML
      += "Cordova is ready! version=" + window.device.cordova;
    console.log(
      "onDeviceReady. You should see this " +
        "message in Visual Studio's output window.");
  }
</script>
```

The console object used in the preceding code allows you to add debug output to your application. These messages are sent to the Visual Studio console by Cordova.

Single-Page or Multipage Application Architecture

When building Cordova applications, you can employ two distinct patterns:

- **Multipage applications:** In multipage applications, multiple HTML pages are used to represent the various screens of your application. Navigation between pages uses the standard browser mechanics, with links defined by anchor tags. Each HTML page includes script references to the Cordova JavaScript code and your application JavaScript.
- **Single-page applications:** In single-page applications, a single HTML file references Cordova and your application JavaScript. Navigation between the various pages of your application is achieved by dynamically updating the rendered HTML. From the perspective of the phone browser, the URL remains the same and there's no navigation between pages.

The choice you make between these two patterns has a significant impact on the structure of your code.

Generally speaking, the multipage pattern is best suited to applications that mostly comprise static content. With this approach you can take HTML/CSS/JavaScript that's currently used on your Web site and package it, using Cordova, for delivery to the phone as an application. But the multipage approach has some disadvantages. First, when the browser navigates from one page to the next, it has

Figure 6 **The Knockout ApplicationViewModel**

```
/// <reference path="..//intellisense.js" />

/*globals ko*/

function ApplicationViewModel() {
  /// <summary>
  /// The ViewModel that manages the ViewModel back-stack.
  /// </summary>

  // --- properties

  this.viewModelBackStack = ko.observableArray();

  // --- functions

  this.navigateTo = function (viewModel) {
    this.viewModelBackStack.push(viewModel);
  };

  this.back = function () {
    this.viewModelBackStack.pop();
  };

  this.templateSelector = function (viewModel) {
    return viewModel.template;
  }
}
```

Figure 7 **The TwitterSearchViewModel**

```
/// <reference path="..//intellisense.js" />

/*globals $ application ko localStorage SearchResultsViewModel TweetViewModel*/

function TwitterSearchViewModel() {
  /// <summary>
  /// A ViewModel for searching Twitter for a given term.
  /// </summary>

  // --- properties

  this.template = "twitterSearchView";
  this.isSearching = ko.observable(false);
  this.searchTerm = ko.observable("");

  // --- public functions

  this.search = function () {
    /// <summary>
    /// Searches Twitter for the current search term.
    /// </summary>

    // implementation detailed later in this article ...
  };
}
```

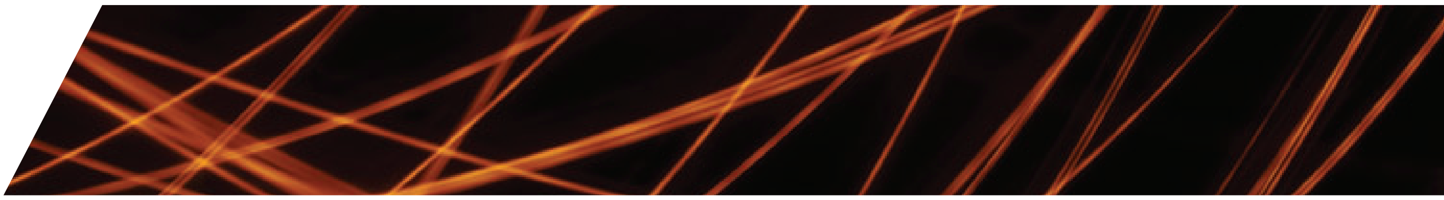
Microsoft GOLD CERTIFIED Partner

DXv2 is here

Simply Stunning.

Today's customers want smart, beautiful apps in every part of their lives, from work to home. Now, with the tools from DevExpress, you can build them better than ever. Apply sophisticated themes, create your own, and even integrate your corporate brand. Incorporate elegant office-inspired or business intelligence dashboard controls into your design so your users can view and manipulate data in more insightful and compelling ways. And do it all on the platforms you know best. DXv2 delivers the tools you need to inspire and be inspired by application design.

DXv2 is the next generation of tools that can take your applications to a whole new level. Your users are ready—what will you build for them? Download your free 30-day trial at **www.DevExpress.com**



DevExpress

to reload and parse all the JavaScript associated with the new page. There's a noticeable pause as the Cordova lifecycle, which creates the link between the JavaScript APIs and C# counterparts, is executed. Second, because your JavaScript code is being reloaded, all application state is lost.

The single-page pattern overcomes the issues associated with the multipage approach. The Cordova and application JavaScript code is loaded just once, resulting in a more responsive UI and removing the need to pass application state from one page to the next. The only disadvantage to this approach is the added complexity, with JavaScript code being required to update the UI when navigation occurs.

The demo application described in this article uses the single-page pattern. For an example of the multipage approach in action, I recommend looking at the DemoGAP CodePlex project (demogap.codeplex.com), which provides a simple demonstration of the Cordova API features within a Windows Phone application.

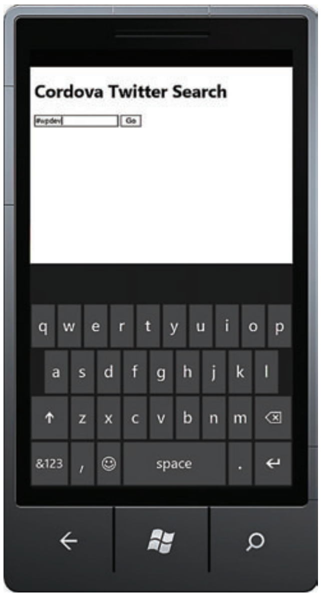


Figure 8 **The TwitterSearch-ViewModel Rendered via the twitterSearchView Template**

The Demo Application

The rest of this article describes "Cordova Twitter Search," a simple Windows Phone application that allows the user to search Twitter based on one or more keywords, as shown in **Figure 5**.

As well as Cordova, this application makes use of the following frameworks:

- **jQuery and jQuery Templates:** jQuery has become the de facto standard framework for manipulation of the browser Document Object Model (DOM). jQuery templates are plug-ins that Microsoft developed (bit.ly/8ZO2V1), making it easy to create reusable HTML templates that can be rendered to the DOM. Twitter Search uses jQuery templates to define the UI for the various pages within the application.
- **Knockout JS:** Knockout is a Model-View-ViewModel (MVVM) framework that makes it easy to construct ViewModels and keep them synchronized with Views in a manner familiar to Silverlight developers. It's this familiarity that led me to choose Knockout over the numerous other suitable JavaScript UI frameworks.

I won't be covering Knockout in detail in this article. If you're interested in learning more about this framework, I recommend reading John Papa's recent article, "Getting Started with Knockout" (msdn.microsoft.com/magazine/hh781029). And if you aren't familiar with the MVVM pattern (where have you been hiding?), I recommend this excellent article by Josh Smith: "WPF Apps with the Model-View-ViewModel Design Pattern" (msdn.microsoft.com/magazine/dd419663).

Developing JavaScript Applications with Visual Studio

Before delving into the details of the application, I want to say a few things about JavaScript application development. One of the challenges facing the JavaScript developer is the dynamic nature of the language. With JavaScript you aren't constrained by a rigid type system; instead, objects can be built dynamically. This poses a challenge to the developers of JavaScript editors and IDEs. With strongly typed languages such as C# and Java, the type information can be used to provide enhanced code navigation, refactoring and IntelliSense. With JavaScript, on the other hand, the lack of type information means that the IDE typically provides far fewer developer aids.

Fortunately, things have improved recently, with Visual Studio 2010 performing pseudo-execution of your JavaScript code in order to determine the "shape" of each object, allowing it to provide JavaScript IntelliSense. In order to take full advantage of the IntelliSense support, we have to provide the IDE with a few "hints" in the form of "references" that tell the IDE which files to include in its pseudo-execution. With the demo project, all files start with a reference comment that tells the IDE to include the intellisense.js file. The content of this file is simply a list of references that ensure the IDE includes all the important application JavaScript files, providing quality IntelliSense support across the application, as shown here:

```
/// Ensure IntelliSense includes all the files from this project.
///
/// <reference path="app.js" />
/// <reference path="viewModel/ApplicationViewModel.js" />
/// <reference path="viewModel/SearchResultsViewModel.js" />
/// <reference path="viewModel/TweetViewModel.js" />
/// <reference path="viewModel/TwitterSearchViewModel.js" />
/// <reference path="lib/jquery-1.6.4.js" />
/// <reference path="lib/cordova-1.5.0.js" />
/// <reference path="lib/knockout-1.2.1.js" />
```

Figure 9 **Code to Follow the Metro Theme**

```
body
{
  background: #000 none repeat scroll 0 0;
  color: #ccc;
  font-family: Segoe WP, sans-serif;
}

h1
{
  font-weight: normal;
  font-size: 42.667px; /* PhoneFontSizeExtraLarge */
}

button
{
  background: black;
  color: white;
  border-color: white;
  border-style: solid;
  padding: 4px 10px;
  border-width: 3px; /* PhoneBorderThickness */
  font-size: 25.333px; /* PhoneFontSizeMediumLarge */
}

input[type="text"]
{
  width: 150px;
  height: 34px;
  padding: 4px;
}
```

5 YEARS OF EXCELLENCE

XCEED
DataGrid
for WPF

Mature, feature-packed, and lightning-fast. The most adopted and trusted WPF datagrid.

IBM®
U2 SystemBuilder™

"IBM U2 researched existing third-party solutions and identified Xceed DataGrid for WPF as the most suitable tool. The datagrid's performance and solid foundation take true advantage of WPF and provide the extensibility required for IBM U2 customers to take their applications to the next level in presentation design and styling."

Vincent Smith
U2 Tools Product Manager at IBM

Microsoft®
Visual Studio® Team System 2010

"Using Xceed DataGrid for WPF in Microsoft Visual Studio System 2010 helped us greatly reduce the time and resources necessary for developing all the data presentation feature we needed. Working with Xceed has been a pleasure."

Norman Guadagno
Director of Product Marketing
for Microsoft Visual Studio Team System

XCEED
Professional Themes
for WPF

Theme your entire app in minutes. Flawless styles for all official WPF controls.

XCEED
DataGrid
for Silverlight

Incredible streaming technology. Speed up your app and say goodbye to paging.

XCEED
Ultimate ListBox
for WPF

The world's first streaming listbox. Simple, drop-in upgrade to the WPF listbox.

NEW

XCEED
Ultimate ListBox
for Silverlight

Fast and fluid, with ground-breaking streaming technology.

NEW

Visual Studio PARTNER

XCEEd
MULTI-TALENTED COMPONENTS

JavaScript is a relaxed and forgiving language, with features such as value coercion and semicolon insertion making it easy to use in a scripting environment. However, these same features often become problematic when managing large quantities of code. For that reason I highly recommend using JSLint, a tool that applies a more rigid set of coding standards to JavaScript. A popular Visual Studio Extension adds JSLint support (jslint4vs2010.codeplex.com), reporting lint errors within the error console. I've used JSLint for the Twitter Search application (and virtually every other JavaScript project I've worked on).

You might notice a "globals" comment at the start of each JavaScript file within the project. JSLint helps prevent variables "leaking" into global scope by accidental omission of the var keyword. The "globals" comment provides JSLint with a formal definition of which variables are allowed to occupy a global scope.

The MVVM Application Structure

The Twitter Search application is, from the perspective of the phone's browser control, a single-page application. Conversely, from the user perspective, it has multiple pages, as seen in **Figure 5**. In order to support this, a Knockout ViewModel is constructed that contains a stack of ViewModel instances, each one representing a page within the application. As the user navigates to a new page, the corresponding ViewModel is added to this stack, and when the user navigates back, the topmost ViewModel is popped off the stack (see **Figure 6**).

I highly recommend using JSLint, a tool that applies a more rigid set of coding standards to JavaScript.

When the application starts, an instance of the ApplicationViewModel is created and bound to the UI using Knockout:

```
document.addEventListener("deviceready", initializeViewModel, false);

var application;

function initializeViewModel() {
  application = new ApplicationViewModel();
  ko.applyBindings(application);
}
```

The UI itself is quite simple, comprising a div element that uses the Knockout template binding to render the ViewModel stack:

```
<body>
  <h1>Cordova Twitter Search</h1>

  <div class="app"
    data-bind="template: {name: templateSelector,
                          foreach: viewModelBackStack}">
  </div>
</body>
```

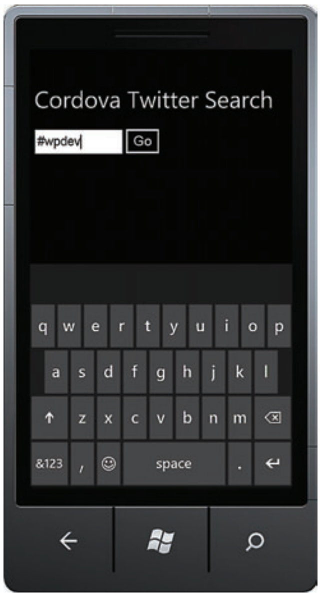


Figure 10 **The Twitter Search Application with a Metro CSS Style Applied**

The Knockout template binding works in a similar manner to the Silverlight ItemsControl in that it binds to an array of ViewModel instances and is responsible for generating the View for each via a template. In this case, the templateSelector function is invoked on the ApplicationViewModel in order to determine the named template for each ViewModel.

If you run this application you'll find that it doesn't actually do anything—that's because there aren't any ViewModels to represent the pages of the application!

The TwitterSearchViewModel

I'll introduce the first ViewModel, TwitterSearchViewModel, which represents the first page of the application. This ViewModel exposes a few simple observable properties that support the UI, namely the searchTerm, which is bound to the user input field, and isSearching, which is a Boolean observable that disables the search button when the Twitter APIs are being queried via HTTP. It also exposes a search function that's bound to the search button, in much the same way that you would bind an ICommand to a Button within Silverlight (see **Figure 7**).

The template property of the ViewModel names the View that's associated with this ViewModel. This View is described as a jQuery template within the index.html file:

```
<script type=text/x-jquery-tmpl" charset="utf-8" id="twitterSearchView"
  <div>
    <form data-bind="submit: search">
      <input type="text"
        data-bind="value: searchTerm, valueUpdate: 'afterkeydown'" />
      <button type="submit"
        data-bind="enable: searchTerm().length > 0 &&
          isSearching() == false">Go</button>
    </form>
  </div>
</script>
```

If you add an instance of the TwitterSearchViewModel to the application ViewModel stack, the application now shows the first page, as shown in **Figure 8**.

Creating a Metro UI with CSS

One of the most striking features of the Windows Phone OS is the Metro design language, which guides all aspects of the phone's look and feel. This design language, which favors content over chrome, is not only pleasing to the eye, it's also practical, delivering highly legible interfaces on the small phone form factor.

The current UI, as shown in **Figure 8**, uses the standard browser styling and as a result isn't very pleasing to the eye! There are already a few well-established frameworks for creating good-looking mobile UIs using HTML and CSS, such as jQuery Mobile (jquerymobile.com). Currently these frameworks tend to concentrate on emulating the iOS look and feel. A Windows Phone Cordova application *could* be styled using jQuery Mobile, although it would probably face some user rejection because it simply wouldn't "fit" with the overall look and feel of the OS.

Fortunately, the chrome-free Metro theme is actually quite easy to replicate using HTML and CSS. In fact, Windows 8 treats HTML

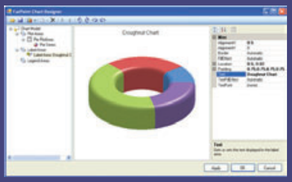
GrapeCity PowerTools

Spreadsheets at their best!

The world's most popular Microsoft® Excel®-compatible spreadsheet components for .NET Windows Forms and ASP.NET development.

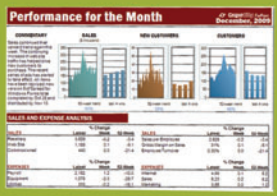
"It's fast. It does what we need it to do. It just works."

Jeffrey Baron
VP App Software Development



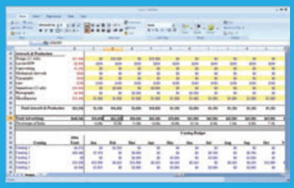
"We used Spread as a total back-end engine for doing calculations and everything"

Jay Kimble
Software Developer



"Most people think that they just need a grid. Well, Spread is a grid, but it also has all of that Excel goodness in it."

Carl Franklin
Host, .NET Rocks!



"We built this tax software where the auditors actually go on site, enter the data and can remotely synch into the master head-quarters."

Rasesh Shah
Developer Evangelist

"Spread delivers a unique grid layout for our software solutions and lets us add user interface components to the grid layout."

Philippe Jolidon
Software Engineer RD

Download your FREE Trial Today!

GCPowerTools.com/Spread

SP Spread

GCPowerTools.com • GvTv.GCPowerTools.com

Figure 11 **The TwitterSearchViewModel Search Function**

```
this.search = function () {
  /// <summary>
  /// Searches Twitter for the current search term.
  /// </summary>

  this.isSearching(true);

  var url = "http://search.twitter.com/search.json?q=" +
    encodeURIComponent(that.searchTerm());

  var that = this;

  $.ajax({
    dataType: "jsonp",
    url: url,
    success: function (response) {

      // Create an array to hold the results.
      var tweetViewModels = [];

      // Add the new items.
      $.each(response.results, function () {
        var tweet = new TweetViewModel(this);
        tweetViewModels.push(tweet);
      });

      // Navigate to the results ViewModel.
      application.navigateTo(new SearchResultsViewModel(tweetViewModels));

      that.isSearching(false);
    }
  });
};
```

as a first-class citizen, allowing you to develop HTML5 Metro applications using the Windows Runtime APIs.

By introducing the correct fonts, font sizes and colors via some simple CSS (shown in **Figure 9**), you can create a UI that closely follows the Metro theme, as shown in **Figure 10**.

One final aspect that's a bit of a giveaway that this is an HTML5 application rather than a native one is that the user can still "pinch" the UI in order to make it zoom. This can be partially solved by adding the following meta property to the index.html page:

```
<meta name="viewport" content="user-scalable=no" />
```

One of the most striking features of the Windows Phone OS is the Metro design language, which guides all aspects of the phone's look and feel.

This informs the browser that the user isn't allowed to scale the rendered content. Unfortunately, the way this is implemented by the Windows Phone browser allows the user to scale the content, but snaps back to the original scale when the interaction ends. This doesn't look terribly good!

I've discovered that, by inspecting the visual tree of the WebBrowser control, it's possible to attach handlers to the manipulation events and prohibit them from bubbling up to the native TileHost that renders the HTML5 content. I've published a brief blog post (bit.ly/vU2o1q) that includes a simple utility class that achieves this. But you should use this with caution because it depends on the internal structure of the WebBrowser control, which may well change in future versions of the Windows Phone OS.

Figure 12 **The TweetViewModel**

```
/// <reference path="..//intellisense.js" />

/*globals application*/

function TweetViewModel(tweet) {
  /// <summary>
  /// A ViewModel that represents a single tweet
  /// </summary>
  /// <param name="tweet">A tweet as returned by the twitter search API</param>

  // --- properties

  this.template = "tweetDetailView";
  this.author = tweet.from_user;
  this.text = tweet.text;
  this.id = tweet.id;
  this.time = tweet.created_at;
  this.thumbnail = tweet.profile_image_url;

  // --- public functions

  this.select = function () {
    /// <summary>
    /// Selects this tweet, causing the application to navigate to a tweet-view.
    /// </summary>
    application.navigateTo(this);
  };
}
```

Searching Twitter

If you look in more detail at the TwitterSearchViewModel search function, it queries the Twitter APIs via the jQuery "ajax" function, which returns a JSONP response. A TweetViewModel instance is constructed from each of the returned tweets, and these are used to construct a SearchResultsViewModel instance (see **Figure 11**).

The SearchResultsViewModel simply contains a list of tweets:

```
/// <reference path="..//intellisense.js" />

/*globals ko*/

function SearchResultsViewModel(tweetViewModels) {
  /// <summary>
  /// A ViewModel that renders the results of a twitter search.
  /// </summary>
  /// <param name="tweetViewModels">An array of TweetViewModel instances</param>

  // --- properties

  this.template = "searchResultsView";
  this.tweets = ko.observableArray(tweetViewModels);
}
```

And the TweetViewModel exposes the properties of an individual tweet and a select function that navigates to the individual tweet view, as shown in **Figure 12**.

Again, the templates that describe the View for each of these ViewModels are added to the index.html file, as shown in **Figure 13**.

With this code in place, when the user hits the "go" button to search Twitter, a new SearchResultsViewModel is added to the ViewModel stack. This will automatically result in the searchResultsView template being rendered within the "app" div. However, because the ViewModel stack is rendered via a foreach template binding,

Raising the Bar...

And Pie, and Spline and Scatter... on Mobile Business Intelligence

Compatible with
Microsoft® Visual Studio® 11 Beta

Visualize your BI in 50+ ways, and on just as many devices. From candlesticks to polar and radial charts, nobody offers the breadth and depth of dynamic, high fidelity, totally mobile charting solutions that you get from Infragistics' NetAdvantage. **Try a free, fully supported trial of NetAdvantage for .NET today!**

www.infragistics.com/NET

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

HTML 5

Figure 13 **Adding Templates to the index.html File**

```
<script type=text/x-jquery-tmpl" charset="utf-8" id="searchResultsView">
  <div>
    <ul data-bind="template: {name: 'tweetView',
                              foreach: tweets}"> </ul>
  </div>
</script>

<script type="text/x-jquery-tmpl" charset="utf-8" id="tweetView">
  <li class="tweet"
      data-bind="click: select">
    <div class="thumbnailColumn">
      <img data-bind="attr: {src: thumbnail}"
                             class="thumbnail"/>
    </div>
    <div class="detailsColumn">
      <div class="author"
           data-bind="text: author"/>
      <div class="text"
           data-bind="text: text"/>
      <div class="time"
           data-bind="text: time"/>
    </div>
  </li>
</script>
```

this won't hide the twitterSearchView template instances; instead they'll be stacked one above the other. This can be solved by the addition of a couple of simple CSS rules:

```
.app>div
{
  display: none;
}

.app>*:last-child
{
  display: block;
}
```

The first selector hides all the immediate children of the div marked with the app class, while the second selector, which has a higher precedence, ensures that the last child is shown.

With these CSS rules in place, the Twitter Search application is fully functional and navigable.

Managing the Back Stack

With the current Twitter Search application you can navigate from the search results page to an individual tweet, but if you hit the phone Back button the application immediately exits. This is because the application navigation occurs entirely within a browser control—hence, from the perspective of the Silverlight framework, the application has a single page. This not only results in a poor user experience, it will almost certainly result in the application being rejected if submitted to the Windows Phone Marketplace.

It's possible to attach handlers to the manipulation events and prohibit them from bubbling up to the native TileHost that renders the HTML5 content.

Fortunately, the solution to this problem is straightforward. The ApplicationViewModel contains a stack of ViewModel instances. If

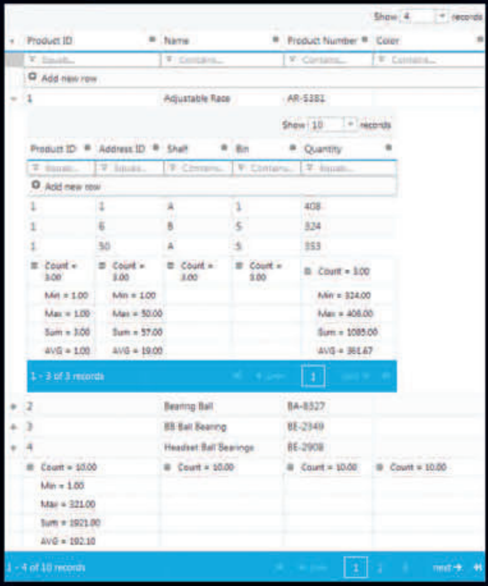
Figure 14 **Handling a Back Button Press**

```
function initializeViewModel() {

  application = new ApplicationViewModel();
  ko.applyBindings(application);

  // Handle the back button.
  application.backButtonRequired.subscribe(function (backButtonRequired) {
    if (backButtonRequired) {
      document.addEventListener("backbutton", onBackButton, false);
    } else {
      document.removeEventListener("backbutton", onBackButton, false);
    }
  });

  var viewModel = new TwitterSearchViewModel();
  application.navigateTo(viewModel);
}

function onBackButton() {
  application.back();
}
```

Figure 15 **Adding a Search Term to a List of Recent Searches**

```
function TwitterSearchViewModel() {
  /// <summary>
  /// A ViewModel for searching Twitter for a given term.
  /// </summary>

  // --- properties

  // ... some properties omitted for clarity ...

  this.recentSearches = ko.observableArray();

  // --- functions

  // ... some functions omitted for clarity ...


  this.loadState = function () {
    /// <summary>
    /// Loads the persisted ViewModel state from local storage.
    /// </summary>
    var state = localStorage.getItem("state");
    if (typeof (state) === 'string') {
      $.each(state.split(","), function (index, item) {
        if (item.trim() !== "") {
          that.recentSearches.push(item);
        }
      });
    }
  };

  function saveState() {
    /// <summary>
    /// Saves the ViewModel state to local storage.
    /// </summary>
    localStorage.setItem("state", that.recentSearches().toString());
  }

  function addSearchTermToRecentSearches() {
    /// <summary>
    /// Adds the current search term to the search history.
    /// </summary>

    that.recentSearches.unshift(that.searchTerm());
    saveState();
  }
}
```

MetroTactual
[me-troh tak-choo-uhl]



Show 4 records

	Product ID	Name	Product Number	Color
	Add new row			
−	1	Adjustable Race	AR-5381	

Show 10 records

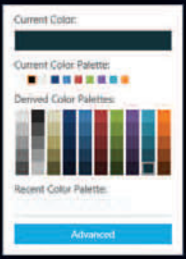
Product ID	Address ID	Shelf	Bin	Quantity
Add new row				
1	1	A	1	408
1	6	B	5	324
1	50	A	5	353
Count = 3.00	Count = 3.00	Count = 3.00	Count = 3.00	Count = 3.00
Min = 1.00	Min = 1.00			Min = 324.00
Max = 1.00	Max = 50.00			Max = 408.00
Sum = 3.00	Sum = 57.00			Sum = 1085.00
AVG = 1.00	AVG = 19.00			AVG = 361.67

1 - 3 of 3 records 1

	Product ID	Name	Product Number	Color
+	2	Bearing Ball	BA-8327	
+	3	BB Ball Bearing	BE-2349	
+	4	Headset Ball Bearings	BE-2908	
	Count = 10.00	Count = 10.00	Count = 10.00	Count = 10.00
	Min = 1.00			
	Max = 321.00			
	Sum = 1921.00			
	AVG = 192.10			

1 - 4 of 10 records 1 2 3 next →

Compatible with
Microsoft® Visual Studio® 11 Beta



[1886] Louis Aimé Augustin Le Prince
[1888] Roundhay Garden Scene
[1899] Classic Theatre
[1900] Singer
The 'Shaw Manufacturing Co.' (1903–1962) of Galesburg, Kansas advertises an engine kit (1903–1915) for motorizing a bicycle in 'Popular Mechanics' magazine for $56.
[1886] Gottlieb Daimler
[1898] De Dion
[1914] Smith Motor Wheel

noun, adjective

1. Modern, clean, sleek, stylish, touch-friendly design and UX
2. Feng Shui for your apps
3. Available in NetAdvantage 12.1 toolsets

See also: NetAdvantage for .NET

Try your free, fully supported trial today.
www.infragistics.com/NET

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

HTML 5

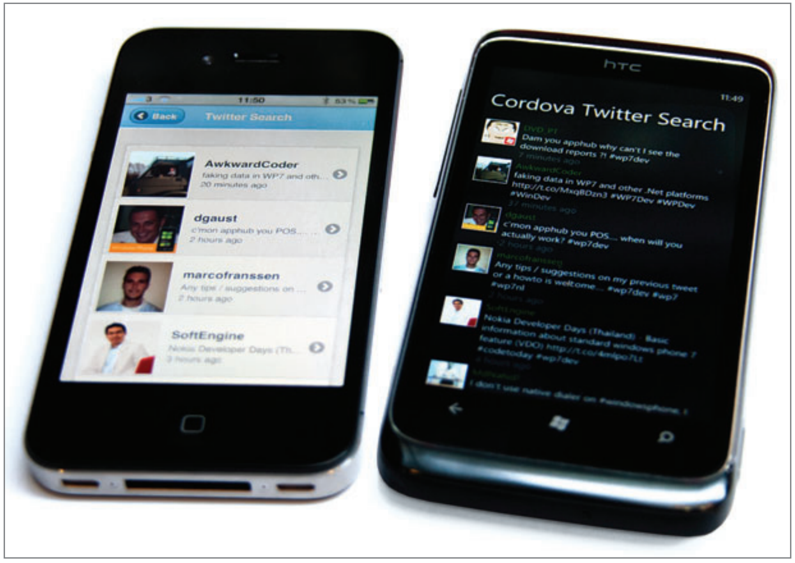


Figure 16 **Twitter Search Running on an iPhone and a Windows Phone Device**

there's more than one ViewModel instance in this stack, you need to handle the hardware Back button press and pop the topmost ViewModel off this stack. Otherwise, you can allow the Silverlight framework to exit the application.

Interestingly, while the Windows Phone browser does support local storage, this function is turned off when the browser renders pages from isolated storage.

To support this, a backButtonRequired dependent observable is added to the ViewModel:

```
function ApplicationViewModel() {

  // --- properties

  this.viewModelBackStack = ko.observableArray();

  this.backButtonRequired = ko.dependentObservable(function () {
    return this.viewModelBackStack().length > 1;
  }, this);

  // --- functions

  // ...
}
```

When the ViewModel is initialized, you can handle changes to this observable and subscribe to the backbutton events that Cordova supplies. When the event fires, you invoke the back function on the ApplicationViewModel, which pops the topmost ViewModel off the stack. The Knockout template binding takes care of removing the ViewModel's associated view from the UI and the CSS styling ensures the view that's now topmost is visible (see **Figure 14**).

Because the backbutton event is supplied via Cordova, the code in **Figure 14** will work just fine if you execute the app using a different phone OS (as long as the phone itself has a hardware Back button).

State Persistence

We'll add one final feature to the Twitter Search application: When a search returns successfully, the search term is added to a list of recent searches (see **Figure 15**).

The addSearchTermToRecentSearches function uses the Knockout convenience function, unshift, which adds an item to the start of the array. Whenever a recent search is added, the state is stored using HTML5 local storage. In this case the state of the array is converted to a comma-separated list via the toString function, and converted back via split. A more complex ViewModel would most likely save multiple property values in JSON format.

Interestingly, while the Windows Phone browser does support local storage, this function is turned off when the browser renders pages from isolated storage. In order to make the earlier code work, the Cordova team had to write a "shim" implementation of the local storage APIs that saves the state within the phone's isolated storage.

With this last change to the Twitter Search application, it's now fully functional.

Portability Proof: Running on an iPhone

As you can see, the Cordova framework makes it possible to create HTML5-based applications for Windows Phone. It's also possible to mimic the native Metro look and feel using simple HTML5 and CSS techniques, while frameworks such as Knockout allow you to properly structure your code.

This article has focused on creating applications for Windows Phone, but the Twitter Search application is portable and should run on an iPhone or Android phone without modification. But would a Metro-style application suit these phones?

As a final demonstration of the versatility of this approach, I've created an iOS version of the Twitter Search application, using jQuery Mobile to mimic the native look and feel. This makes great use of the MVVM pattern, in that only the View needs to be changed—all the ViewModel logic is exactly the same. Using the cloud-based Build service I was able to create an iOS "ipa" package and install it on an iPhone, all from a Windows machine. You can see the two applications running side by side in **Figure 16**.

The full source code for both the iOS and Windows Phone versions of the application accompanies this article. ■

COLIN EBERHARDT *is a technical architect at Scott Logic Ltd. and lead architect at Visiblox (visiblox.com), which provides charting controls for a range of Microsoft .NET Framework technologies. You can follow him on Twitter at twitter.com/ColinEberhardt.*

THANKS to the following technical experts for reviewing this article: *Olivier Bloch, Glen Gordon and Jesse MacFadyen*

Just the Right Touch

Get in touch with the new NetAdvantage for .NET 2012 V. 1 today, with a free, fully supported trial!

www.infragistics.com/NET

Compatible with
Microsoft® Visual Studio® 11 Beta

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

HTML 5

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

YOUR MAP TO THE .NET DEVELOPMENT PLATFORM

CODE ON CAMPUS

Intense Take-Home Training for Developers, Software Architects and Designers

PLATINUM SPONSOR SUPPORTED BY

esri Microsoft Microsoft Visual Studio msdn Visual Studio MAGAZINE

Register Today and Save $400!

Use Promo Code REDMAY

It All Happens Here!

Visual Studio Live! is returning to Microsoft Headquarters for the third year running! In Redmond, developers, software architects and designers will connect for five days of unbiased and cutting-edge education on the Microsoft platform – all while getting an insider's look into the house that Bill built.

Topics include:

- Windows 8, WinRT and Metro
- Silverlight / WPF
- Web
- Visual Studio 2010+ / .NET 4.0+
- SharePoint
- Cloud Computing
- Data Management
- HTML5
- Windows Phone 7

Register today for Visual Studio Live! Redmond and become a more valuable part of your company's development team.

vslive.com/redmond

Scan this code to register and learn more about what Visual Studio Live! Redmond has to offer.

PRODUCED BY 1105 MEDIA INC

MEDIA SPONSOR THE CODE PROJECT WWW.CODEPROJECT.COM

Redmond, WA

August 6-10

Microsoft Headquarters

Using HTML5 to Create Mobile Experiences

Brandon Satrom

Last month, I introduced you to CSS3 media queries (msdn.microsoft.com/magazine/hh882445), a new module that allows you to adapt page styles based on conditional rules. Though media queries have broad applicability across the device spectrum, they're often mentioned in the context of building mobile sites and applications. To wit, the introduction to media queries in the previous article was framed around the creation of tablet and mobile experiences.

Considering the difficulties building mobile sites and apps presents, it's no wonder that media queries took off. Compared with undesirable alternatives such as browser sniffing (sometimes called device detection) and having to create mobile experiences on a per-platform basis, media queries seem like a true gift. They're brilliant modules, for sure, and the reason I wrote about them last month is because you should be using them today.

This article discusses Internet Explorer 10 Consumer Preview. All information is subject to change.

This article discusses:

- Making grids and layouts fluid
- Creating flexible images
- Using HTML5 frameworks to build mobile Web apps

Technologies discussed:

CSS3, Internet Explorer 9, Internet Explorer 10 Consumer Preview, JavaScript

Code download available at:

code.msdn.microsoft.com/mag201205HTML5

Responsive Web Design, Revisited

But there's more to the story: CSS media queries are great, but they're only a piece of what you really need to create great mobile Web experiences. Last month, I mentioned the term *responsive Web design*, a term coined by Ethan Marcotte in his seminal article of the same name (bit.ly/9AMjxh). Marcotte focuses on media queries, but he also points out two other necessary practices: fluid grids and flexible images. Media queries are the engines that drive responsive, adaptive sites, but they're only effective when site design is also responsive and adaptive. This month, I'll introduce you to some ideas around these other two pillars of responsive Web design. I'll start with an overview of some up-and-coming CSS layout modules, and then discuss some techniques for making non-textual elements such as images and embedded video adaptive as well. Along the way, I'll note some frameworks and libraries that help you adopt these techniques. I'll also mention some popular frameworks for creating mobile Web applications, and wrap up with a brief discussion on using HTML5 to build "native" applications. By the time you've finished this article, you should have a solid foundation for implementing responsive Web design in your own applications.

Fluid Grids and Layouts

Using a grid for typographic design is a practice that has been around in some form or another for centuries, predating even the invention of moveable type. That two-dimensional structure made up of intersecting vertical and horizontal axes allows a designer to align and organize elements in a visually pleasing way on a layout, as illustrated in **Figure 1**. Over the last few years, Web designers have begun to apply many of the same principles to their digital work, and a number of popular frameworks, such as the 960 Grid System (960.gs) and the Semantic Grid System (semantic.gs), now make grid layouts accessible to all.

However, the direct application of a typographic grid to the Web has a key flaw: print layouts are fixed, Web layouts are not. What's more, many grid implementations aren't terribly semantic, meaning that they require the addition of markup to define the grid, mixing presentation with content in your HTML pages.

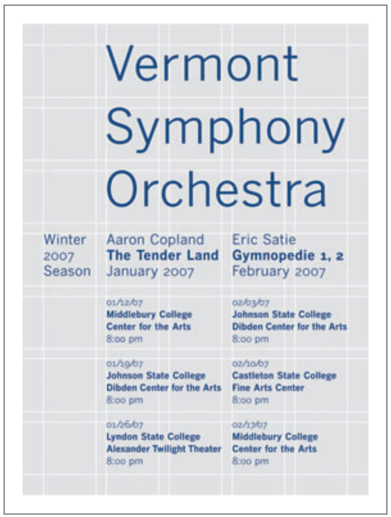


Figure 1 **A Typographic Grid**

To be truly responsive, grids (or layouts) should adapt to changing experiences.

Which brings us to the "fluid" in Marcotte's "fluid grid." To be truly responsive, grids (or layouts) should adapt to changing experiences. As I discussed last month, media queries help you define the rules for repositioning elements, but in order to be effective, those elements must first be defined in a fluid or flexible container. The tools I mentioned earlier (as well as many others) do address this issue either natively (the Semantic Grid) or through the use of a complementary library (Adapt.js for the 960 Grid—adapt.960.gs), but there are also new and emerging CSS modules that assist in the creation of fluid layouts.

Note that I'm taking some liberties with Marcotte's term fluid *grids* by restating it as fluid *layouts*. I'm doing so because some of the new CSS modules, though not based on a grid, can still help you create fluid, adaptable containers.

Let's look first at the CSS3 Flexible Box Layout Module (or Flexbox), which can be found at bit.ly/yguOHU. Flexbox is designed to help you create layout containers for elements that will automatically position children in a horizontal or vertical flow, and provide automated spacing (Goodbye "ul li { float: right; }"!). This module is supported—with vendor prefixes—in the Internet Explorer 10 Platform Preview, Firefox, Chrome, Safari, iOS Safari and Android (check out caniuse.com/flexbox for more information).

We'll start by applying Flexbox to the Photo Gallery site I introduced last month. Given the CSS in **Figure 2**, you can see the result, styled up a bit for illustration purposes, in **Figure 3**. Please note that the CSS in **Figure 2** is using only the "-ms-" vendor prefix. In the sample code available online (code.msdn.microsoft.com/mag201205HTML5), I include the other vendor prefixes (-webkit, -moz, -o), and you should do this for your sites as well.

This is nice, but of course it looks like what we already had. To illustrate the flex in Flexbox, resize your browser window, or open the page in a mobile emulator or on a device. There are no media queries defined in this example and yet the layout is acting a bit more fluid. Combine the two modules and you'll be able to create fluid containers that align, and space and shift elements in a responsive manner. For instance, you can create a media query rule for screens smaller than 480 pixels, change box-orient to vertical and voilà—you have the beginning of a mobile layout.

The CSS Grid Layout (or just CSS Grid), which can be found at bit.ly/ylx7Gq, is a newer specification, submitted to the World Wide Web Consortium (W3C) CSS Working Group in April 2011 and currently implemented only in the Internet Explorer 10 Consumer Preview. The idea is to provide robust grid support natively within the browser. For developers and designers, the result is a rich typographic grid without the need for table layouts or the presence of semantically neutral markup.

The CSS Grid enables you to define a page layout with predefined rows and columns, as well as rules that specify how content flows into and across those elements. The first step is to define a grid container, specifying "grid" as the display property for selected elements:

```
body {
  display: -ms-grid; // A vendor prefix is required.
}
```

I'm selecting the body element here, which means my grid will fill the entire document. This isn't required, and I could easily craft a grid from a smaller section of the page or even define multiple grids on a single page. Once I've defined the grid, I need to define the size of its rows and columns:

```
body {
  display: -ms-grid;
  -ms-grid-rows: 50px 30% 20% auto;
  -ms-grid-columns: 75px 25px 2fr 1fr;
}
```

Here, I'm specifying a grid of four columns and four rows, making the size absolute in a few cases (50px, 75px, for example), relative to the size of the window for a few (30%, 20%) and automatic based on the width of its content (auto). I'm also using the new fraction value unit, fr, which is defined in the CSS Grid specification as "… a

Figure 2 **CSS for Using the Flexbox Module**

```
ul.thumbnails {
  width: 100%;
  background: #ababab;
  display: -ms-box;
  -ms-box-orient: horizontal;
  -ms-box-pack: center;
  -ms-box-align: center;
}
ul.thumbnails li {
  -ms-box-flex: 1;
}
```

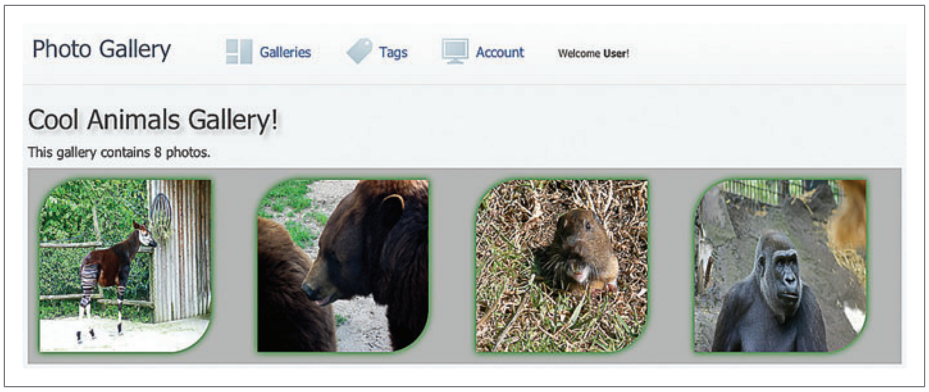


Figure 3 **Photo Gallery Images with Flexbox Applied**

fraction of the available space." Fraction values are calculated after fixed sizes are assigned and then divided proportionately among all rows and columns defined with these values. In the context of my example, this means that once 100 pixels are set aside for columns one and two, column three will be given two-thirds of the remaining space and column four will be given one-third.

With the grid defined, it's easy to position child elements within the grid by assigning row and column values, like so:

```
#main {
  -ms-grid-row: 2;
  -ms-grid-column: 2;
  -ms-grid-row-span: 2;
  -ms-grid-row-align: center;
}
```

Here, I'm placing my "main" sectioning element at the second row and column of the grid. I'm allowing that element to span two rows and am centering the content inside the container. The Microsoft Internet Explorer Test Drive Demo site uses its implementation of CSS Grid Layout to create an exact implementation of the popular Grid System site, thegridsystem.org, and you can check it out for yourself at bit.ly/gEkZkE.

If you've ever tried something similar with markup and CSS2.1, you've no doubt seen the flexibility that the CSS Grid can provide. What's more, when combined with media queries, the CSS Grid can be used to create adaptive layouts that are easy to tweak with fewer rule changes as users adjust device size and orientation.

The final layout specification I'll present is the CSS Multi-Column Layout Module, which you'll find at bit.ly/yYEdts. The CSS Multi-Column is at the "Candidate Recommendation" state (bit.ly/x5IbJv), and enjoys wide support across all of the browsers, including planned support for Internet Explorer 10. As it sounds, Multi-Column allows you to lay out columns on a page without manual positioning or floats. All you need is to apply the "column-width" property (with prefixes, where required) on a container, like so:

```
article {
  width: 960px;
  column-width: 240px;
}
```

With this rule, article elements will be split into columns of 240 pixels, creating as many columns as the container allows (in this case, it takes four 240-pixel columns to fill the 960-pixel container). I could also use the column-count property to define a fixed number of columns, in which case the column widths would be evenly distributed within the width of my container.

As with the Flexbox and Grid modules, combining the module with media queries lets you quickly define simple, adaptable rules to deliver an ideal user experience to mobile users.

The three modules I've described have a lot in common, and each has features you can use to create the kind of fluid layouts that are a requirement for truly responsive Web sites. I encourage you to research and play with each so that you can choose the appropriate module when solving a given layout challenge.

In the realm of responsive Web design—mobile in particular—media is the sticky wicket.

You'll also want to check out the emerging frameworks that leverage these specifications. Using one of these can be a nice jump-start to building fluid layouts for your own sites. Two notable frameworks are Skeleton (getSkeleton.com) and Bootstrap (twitter.github.com/bootstrap), a full site starter kit from Twitter. I recently rebuilt one of my own sites with the help of Skeleton (check it out at html5tx.com).

Responsive Media

In the realm of responsive Web design—mobile in particular—media is the sticky wicket. Let's start with images. One of the easiest ways to style images to make them more responsive is to add the following to your stylesheets:

```
img {
  max-width: 100%;
}
```

This rule will always scale your images (up or down) to fit within that image's parent container. Thus, if your container elements are responsive (perhaps using one of the techniques previously described), your images are as well.

The challenge with this method is that the images on your site need to be large enough to scale to any size that they might conceivably need to be. In the Photo Gallery site I've been using, the raw images are quite large, and thus can handle being resized.

The use of huge, resizable images, however, leads to a huge problem for mobile: overhead, and thus to potentially poor mobile experiences. Even if you're resizing a large image to

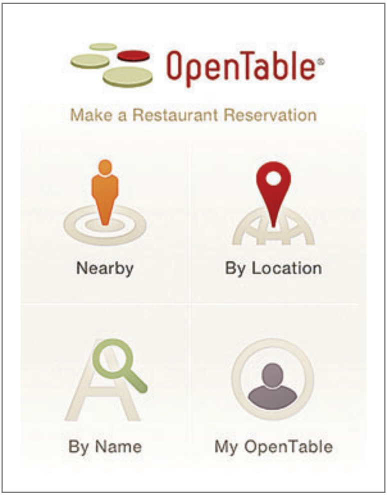


Figure 4 **A Sample Mobile Screen Built with jQuery Mobile**

/update/2012/05

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

BEST SELLER

ActiveReports 6 | from $685.02

GrapeCity PowerTools

Latest release of the best selling royalty free .NET report writer.

- Fast and flexible reporting engine
- Flexible event-driven API to completely control the rendering of reports
- Wide range of export and preview formats including viewers for WinForms, Web, Flash, PDF
- XCopy deployment
- Royalty-free licensing for Web and Windows, plus support for Azure

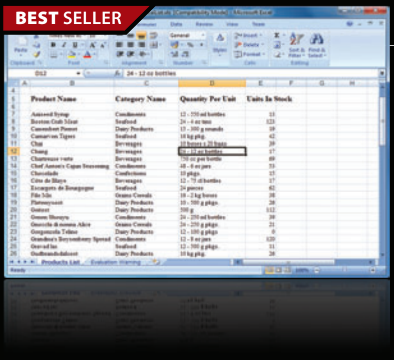
BEST SELLER

GdPicture.NET | from $3,135.02

GdPicture Imaging technologies

A full-featured document-imaging and management toolkit for software developers.

- Acquire, process, create, view, edit, annotate, compose, split, merge and print documents within your Windows & Web applications
- Read, write and convert vector & raster images in more than 90 formats, including PDF, PDF/A, TIFF , GIF, JPEG, PNG, JBIG2, WMF, BMP, WBMP, ICO, PCX, PNM, XPM, JPEG 2000, HDR, PSD, TGA, PICT, EXR, DDS, PPM, SGI, PBM, PGM, PFM, XBM, IFF and RAW camera files

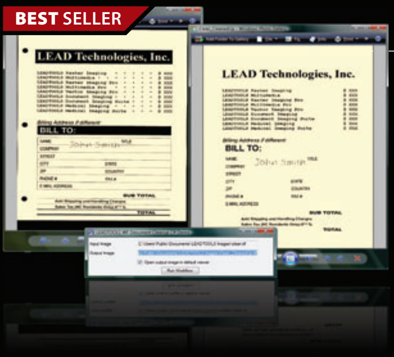

BEST SELLER

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

ASPOSE The .NET & Java Component Publisher

Every Aspose .NET component in one package.

- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Add charting, email, spell checking, barcode creation, OCR, diagramming, imaging, project management and file format management to your .NET applications
- Common uses also include mail merge, adding barcodes to documents, building dynamic Excel reports on the fly and extracting text from PDF files


BEST SELLER

LEADTOOLS Document Imaging SDK V17.5 | from $2,245.50

LEAD TECHNOLOGIES

Add powerful document imaging functionality to Windows, Web & Mobile applications.

- Recognition Add-ons including OCR, Asian OCR, Arabic OCR, ICR, OMR and MICR
- Barcode Add-ons available: 1D Barcode and 2D Barcode
- PDF Add-ons available: Raster PDF Read and Write, and Advanced PDF
- Full featured, zero footprint Image Viewers and annotations for any device that supports HTML5

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

GSA Schedule Contract GS-35F-0188R

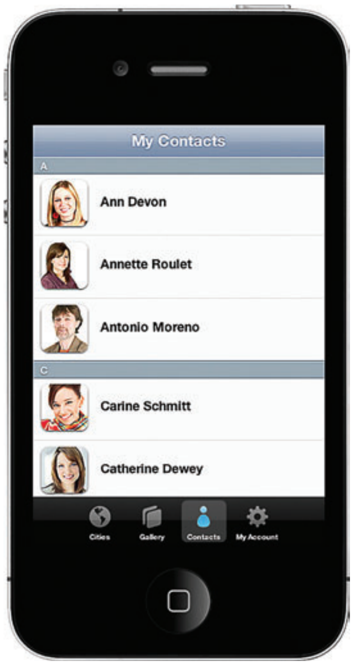


Figure 5 **A Kendo UI Mobile Demo Application Viewed on an iOS-Based Device**

fit a 320 x 240 window, the full image is transferred to the device. This means you could potentially be sending a 2MB photo when a 10KB photo is all the device requires. In a mobile world, bandwidth still matters, so the device-width strategy must be supplanted with other approaches.

Unfortunately, this is still a challenge, and the W3C doesn't yet formally support a particular approach. A number of methods exist for dealing with responsive images, but all fall into one of two categories: they either deal with the problem on the server, or attempt to do so on the client. Many server approaches, like the one described at bit.ly/rQGOKw, rely on the use of 1 x 1 pixel placeholder images, client cookies and server-rewrite rules in order to deliver the right image to the right device. Client approaches, such as the one described at bit.ly/tIdNYh, often utilize JavaScript, <noscript> fallbacks, conditional comments and some interesting tricks in CSS. Both approaches feel like hacks (because they are), but they represent the best we've been able to come up with given the constraints of the <img> tag. In the short term, you'd be wise to take a look at both approaches and decide which works for your applications.

Over the long term, however, there might be hope. In an article in *Smashing Magazine*, "HTML5 Semantics" (bit.ly/rRZ5BI), Bruce Lawson of Opera argues for the addition of a <picture> element that would behave similarly to the <audio> and <video> tags, meaning that developers would have access to multiple <source> child elements inside of the parent <picture>. When combined with inline media queries, a new <picture> element could provide a nice way to finally deliver a robust solution for responsive images:

```
<picture alt="cat gallery">
  <source src="nyan-high.png" media="min-width:800px" />
  <source src="nyan-med.png" media="min-width:480px" />
  <source src="nyan-low.png" />
  <!-- fallback for unsupporting browsers -->
  <img src="nyan-med.png" alt="cat gallery" />
</picture>
```

Though this solution has proven to be popular, and a W3C Community Group has been formed around it (bit.ly/AEjoNt), there's no formal working group that I'm aware of. Perhaps we'll see this element make it in time for HTML6.

Similar challenges exist for adding responsive video to sites and applications, though more robust solutions exist in HTML5 for video than for images. For starters, the media-query-enhanced <source> element—as illustrated with the previously mentioned fictional <picture> element—is valid for <video>, as illustrated here:

```
<video>
  <source src="nyan-mashup-high.webm" media="min-width:800px" />
  <source src="nyan-mashup-med.webm" media="min-width:480px" />
  <source src="nyan-mashup-low.webm" />
  <!-- Insert Silverlight or Flash Fallback here -->
</video>
```

If you're serving the video from your own servers or are using a service that provides multiple versions to embed, I highly recommend using this syntax to ensure that your users get a device-friendly video.

In a mobile world, bandwidth still matters.

While this solution will help save your users' bandwidth, you still need to think about sizing those embedded video elements, just as you do with images. With media queries, it's easy to adapt your video elements based on different screen sizes, but if you're looking for a solution that's a bit more automated, take a look at FitVids.js (fitvidsjs.com), a jQuery plug-in that will automatically make your video elements fluid and responsive. Keep in mind, however, that as a jQuery plug-in, this solution won't work for users with JavaScript disabled.

Building Mobile Web Apps with HTML5 Frameworks

Now that we've covered the other two pillars of responsive Web design, fluid layouts and flexible images, let's talk a bit about cases

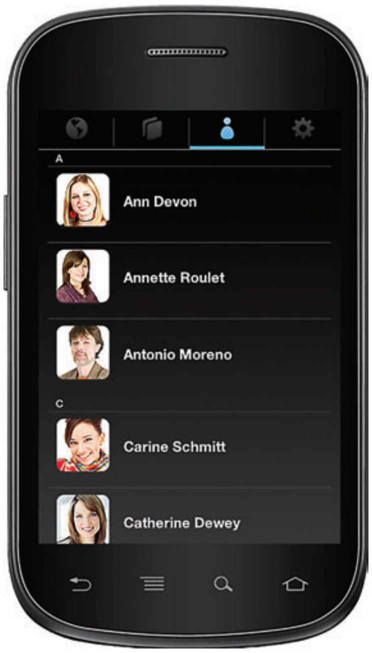


Figure 6 **A Sample Kendo UI Mobile Application Viewed on an Android-Based Device**

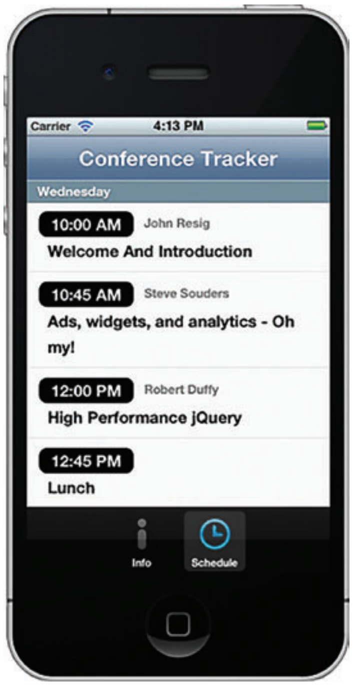


Figure 7 **An iOS Application Built with HTML, JavaScript and CSS**

where you're not just building mobile-friendly Web sites or apps—you want to specifically target the mobile experience.

In the world of development, traditional desktop (or client) and Web paradigms have given way to a third type of application, often called *native* applications because they're resident on a given device (a Windows Phone-based smartphone or iPad, for example), are developed using device-specific frameworks (iOS and Android) and are installed via an App Store or Marketplace.

As rich and robust as these frameworks are, sometimes Web developers wish to provide a similar "native feel" to their mobile Web applications. Such applications still reside on your servers and can be delivered outside of a device App Store or Marketplace.

Though you can certainly build these types of applications by hand, it's common to use frameworks to do so. One popular choice for mobile Web applications is jQuery Mobile (jquerymobile.com), a mobile development framework that provides an HTML5-based UI system for targeting nearly every mobile platform. **Figure 4** shows an example of the mobile application for OpenTable.com, which was built using jQuery UI.

Another popular option for building mobile applications with a native look and feel is Kendo UI Mobile (kendoui.com), an HTML5, JavaScript and CSS framework from Telerik Inc. Kendo UI allows you to create mobile applications that look fully native on iOS- and Android-based devices—and to do so with a single codebase. **Figure 5** and **Figure 6** show this feature in action, which you can check out for yourself at bit.ly/wBgFBj.

Building Native Applications with HTML5

Providing a native feel to Web applications is a great way to not only leverage your skills as a Web developer, but also to create applications that conform to a user's expectations in a mobile setting. Still, these applications can only go so far in leveraging native sensors and APIs on those devices. While some features—like geolocation—are provided to mobile browsers, many—like the accelerometer or video—are not. In order to access these features, native applications are still the way to go.

The great news, however, is that the popularity of Web programming has enabled HTML5, JavaScript and CSS to "go native," as it were. Popular frameworks such as PhoneGap (phonegap.com) and Titanium Appcelerator (appcelerator.com) enable you to use HTML5 and JavaScript to build native applications for iOS, Android and Windows Phone, with device API access to boot. What's more, mobile development frameworks like jQuery Mobile and Kendo UI Mobile work just as well in these environments as they do in the browser. **Figure 7** shows a native iOS application built using PhoneGap and Kendo UI. For more information, check out the blog post at bit.ly/zpIAPY.

Microsoft has taken native Web development to a new level by formally adding support for building Windows 8 applications using HTML5, JavaScript and CSS, all with no additional abstractions or frameworks required. You can check out the Consumer Preview of Windows 8, as well as the new developer tools for these platforms, at dev.windows.com.

When it comes to the mobile Web, you've got choices! If you're a Web programmer who wants to build native experiences, complete with augmented reality features, check out Windows 8 or a framework such as PhoneGap or Titanium Appcelerator. If you're just looking for a native feel in the browser, look at jQuery UI and Kendo UI Mobile. Finally, if your goal is to create a single Web site or application that responds to many devices and experiences, try the responsive strategies I discussed in this and last month's article. There's no question that mobile is one of the hottest platforms for development right now. Your best bet, no matter which of the strategies or platforms you choose, is to make mobile your top development priority. ■

BRANDON SATROM *is product manager for Kendo UI (kendoui.com), an HTML5 and mobile toolset from Telerik Inc. He blogs at userinexperience.com, can be followed on Twitter at twitter.com/brandonsatrom and is in the middle of frantically writing the book, "Building Metro Style Apps for Windows 8 in JavaScript," with Chris Sells.*

THANKS to the following technical experts for reviewing this article: *Jon Box, Burke Holland and Clark Sell*

The world leader in advanced Microsoft SQL Server Integration Services (SSIS) tasks, components and scripts

SAS® Adapters
Reusable Scripts
USPS Address Parse
Parallel Loop
EDI Source
Table Difference
And More ....

CozyRoc is rare breed among technology companies

Over 100 Reusable Components

COZYROC™
Go to the next level

www.cozyroc.com
sales@cozyroc.com
(919) 249-7421

What's New for Mobile Development in ASP.NET MVC 4

Keith Burnell

For a long time, developers have been looking for the holy grail of tooling that allows a single codebase to reach every platform, and this is more important than ever today. Because of the increasing popularity and variety of mobile smartphones and tablets around the world, it's crucial that your site have a pleasant and usable mobile interface. You could certainly go the native application route and create specific applications for iOS, Android, Windows Phone, BlackBerry and so forth, but unfortunately, native applications require platform-specific skill sets and code.

Fortunately, HTML5 and CSS3 appear to be the toolset that will finally allow you to truly "write once, run anywhere." Although not yet finished specifications, HTML5 and CSS3 are quickly becoming industry standards for multitargeted, browser-based sites, with most of the features already supported by the major browsers. The great thing about the HTML5/CSS3 combo is that it's just straight HTML markup that can be leveraged from any HTML-based toolset—from PHP to ASP.NET MVC.

This article discusses a prerelease version of ASP.NET MVC 4. All information is subject to change.

This article discusses:

- Adaptive rendering
- Creating views for specific browsers and devices
- Installing JQuery Mobile
- Using the Mobile Application project template

Technologies discussed:

HTML5, CSS3, ASP.NET MVC 4, jQuery Mobile

This article will dive into the new features of ASP.NET MVC 4, many of which leverage the browser-agnostic capabilities of HTML5 and CSS3 to make it easier for you to develop sites targeted specifically at mobile browsers, as well as those aimed at both mobile and desktop browsers.

If you don't yet have ASP.NET MVC 4 installed and are running Visual Studio 2010, you can download the bits from the Web Platform Installer or directly from the ASP.NET MVC Web site (asp.net/mvc/mvc4). Visual Studio 11 includes ASP.NET MVC 4.

Adaptive Rendering

Adaptive rendering is the process of displaying a Web site differently depending on the capabilities of the targeted device and browser. Many adaptive rendering techniques have been included with ASP.NET MVC 4.

Viewport Meta Tag By default, browsers—even those on mobile devices and tablets—assume they're rendering pages on the desktop and display content at a width of 980 pixels and zoomed out. **Figure 1** shows a site created using the default Internet site template from ASP.NET MVC 3. Notice that even though the site is being displayed in a mobile browser, the browser still assumes it's rendering for the desktop. Although the site rendered in this fashion is still usable, it's certainly not ideal.

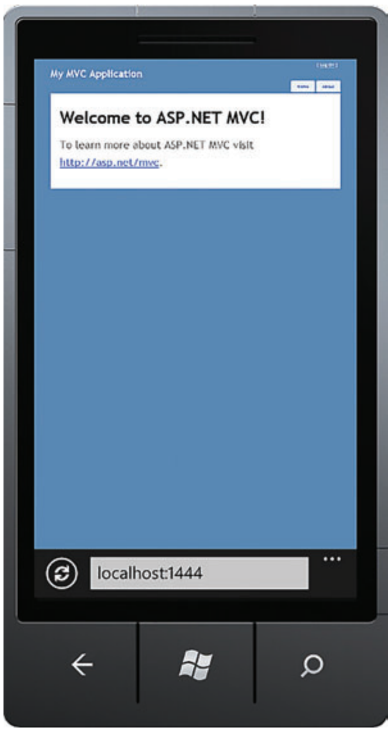


Figure 1 **The Default Display Created Using ASP.NET MVC 3**

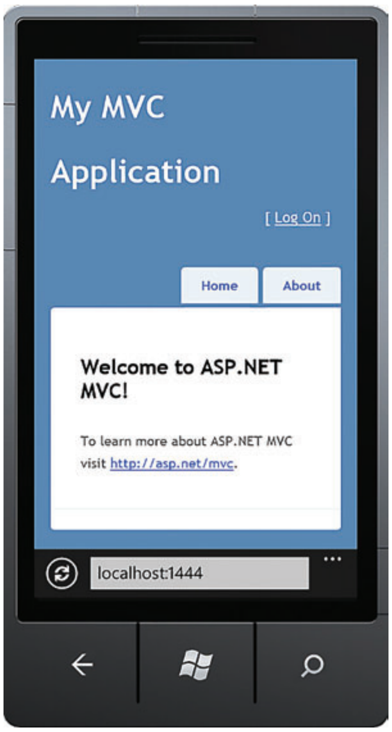


Figure 2 **Scaling to the Device with ASP.NET MVC 4**

Figure 2 shows the page after the meta viewport tag was added to the _Layout.cshtml. The site is now scaled to the width of the device. If you create a new ASP.NET MVC 4 project using any of the project templates (except the Web API template), you can open up the _Layout.cshtml file and you'll see the viewport meta tag in the head section.

CSS Media Queries When you're developing a multitargeted site, you generally want mobile users to see a view that's different from the one your desktop users see. Typically, the functionality is nearly the same, but the style and display of the content are not. With CSS media queries you can conditionally apply specific CSS styles based on the capabilities of the browser making the request to your Web site:

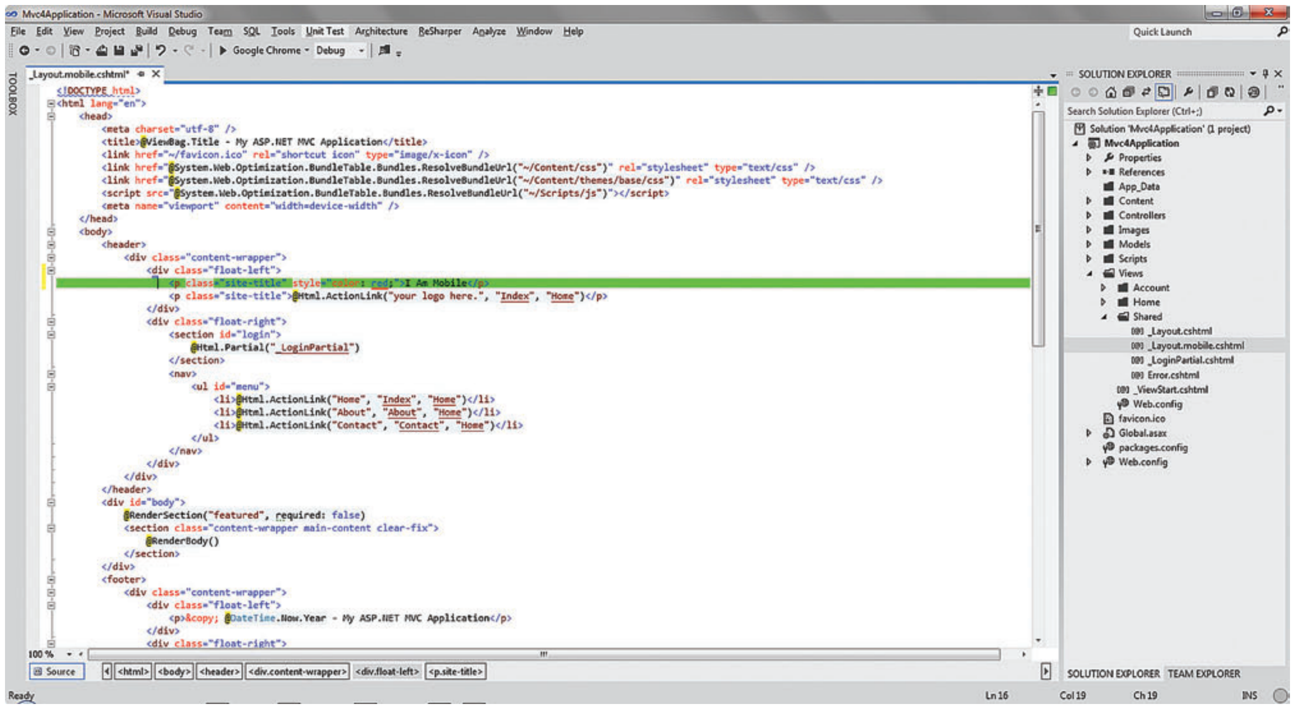
```
@media only screen and (max-width: 850px) {
  header{
    float: none;
  }
}
```

This media query will apply the contained styles only when the media is screen, not print, and the maximum width of the area the site is being rendered on is less than 850 pixels. This technique allows you to change the style of your content in any conceivable fashion based on the rendering capabilities of the browser.

Instead, you can use the viewport meta tag to tell the browser explicitly the width, height and scale to render the content. You can also configure the viewport meta tag to render based on the capabilities of the device:

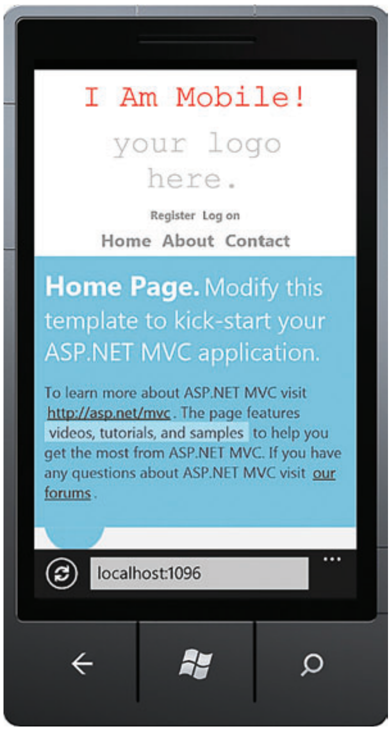
```
<meta name="viewport" content="width=device-width" />
```

When dealing with mobile browsers, you always have to take bandwidth into account. Generally when you're using a mobile device, you aren't connected to Wi-Fi or another high-speed network, so when you serve up your site to these devices you'll want to do so in the most efficient manner possible. For example,



```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title - My ASP.NET MVC Application</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <link href="@System.Web.Optimization.BundleTable.Bundles.ResolveBundleUrl("~/Content/css")" rel="stylesheet" type="text/css" />
        <link href="@System.Web.Optimization.BundleTable.Bundles.ResolveBundleUrl("~/Content/themes/base/css")" rel="stylesheet" type="text/css" />
        <script src="@System.Web.Optimization.BundleTable.Bundles.ResolveBundleUrl("~/Scripts/js")"></script>
        <meta name="viewport" content="width=device-width" />
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title" style="color: red;">I Am Mobile</p>
                    <p class="site-title">@Html.ActionLink("your logo here.", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("About", "About", "Home")</li>
                            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
                <div class="float-right">
```

Figure 3 **Mobile-Specific _Layout.cshtml**


I Am Mobile!

your logo here.

Register Log on

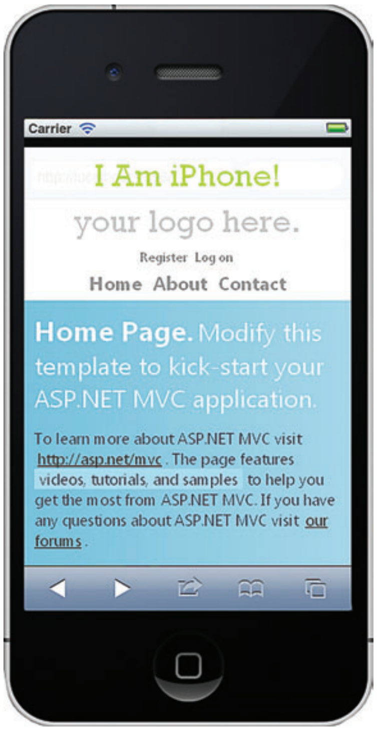
Home About Contact

Home Page. Modify this template to kick-start your ASP.NET MVC application.

To learn more about ASP.NET MVC visit http://asp.net/mvc . The page features videos, tutorials, and samples to help you get the most from ASP.NET MVC. If you have any questions about ASP.NET MVC visit our forums .

localhost:1096

Figure 4 **Mobile-Specific View**


Carrier

I Am iPhone!

your logo here.

Register Log on

Home About Contact

Home Page. Modify this template to kick-start your ASP.NET MVC application.

To learn more about ASP.NET MVC visit http://asp.net/mvc . The page features videos, tutorials, and samples to help you get the most from ASP.NET MVC. If you have any questions about ASP.NET MVC visit our forums .

Figure 5 **iPhone-Specific View**

if images aren't necessary to the functionality of your site, don't include them in mobile views. If you do need images, make sure to serve images of the correct size—that is, be sure to actually send the image sized to the dimensions at which it is to be displayed. Just as you can specify an image with CSS, with CSS media queries you can specify different images based on the capabilities of the device and browser.

All of the default ASP.NET MVC 4 project templates, excluding the Web API template, include some examples of CSS media queries. To demonstrate this, create and run a new ASP.NET MVC 4 project with the Internet Application project template. Maximize your browser and then slowly start reducing the size of the window. Once you get the window size to 850 pixels or less, you'll notice many changes to the display of the default page. To get details on the exact changes, take a look at the Site.css file starting with line 466.

When It Just Can't Be Done with CSS Alone

Sometimes modifying the styles isn't enough to get your site rendered and usable on all devices. In such cases, your only real option is to create views tailored to the browser types and devices you want to reach.

Serving up Mobile-Specific Views The idea of serving up specific pages based on the browser making the request isn't new. Developers have been browser sniffing for a long time. In earlier versions of ASP.NET MVC you could create a custom view engine implementation that inherits from either the Web Forms or Razor view engines, override the FindView method, add some browser sniffing, and return a view specific to the consuming browser. There are two new features in ASP.NET MVC 4 that allow you to accomplish this at different levels with a lot less effort.

New functionality in ASP.NET MVC 4 lets you globally override views for mobile devices using convention rather than configuration. When ASP.NET MVC 4 is servicing a request from a mobile browser and it's determining what view to serve, it first looks for views with the naming convention of [*view*].mobile.cshtml. If a view matching the naming convention is found, ASP.NET MVC will serve it up; otherwise it will fall back to the standard view.

As you can see in **Figure 3**, I made a copy of _Layout.cshtml and renamed it _Layout.mobile.cshtml per this convention. I highlighted the line of HTML that was added to make it obvious which _Layout.cshtml is being used to render the page. When the site is rendered in the desktop browser nothing changes, but when I render the site in the mobile browser, as shown in **Figure 4**, you can see that the mobile version of _Layout.cshtml is being used.

Serving up Browser-Specific Views For the most part, it's no longer necessary to serve up browser-specific views or content on the desktop. If you've been developing Web sites for any length of time, however, chances are you wrote code or CSS to get something to work in one specific browser or another. That's where we are now with mobile browsers, but the problem is compounded because of the vast number of mobile platforms, each with its own browser. And, as if that weren't enough, we now have the concept of "native" sites. It's no longer sufficient that your site renders well on mobile browsers; to be truly top-notch, the site needs to have the look and feel of applications that run natively on the device. This requires specific views to be served not only for mobile browsers in general, but for specific mobile browsers with styling to match the platform.

When you're developing a multitargeted site, you generally want mobile users to see a view that's different from the one your desktop users see.

To accomplish this, ASP.NET MVC 4 introduced Display Modes. This new feature allows you to combine the ease of convention over configuration with the robustness of using browser sniffing to serve up browser-specific views.

In order to take advantage of Display Modes you must first define them in the Application_Start method of the Global.asax, like so:

```
DisplayModeProvider.Instance.Modes.Insert(0, new DefaultDisplayMode("iPhone")
{
  ContextCondition = (context => context.GetOverriddenUserAgent().IndexOf
    ("iPhone", StringComparison.OrdinalIgnoreCase) >= 0)
});
```

This is a Display Mode created for the iPhone browser. The first instance of "iPhone" defines the suffix of the view that ASP.NET MVC will look for when determining what view to render. The second instance of "iPhone" refers to the user agent making the request and defines the condition ASP.NET MVC will use to match the naming convention [*view*].iPhone.cshtml. You can essentially translate this as: When an iPhone browser is making the request, look first for views matching the suffix "iPhone."

To demonstrate the iPhone browser Display Mode, I made another copy of _Layout.cshtml and named it _Layout.iPhone.cshtml per the naming convention defined when I created the Display Mode. I then modified it to make it obvious that the iPhone layout is being used when I browse the site with the iPhone. If I look at the site in the desktop browser or the Windows Phone emulator browser, I don't see my modification, but when I pull up the site in the iPhone browser, as shown in **Figure 5**, I can see the modification.

The addition of these features to ASP.NET MVC 4 allows you to easily serve up mobile- and browser-specific views with little to no plumbing. Using view overriding and display modes in ASP.NET MVC 4, you can serve up mobile- and browser-specific views, giving you the opportunity to fully tailor your site to the device on which it's being rendered.

jQuery Mobile and jQuery.Mobile.MVC

jQuery Mobile is an open source library for building UIs for mobile devices based on jQuery Core. Because jQuery Mobile is a well-documented toolset and because implementing jQuery Mobile in ASP.NET MVC 4 is no different than implementing it in any other language or framework, I'm not going to go into it here, except to note how to incorporate it into ASP.NET MVC 4.

jQuery Mobile isn't included by default with the ASP.NET MVC 4 project templates (other than the Mobile Application project template), but adding it to an ASP.NET MVC 4 application isn't difficult. You should use NuGet to install the scripts and other necessary files and then go into _Layout.cshtml and manually add the required script and CSS references. Alternatively, you can install the jQuery.Mobile.MVC NuGet package, which will install all the scripts and other necessary files; create a _Layout.Mobile.cshtml; and add references to all the jQuery Mobile script and CSS files. The jQuery.Mobile.MVC NuGet package also adds view-switching functionality that lets users viewing the mobile version of the site switch to the full desktop view, as shown in **Figure 6**.

Project Template

If you want to create a standalone site that specifically targets mobile browsers, ASP.NET MVC 4 has created a project template that lets you to do exactly that. When you create a new ASP.NET MVC 4 project, you now have the option to choose a "Mobile Application" project template.

Create a project with the ASP.NET MVC 4 Mobile Application project template and then take a look at the overall project structure. You'll see that nothing obvious has changed as compared with the standard ASP.NET MVC 4 application project template—you still have the same models, views and controllers. When you run the application, however, you'll see that the site is specifically tailored for mobile browsers.

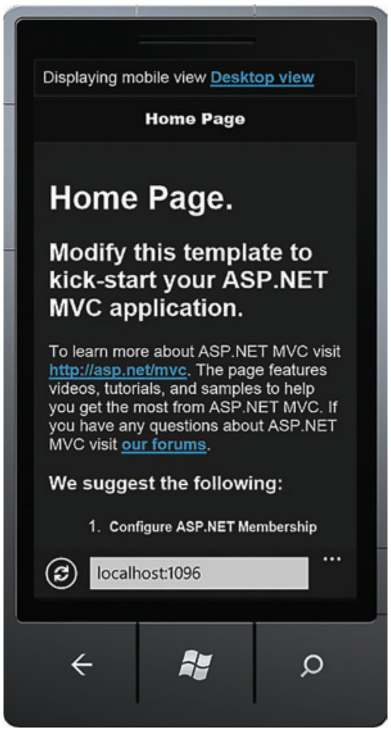
As I briefly mentioned earlier, the ASP.NET MVC 4 Mobile project template includes jQuery Mobile out of the box. Moreover, it implements jQuery Mobile in all the default views:

```
<h2>@ViewBag.Message</h2>
<p>
  To learn more about ASP.NET MVC visit <a href="http://asp.net/mvc"
    title="ASP.NET MVC Website">http://asp.net/mvc</a>.
</p>

<ul data-role="listview" data-inset="true">
  <li data-role="list-divider">Navigation</li>
  <li>@Html.ActionLink("About", "About", "Home")</li>
  <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
```

This code is from Views/Home/Index.cshtml. If you aren't familiar with jQuery Mobile and how it's implemented, the first things you probably noticed are the "data-" ("data dash") attributes. These are what jQuery Mobile uses to configure controls on the page.

The ASP.NET MVC 4 Mobile Application project template is great for when you're creating a site that targets only mobile browsers, or to serve as a reference when you want to implement mobile browser-specific features. Most of the time, you'll probably be creating a site that's really aimed at desktop browsers. Still, you want your site to render in a usable fashion on mobile browsers without having to spend a ton of extra time and effort making mobile-specific code modifications. Fortunately, the ASP.NET MVC 4 Mobile Application project template can be used here and as a starting point for introducing mobile-specific features into an existing desktop browser-based MVC application.

With the popularity of mobile devices, it's no surprise that improving the mobile Web site development experience was such a focus in ASP.NET MVC 4. Utilizing the new features in ASP.NET MVC 4 will ensure that sites not only reach their full audience, but do so without requiring major coding and duplication in the UI layer. ■



Displaying mobile view Desktop view

Home Page

Home Page.

Modify this template to kick-start your ASP.NET MVC application.

To learn more about ASP.NET MVC visit http://asp.net/mvc. The page features videos, tutorials, and samples to help you get the most from ASP.NET MVC. If you have any questions about ASP.NET MVC visit our forums.

We suggest the following:

1. Configure ASP.NET Membership

localhost:1096

Figure 6 jQuery Mobile View with View-Switching Functionality

Keith Burnell *is a senior software engineer with Skyline Technologies. He's been developing software for more than 10 years, specializing in large-scale ASP.NET and ASP.NET MVC Web site development. Burnell is an active member of the developer community and can be found on his blog (dotnetdevdude.com) and on Twitter at twitter.com/keburnell.*

Thanks to the following technical experts for reviewing this article: *John Ptacek and Clark Sell*

Introducing the Navigation for ASP.NET Web Forms Framework

Graham Mendick

The Navigation for ASP.NET Web Forms framework, an open source project hosted at navigation.codeplex.com, lets you write Web Forms code with unit test coverage and adherence to don't repeat yourself (DRY) principles that would make an ASP.NET MVC application green with envy.

Although there has been some abandonment of Web Forms in favor of MVC, with some developers growing tired of large codebehinds unreachable by unit tests, this new framework—in conjunction with data binding—makes a compelling argument for taking a fresh look at Web Forms.

Data binding with ObjectDataSource controls has been around since Visual Studio 2005, allowing for cleaner codebehinds and data-retrieval code to be unit tested, but there have been some problems inhibiting its uptake—for example, raising an exception was the only way of reporting a business validation failure back to the UI.

This article discusses:

- Setting up and configuring a sample application
- Navigation basics
- Passing data in Web Forms

Technologies discussed:

ASP.NET Web Forms

Code download available at:

code.msdn.microsoft.com/mag201205WebForms
navigation.codeplex.com

The vast majority of the Web Forms development effort for the upcoming release of Visual Studio has been invested in data binding, bringing across model binding concepts from MVC to resolve these issues—for example, the introduction of model state addresses the business validation failure communication issue. However, two thorns remain in the side of data binding related to navigation and data passing—but they can be painlessly extracted using the Navigation for ASP.NET Web Forms framework (which I'll call the "Navigation framework" hereafter for brevity).

The first thorn is that there's no abstraction for navigation logic, unlike in MVC where it's encapsulated in the controller method return types. This results in redirect calls inside data-bound methods, preventing them from being unit tested. The second thorn is that the type of a parameter of an ObjectDataSource determines where its value comes from—for example, a QueryStringParameter always gets its data from the query string. This prevents the same data source from being used in different navigational contexts—such as postback and non-postback—without substantial logic in the dreaded codebehind.

The Navigation framework removes these thorns by taking a holistic approach to navigation and data passing. Regardless of the type of navigation being performed—be it hyperlink, postback, AJAX history or unit test—the data being passed is always held in the same way. In future articles, I'll show how this leads to empty codebehinds with fully unit-tested data retrieval and navigation logic and also to Search Engine Optimization (SEO)-friendly, progressively enhanced single-page applications with no code duplication for JavaScript-enabled and -disabled scenarios. This article introduces the Navigation framework and demonstrates some of its basic—but key—concepts by building a sample Web application.

Sample Application

The sample Web application is an online survey. This survey has only two questions and displays a "thank you" message upon completion. Each question is represented by a separate ASPX page called Question1.aspx and Question2.aspx respectively, and the "thank you" message has its own page called Thanks.aspx.

The first question asked is, "Which ASP.NET technology are you currently using?" to which the possible answers are either "Web Forms" or "MVC." So to Question1.aspx I'll add the question and the hardcoded radio button answers:

```
<h1>Question 1</h1>
<h2>Which ASP.NET technology are you currently using?</h2>

<asp:RadioButtonList ID="Answer" runat="server">
  <asp:ListItem Text="Web Forms" Selected="True" />
  <asp:ListItem Text="MVC" />
</asp:RadioButtonList>
```

The second question, "Are you using the Navigation for ASP.NET Web Forms framework?" has answers of "Yes" or "No" and is marked up in a similar fashion.

Getting Started

The most straightforward way to set up the survey Web project to use the Navigation framework is to install it using the NuGet Package Manager. Running the command "Install-Package Navigation" from within the Package Manager Console will add the required reference and configuration. If you're not using Visual Studio 2010, manual setup instructions can be found at navigation.codeplex.com/documentation.

Navigation Configuration

The Navigation framework can be thought of as a state machine, where each different state represents a page and moving from one state to another—or navigating between pages—is termed a transition. This predefined set of states and transitions is configured in the StateInfo.config file created by the NuGet installation. Without this underpinning configuration, running the survey application will throw an exception.

Because states are essentially just pages, the survey application requires three states, one for each of its three pages:

```
<state key="Question1" page="~/Question1.aspx">
</state>
<state key="Question2" page="~/Question2.aspx">
</state>
<state key="Thanks" page="~/Thanks.aspx">
</state>
```

From now on I'll refer to the different states by their key names, Question1, Question2 and Thanks, rather than by the pages they represent.

Because transitions describe the possible navigations between states, the survey application requires two transitions. One is for the navigation from Question1 to Question2 and another is for the navigation from Question2 to Thanks. A transition appears as a child of the state being exited and points, via its "to" attribute, at the state being entered:

```
<state key="Question1" page="~/Question1.aspx">
  <transition key="Next" to="Question2"/>
</state>
<state key="Question2" page="~/Question2.aspx">
  <transition key="Next" to="Thanks"/>
</state>
<state key="Thanks" page="~/Thanks.aspx">
</state>
```

Dialogs are the final element of the configuration and represent a logical grouping of states. The survey application only requires one dialog because Question1, Question2 and Thanks are effectively a single navigation path. The dialog's "initial" attribute must point to the starting state—that is, Question1:

```
<dialog key="Survey" initial="Question1" path="~/Question1.aspx">
  <state key="Question1" page="~/Question1.aspx">
    <transition key="Next" to="Question2"/>
  </state>
  <state key="Question2" page="~/Question2.aspx">
    <transition key="Next" to="Thanks"/>
  </state>
  <state key="Thanks" page="~/Thanks.aspx">
  </state>
</dialog>
```

You'll notice that each dialog, state and transition has a key attribute. I chose to name the state keys after the page names, but this isn't necessary. Note, though, that all keys must be unique within their parent; for example, you can't have sibling states with the same key.

With Question1.aspx as the start page, the survey application now starts successfully in the Question1 state. However, the survey remains stuck in this state because there's no way to progress to Question2.

Navigation

It's useful to split the different sorts of Web Forms navigation into two camps. The non-postback camp is where control is passed from one ASPX page to another and takes the form of hyperlinks, redirects or transfers. The postback camp is where control remains on the same page and takes the form of postbacks, partial page requests or AJAX history. This second kind will be examined in a future article discussing the single-page interface pattern. In this article, I'll focus on the first type of navigation.

To move between pages, a URL must be built. Prior to Visual Studio 2008, the only option was to manually construct URLs from hardcoded ASPX page names, causing tight coupling between pages that made applications brittle and hard to maintain. The introduction of routing alleviated this problem, with configurable route names used in place of page names. However, the fact that routing throws an exception if used outside a Web environment—combined with routing's resistance to mocking—makes it an enemy of unit testing.

The Navigation framework retains the loose coupling provided by routing and is a friend of unit testing. Similar to the use of route names, instead of hardcoding ASPX page names, it's the dialog and transition keys configured in the preceding section that are referenced in code; the state navigated to is determined by the respective "initial" and "to" attributes.

Returning to the survey, the Next transition key can be used to move from Question1 to Question2. I'll add a Next button to Question1.aspx and the following code to its associated click handler:

```
protected void Next_Click(object sender, EventArgs e)
{
  StateController.Navigate("Next");
}
```

The key passed in to the Navigate method is matched against the configured child transitions of the Question1 state and then the state identified by the "to" attribute is displayed—that is, Question2. I'll add the same button and handler to Question2.aspx. If you run the survey, you'll find you can navigate through the three states by clicking the Next buttons.

You might have noticed the second question is Web Forms-specific and, as such, is irrelevant when "MVC" is selected as the first answer. The code needs changing to handle this scenario, navigating directly from Question1 to Thanks and bypassing Question2 entirely.

The current configuration doesn't allow navigation from Question1 to Thanks because the only transition listed is to Question2. So I'll change the configuration by adding a second transition below the Question1 state:

```
<state key="Question1" page="~/Question1.aspx">
  <transition key="Next" to="Question2"/>
  <transition key="Next_MVC" to="Thanks"/>
</state>
```

With this new transition in place it's simple to adjust the Next button click handler to pass a different transition key depending on the answer chosen:

```
if (Answer.SelectedValue != "MVC")
{
  StateController.Navigate("Next");
}
else
{
  StateController.Navigate("Next_MVC");
}
```

A survey wouldn't be much good if it didn't allow the user to change answers. Currently there's no way to return to a previous question (aside from the browser back button). To navigate back you might think you need to add two transitions below Thanks, pointing to Question1 and Question2, and another below Question2, pointing to Question1. Although this would work, it's unnecessary because back navigation comes free with the Navigation framework.

Breadcrumb navigation is a set of links providing access to each previous page the user visited in reaching the current one. Web Forms has breadcrumb navigation built into its site map functionality. However, because site maps are represented by a fixed navigational structure, for a given page these breadcrumbs are always the same regardless of the route taken. They can't handle situations like that found in the survey where the route to Thanks sometimes excludes Question2. The Navigation framework, by keeping track of states visited as navigations occur, builds up a breadcrumb trail of the actual route taken.

To demonstrate, I'll add a hyperlink to Question2.aspx and in the codebehind programmatically set its NavigateUrl property using back navigation. A distance parameter must be passed indicating how many states to go back to, a value of 1 meaning the immediate predecessor:

```
protected void Page_Load(object sender, EventArgs e)
{
  Question1.NavigateUrl = StateController.GetNavigationBackLink(1);
}
```

If you run the app and answer "Web Forms" to question one you'll see the hyperlink on Question2.aspx takes you back to the first question.

I'll do the same for Thanks.aspx, although it's a bit trickier because two hyperlinks are needed (one for each question), and the user may not have seen both questions—that is, if he answered "MVC" to the first. The number of precedent states can be checked before deciding how to set up the hyperlinks (see **Figure 1**).

The survey is now functional, allowing you to fill in questions and amend previous answers. But there's little point to a survey if these answers aren't put to use. I'll show how they can be passed from Question1 and Question2 to Thanks, where they'll be displayed in summary form.

Data Passing

There are as many different ways of passing data in Web Forms as there are ways to navigate. With non-postback navigation, where control is passed from one page to another (via hyperlink, redirect or transfer), query string or route data can be used. With postback navigation, where control remains on the same page (via postback, partial page request or AJAX history), control values, view state or event arguments are the likely candidates.

Prior to Visual Studio 2005, codebehinds were burdened with handling this passed-in data, so they swelled with value-extraction and type-conversion logic. Their load was considerably lightened with the introduction of data source controls and select parameters ("value providers" in the next version of Visual Studio). However, these select parameters are tied to a specific data source and they can't dynamically switch sources depending on the navigational context. For example, they can't retrieve their values alternatively from a control or from the query string depending on whether it's a postback or not. Working around these limitations causes code to leak back into the codebehind, reverting back to square one of bloated and untestable codebehinds.

The Navigation framework avoids such problems by providing a single data source regardless of the navigation involved, called state data. The first time a page is loaded, the state data is populated with any data passed during the navigation, in a fashion similar to query string or route data. A marked difference, however, is that state data isn't read-only, so as subsequent postback navigations occur, it can be updated to reflect the page's current incarnation. This will prove beneficial when I revisit back navigation toward the end of this section.

I'll change the survey so the answer to the first question is passed to the Thanks state where it will be redisplayed to the user. Data is passed while navigating via a collection of key-value pairs, called NavigationData. I'll change the Next click handler of Question1.aspx so the answer to the first question is passed to the next state:

```
NavigationData data = new NavigationData();
data["technology"] = Answer.SelectedValue;
if (Answer.SelectedValue != "MVC")
{
  StateController.Navigate("Next", data);
}
else
{
  StateController.Navigate("Next_MVC", data);
}
```

This NavigationData passed during navigation is used to initialize the state data that's made available to the next state via the Data property on the StateContext object. I'll add a Label to Thanks.aspx and set its Text property to display the answer passed in:

```
Summary.Text = (string) StateContext.Data["technology"];
```

If you run the survey, you'll notice this summary information is only displayed when the answer to the first question is "MVC"; an answer of "Web Forms" is never shown. This is because NavigationData is only available to the next state, but not to any states reached as a result of subsequent navigation. So an answer of "Web Forms"

/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 ...

.NET | BizTalk Server | SQL Server | Mobile | activex | SharePoint 2010 | DELPHI | 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

Figure 1 **Dynamic Back Navigation**

```
protected void Page_Load(object sender, EventArgs e)
{
  if (StateController.CanNavigateBack(2))
  {
    Question1.NavigateUrl = StateController.GetNavigationBackLink(2);
    Question2.NavigateUrl = StateController.GetNavigationBackLink(1);
  }
  else
  {
    Question1.NavigateUrl = StateController.GetNavigationBackLink(1);
    Question2.Visible = false;
  }
}
```

is present in the Question2 state data, but isn't available by the time Thanks is reached. One way to solve this is to change Question2.aspx so that it relays the answer to the first question—that is, it takes the answer out of its state data and passes it to Thanks when it navigates:

```
NavigationData data = new NavigationData();
data["technology"] = StateContext.Data["technology"];
StateController.Navigate("Next", data);
```

This approach isn't ideal because it couples Question1 and Question2 together, forcing the latter state to be aware of the data being passed in by the former. For example, a new question can't be inserted between the first and second without a corresponding change to Question2.aspx. A future-proof implementation involves creating a new NavigationData containing all of the Question2 state data; this is achieved by passing true to the NavigationData constructor:

```
NavigationData data = new NavigationData(true);
StateController.Navigate("Next", data);
```

Another key difference between state data and query string or route data is that with state data you're not restricted to passing strings. Rather than pass the answer as a string, as was done for Question1, for Question2 I'll pass a bool to Thanks, with a value of true corresponding to "Yes":

```
NavigationData data = new NavigationData(true);
data["navigation"] = Answer.SelectedValue == "Yes" ? true : false;
StateController.Navigate("Next", data);
```

You can see its data type is preserved when it's retrieved from the Thanks state data:

```
Summary.Text = (string) StateContext.Data["technology"];
if (StateContext.Data["navigation"] != null)
{
  Summary.Text += ", " + (bool) StateContext.Data["navigation"];
}
```

The survey is complete, except for one problem: Answers to questions aren't retained when using the back navigation hyperlinks. For example, when returning to Question1 from Thanks the context is lost, so the default "Web Forms" radio button is always selected regardless of the answer given.

In the previous section you saw the benefit of back navigation over static site map breadcrumbs. Another limitation of the breadcrumbs generated by the site map is that they don't carry any data. This means following them can lose contextual information. For example, they can't pass the "MVC" answer previously selected when they return to Question1 from Thanks. The Navigation framework, by keeping track of the state data associated with the states visited as navigations occur, builds up a context-sensitive breadcrumb trail. During a back navigation this state data is restored, allowing the page to be recreated exactly as before.

Armed with context-sensitive back navigation, I can change the survey so that answers are retained when revisiting states. The first stage is to set the answers into state data in the Next click handlers, prior to navigating away:

```
StateContext.Data["answer"] = Answer.SelectedValue;
```

Now, when Question1 or Question2 are revisited, the state data will contain the answer previously selected. It's then a simple matter to retrieve this answer in the Page_Load method and preselect the relevant radio button:

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!Page.IsPostBack)
  {
    if (StateContext.Data["answer"] != null)
    {
      Answer.SelectedValue = (string)StateContext.Data["answer"];
    }
  }
}
```

The survey, now complete, isn't susceptible to the errors commonly encountered in Web applications when users press the browser back button (or have multiple browser windows open). Such problems typically arise when page-specific data is persisted in a server-side session. Although there's only one session object, there can be multiple "current" versions of a single page. For example, using the back button to retrieve a "stale" version of a page from the browser cache might cause the client and server to be out of sync. The Navigation framework faces no such issues because it doesn't have any server-side cache. Instead, the state, state data and breadcrumb trail are all held in the URL. This does mean, however, that a user can change these values by editing the URL.

Making MVC Jealous

Earlier I stated that the Navigation framework lets you create Web Forms code to make MVC jealous. After such a bold claim you might be feeling a bit short-changed by the survey sample application, because it probably has MVC holding its nose to avoid the unsavory whiff of codebehind about it. But please don't despair; this was merely an introduction to the core concepts. Future articles will focus on architectural integrity, with particular attention paid to unit testing and DRY principles.

In the second installment I'll build a data-bound sample with empty codebehinds and complete unit test code coverage. This coverage will even include the navigational code, notoriously difficult to test in an MVC application.

In the third installment I'll build an SEO-friendly single-page application. It'll use progressive enhancement, employing ASP.NET AJAX when JavaScript is enabled and degrading gracefully when it's disabled, with the same data-bound methods used in both scenarios. Again, this is tricky to achieve in an MVC application.

If this whets your appetite and you can't wait to try out some of the more advanced functionality, be sure to download the comprehensive feature documentation and sample code from navigation.codeplex.com. ■

GRAHAM MENDICK *is Web Forms' biggest fan and wants to show it can be just as architecturally sound as ASP.NET MVC. He authored the Navigation for ASP.NET Web Forms framework, which he believes—when used with data binding—can breathe new life into Web Forms.*

THANKS to the following technical expert for reviewing this article: *Damian Edwards*

MANAGED TEAM FOUNDATION SERVER HOSTING

DiscountASP.NET is here to help development teams from all across the globe with their software development projects. With our NEW MANAGED TFS HOSTING solution, development teams no longer have to worry about the hassle of managing their own TFS server. We can do it for you! Plus, we provide all the tools and features you need to so you can effectively manage your software development projects:

- Your Own Private Instance of TFS
- 30gb of TFS disk space
- Source Code Management
- Bug/Issue Tracking
- Sharepoint
- Reporting Services
- Unlimited Projects
- TFS Build Server*
- NO LONG TERM CONTRACTS!

discount ASP.net

Team Foundation Server Hosting

Microsoft Partner
Gold Hosting

For More Info, visit
www.DiscountASP.NET/TFS/MSDN

*TFS Build Server available as an add-on

Managing Complexity in T4 Code-Generation Solutions

Peter Vogel

In my article, "Lowering the Barriers to Code Generation with T4" in the April issue of *MSDN Magazine* (msdn.microsoft.com/magazine/hh882448), I described how the Microsoft Text Template Transformation Toolkit (T4) makes it much easier for developers to create code-generation solutions (and how they can start recognizing code-generation opportunities). However, as with any programming environment, T4 solutions can grow into complex, monolithic solutions that can't be maintained or extended. Avoiding that fate requires recognizing the various ways that code in T4 solutions can be refactored and integrated into code-generation solutions. And that requires some understanding of the T4 code-generation process.

T4 Code-Generation Process

The heart of the T4 code-generation process is the T4 engine, which accepts a T4 template consisting of boilerplate code, control blocks, class features and directives. From these inputs, the engine creates a temporary class (the "generated transformation class") that inherits from the Microsoft TextTransformation class. An application domain is then created and, in that domain, the generated transformation class is compiled and executed to produce the output, which might be anything from an HTML page to a C# program.

The boilerplate text and the control blocks in the template are incorporated into a single method (called TransformText) in the generated transformation class. However, the code in *class features* (enclosed in <#+ … #> delimiters) are not placed in that method—class feature code is added to the generated transformation class outside of any methods the T4 process creates. In my previous article, for instance, I used a class feature block to add an ArrayList—declared outside of any method—to the generated transformation class. I then accessed that ArrayList in my code generation's control blocks as part of the process of generating the code. In addition to adding fields, like the ArrayList, class features can also be used to add private methods that can be called from your code blocks.

The engine is the heart of the process, but two other components also participate in the T4 code-generation process: the directive processor and the host. Directives provide a parameter-based way to add code or otherwise control the process. For example, the T4 Import directive has a Namespace parameter for creating a using or Imports statement in the generated transformation class; the Include directive has a file parameter for retrieving text from another file and adding it to the generated transformation class. The default directive processor handles the directives that come with T4.

The T4 code-generation process also needs a host to integrate the process with the environment the engine is executing in. The host, for instance, provides a standard set of assemblies and namespaces to the engine so that not all the assemblies the generated transformation class code needs have to be specified in the template. When requested by the engine or the directive processor, the host adds references to assemblies; retrieves (and sometimes reads) files for the engine; and can even retrieve custom directive processors or provide default values for directives whose parameters have been omitted. The host also provides the AppDomain the generated transformation class executes in and displays any error and warning messages generated by the engine. Visual Studio can host the T4 engine (through the custom tool TextTemplatingFileGenerator), as can the TextTransform command-line utility that processes T4 templates outside of Visual Studio.

This article discusses:

- The T4 code-generation process
- Mechanisms for refactoring code
- Using T4 at run time
- Warnings and errors

Technologies discussed:

The Microsoft .NET Framework, Text Template Transformation Toolkit (T4), Visual Studio 2010

Figure 1 **A Sample T4 Template**

```
<#@ template language="VB" #>

public partial class ConnectionManager
{
<#
  For Each conName As String in Connections
#>
  private void <#= conName #>(){}
<#
  Next
#>
<#+
  Private Function GetFormattedDate() As String
    Return DateTime.Now.ToShortDateString()
  End Function
#>
```

As an example of this process, take look at the T4 template with a combination of static code, control blocks and a class feature shown in **Figure 1**.

The generated transformation class would look something like what's shown in **Figure 2**.

Given this description of the T4 code-generation process, you can refactor potentially monolithic solutions into more maintainable components using any of these three options:

1. Class features
2. Extending the base TextTransformation class
3. Custom directive processors

These mechanisms also allow you to reuse code across multiple code-generation solutions. To demonstrate these options I'll use a trivially simple case study: adding a copyright notice to the generated code. The "meta-ness" of writing code to generate code creates enough complexity without picking a complicated case study—you can do that on your own.

There is, at least, one other option I'm not going to discuss: developing your own host so you can invoke host-specific features from your T4 template. Creating a new host is really necessary only if you intend to invoke T4 processing from outside Visual Studio and don't want to use the TextTransform command-line tool.

Class Features

Class features provide the easiest way of reducing complexity in your T4 process and reusing code. Class features allow you to encapsulate parts of your code-generation process into methods that you can call from the TransformText method that forms the "mainline" of your code. You can also leverage the Include directive to reuse class features across multiple solutions.

Figure 2 **The Generated Transformation Class**

```
Public Class GeneratedTextTransformation
  Inherits Microsoft.VisualStudio.TextTemplating.TextTransformation

  Public Overrides Function TransformText() As String
    Me.Write("public partial class ConnectionManager{")
    For Each conName As String in Connections
      Me.Write("private void ")
      Me.Write(Me.ToStringHelper.ToStringWithCulture(conName))
      Me.Write("(){}")
    Next
  End Function

  Private Function GetFormattedDate() As String
    Return DateTime.Now.ToShortDateString()
  End Fuction
End Class
```

The first step in using a class feature is to add a T4 file to your project containing the class features you want to reuse in multiple code-generation solutions, enclosed in the T4 <#+ … #> delimiters. This can include methods, properties and fields. Your template files can also contain T4-specific features such as directives. Because your file is still a T4 file, it will generate code that will be compiled into your application, so you should suppress code generation for the file. The easiest way to do this is to clear the Custom Tool property of your class feature files. This example defines a method called ReturnCopyright as a class feature, written in Visual Basic:

```
<#+
 Public Function ReturnCopyright() As String
   Return "Copyright by PH&V Information Services, 2012"
 End Function
#>
```

Once you've defined a class feature, you can add it to a template using the Include directive and use the feature from a control block in a T4 template. The next example, which assumes that the previous class feature was defined in a file called CopyrightFeature.tt, uses the ReturnCopyright method as an expression:

```
<#@ Template language="VB"   #>
<#@ Output extension=".generated.cs" #>

<#= ReturnCopyright() #>

<#@ Include file="CopyrightFeature.tt" #>
```

Alternatively, you can simply use the T4 normal boilerplate syntax to generate similar code:

```
<#+
  Public Function ReturnCopyright() As String
#>
  Copyright by PH&V Information Services, 2012
<#+
  End Function
#>
```

You can also use the Write and WriteLine methods within your class feature to generate code, as this class feature does:

```
<#+
  Public Sub WriteCopyright()
    Write("Copyright by PH&V Information Services, 2012")
  End Function
#>
```

The latter two approaches would use the following code to call the method once an Include directive added it to the template:

```
<# WriteCopyright() #>
```

You can parameterize class features using normal function arguments and daisy-chain features together by using the Include directive in T4 files that are themselves included in other T4 files to build up a well-structured, reusable library.

Compared to the other solutions, using class features has at least one benefit and one drawback. You experience the benefit as you develop your code-generation solution: class features (and Includes in general) don't involve compiled assemblies. For performance reasons, the T4 engine may lock the assemblies it uses when it loads the generated transformation class into its application domain. This means that as you test and modify compiled code, you might find that you can't replace the relevant assemblies without shutting down and restarting Visual Studio. This won't happen with class features. Note that this has been addressed in Visual Studio 2010 SP1, which no longer locks assemblies.

Developers might encounter the drawback, however, when they use your code-generation solution: They must add not only your T4 template to their project, but also all your supporting T4 Include files. This is a scenario where you might consider creating a Visual Studio template containing all of the T4 files a developer needs for your code-generation solution so they can be added as a group. It would also make sense to segregate your Include files into a folder within the solution. The following example uses the Include directive to add a file containing class features from a folder called Templates:

```
<#@ Include file="Templates\classfeatures.tt" #>
```

You are not, of course, limited to just using class features in Include files—you can include any arbitrary set of control blocks and text that you want to be incorporated into the generated transformation class TransformText method. However, like avoiding GoTos, using well-defined members in Include files helps to manage the conceptual complexity of your solution.

Extending the TextTransformation Class

Replacing the TextTransformation class with a class of your own allows you to incorporate custom functionality into methods in that class that can be called in the generated transformation class. Extending the TextTransformation class is a good choice when you have code that will be used in many (or all) of your code-generation solutions: essentially, you factor that code out of your solutions and into the T4 engine.

The first step in extending the TextTransformation class is to create an abstract class that inherits from the class:

```
public abstract class PhvisT4Base:
  Microsoft.VisualStudio.TextTemplating.TextTransformation
{
}
```

If you don't have the Microsoft.VisualStudio.TextTemplating DLL that contains the TextTransformation class, download the SDK for your version of Visual Studio before adding the reference.

Depending on your version of T4, you might have to provide an implementation of the TransformText method as part of inheriting from TextTransformation. If so, your method must return the string containing the output of the generated transformation class, which is held in the TextTransformation class GenerationEnvironment property. Your override of the TransformText method, if required, should look like this:

```
public override string TransformText()
{
  return this.GenerationEnvironment.ToString();
}
```

As with class features, you have two options for adding code to the generated transformation class. You can create methods that return string values, as this example does:

```
protected string Copyright()
{
  return @"Copyright PH&V Information Services, 2012";
}
```

This method can be used either in an expression or with the T4 Write or WriteLine methods, as in these examples:

```
<#= CopyRight() #>
<#  WriteLine(CopyRight()); #>
```

Alternatively, you can use the TextTransformation base class Write or WriteLine methods directly in the methods you add to the TextTransformation base class, as this example does:

```
protected void Copyright()
{
  base.Write(@"Copyright PH&V Information Services, 2012");
}
```

This method can then be called from a control block, like this:

```
<# CopyRight(); #>
```

The final step in replacing the default TextTransformation class is to specify in your T4 template that the generated transformation class is to inherit from your new class. You do that with the Inherits parameter of the Template attribute:

```
<#@ Template language="C#" inherits="PhvT4Utils.PhvisT4Base" #>
```

You must also add to your T4 template an assembly directive that references the DLL containing your base class, using the full physical path name to the DLL:

```
<#@ Assembly name="C:\T4Support\PhvT4Utils.dll" #>
```

Alternatively, you can add your base class to the global assembly cache (GAC).

In Visual Studio 2010, you can use environment and macro variables to simplify the path to the DLL. For instance, during development, you could use the $(ProjectDir) macro to reference the DLL containing your base class:

```
<#@ Assembly name="$(ProjectDir)\bin\PhvT4Utils.dll" #>
```

At run time, assuming you install your class files to the Program Files folder, you can use the environment variable %programfiles% on

Warnings and Errors

Any robust process provides feedback on its progress. In a class feature or when extending the TextTransformation class, you can report on problems in the execution of the generated transformation class by adding calls to the TextTransformation class Error and Warning methods. Both methods accept a single string input and pass the string to the Microsoft Text Template Transformation Toolkit (T4) host (the host is responsible for deciding how to display the messages). In Visual Studio, the messages appear in the Error List window.

This example reports on an error in a class feature:

```
<#= GetDatabaseData("") #>

<#+
  private string GetDatabaseData(string ConnectionString)
  {
    if (ConnectionString == "")
    {
      base.Error("No connection string provided.");
    }
    return "";
  }
...
```

In a class feature, Error and Warning can't be used to report on problems in the T4 process, only on errors that occur when the generated transformation class executes. To put it another way, in a class feature, Error and Warning messages will appear only if the generated transformation class can be correctly assembled from your T4 file, compiled and executed.

If you're creating a custom directive processor, you can still integrate with the T4 error reporting process by adding CompilerError objects to the class Errors property. The CompilerError object supports passing multiple pieces of information about the error (including line number and file name), but this example simply sets the ErrorText property:

```
System.CodeDom.Compiler.CompilerError err =
  new System.CodeDom.Compiler.CompilerError();
err.ErrorText = "Missing directive parameter";
this.Errors.Add(err);
```

—P.V.

32-bit versions of Windows or, on 64-bit versions, %ProgramW6432% for the Program Files folder, or %ProgramFiles(x86)% for the Program Files (x86) folder. This example assumes that the DLL has been placed in C:\Program Files\PHVIS\T4Tools on a 64-bit version of Windows:

```
<#@ Assembly name="%ProgramW6432%\PHVIS\T4Tools\PHVT4Utils.DLL" #>
```

As with other solutions with compiled code, Visual Studio might lock the DLL containing your code during execution of the generated transformation class. If this happens while you're developing your TextTransformation class, you'll need to restart Visual Studio—and recompile your code—before you can make more changes.

Custom Directive Processors

The most powerful and flexible way to manage complexity in the code-generation process is to use custom directive processors. Among other features, directive processors can add references to the generated transformation class and insert code into methods that execute before and after the generated transformation class TransformText method. Directives are also easy for developers to use: they just have to provide values for the directive's parameters and then use the members the directives make available.

The key issue with directive processors is that you must add a key in the Windows registry in order for a developer to use your processor. Here, again, it's worthwhile to consider packaging up your T4 solution so that the registry entry can be made automatically. On 32-bit versions of Windows, the key must be:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\visualstudioversion\
TextTemplating\DirectiveProcessors
```

For 64-bit versions of Windows, the key is:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\VisualStudio\10.0\
TextTemplating\DirectiveProcessors
```

The name of your key is the name of your directive processor class, which, following Microsoft naming convention, should end with "DirectiveProcessor." You'll need the following subkeys:

- **Default:** Empty or a description of your directive.
- **Class:** The full name of your class in the format *Namespace.ClassName*.
- **Assembly/CodeBase:** The name of your DLL if you've placed your directive's DLL in the GAC (Assembly) or the full path to your directive's DLL (CodeBase).

When developing your processor, if you don't get this entry right, Visual Studio will generate an error that it was unable to find your directive processor or resolve its type. After you fix any errors in your registry entries, you might need to restart Visual Studio to pick up your changes.

To use your directive processor, a developer adds a directive with any name to a T4 template and ties the directive to your processor through the directive's Processor parameter (which references the Windows registry key). When the T4 template executes, your directive processor is passed the name of the directive that the developer used along with any parameters the developer specified.

This example ties a directive named Copyright to a processor called CopyrightDirectiveProcessor and includes a parameter called Year:

```
<#@ Copyright Processor="CopyrightDirectiveProcessor" Year="2012" #>
```

As with a class feature, the output from a directive processor is added to the generated transformation class outside of the

TransformText method. As a result, you'll use your processor to add new members to the generated transformation class that developers can use in their template control blocks. The previous sample directive might have added a property or a string variable that a developer could use in an expression, like this:

```
<#= Copyright #>
```

Of course, the next step is to create a directive process to process this directive.

Creating a Custom Directive Processor

T4 directive processors inherit from the class Microsoft.VisualStudio.TextTemplating.DirectiveProcessor (download the Visual Studio SDK to get the TextTemplating library). Your directive processor must, from its GetClassCodeForProcessingRun method, return the code that will be added to the generated transformation class. However, before calling the GetClassCodeForProcessingRun method, the T4 engine will call the processor's IsDirectiveSupported method (passing the name of your directive) and the ProcessDirective method (passing the name of the directive and the values of its parameters). From the IsDirectiveSupported method, you should return false if your directive shouldn't be executed, and true otherwise.

Because the ProcessDirective method is passed all the information about the directive, that's where you usually build the code that the GetClassCodeForProcessingRun will return. You can extract the values of the parameters specified in the directive by reading them from the method's second parameter (called *arguments*). This code, in the ProcessDirective method, looks for a parameter called Year and uses it to build a string containing a variable declaration. The string is then returned from GetClassCodeForProcessingRun:

```
string copyright = string.Empty;
public override void ProcessDirective(
  string directiveName, IDictionary<string, string> arguments)
{
  copyright = "string copyright " +
              "= \"Copyright PH&V Information Services, " +
              arguments["Year"] +"\";";
}

public override string GetClassCodeForProcessingRun()
{
  return copyright;
}
```

Your directive processor can also add references and using/import statements to the generated transformation class to support the code added through GetClassCodeForProcessingRun. To add references to the generated transformation class, you just need to return the names of the libraries in a string array from the GetReferencesForProcessingRun method. If, for instance, the code being added to the generated transformation class needed classes from the System.XML namespace, you'd use code like this:

```
public override string[] GetReferencesForProcessingRun()
{
  return new string[] {"System.Xml"};
}
```

Similarly, you can specify namespaces to be added to the generated transformation class (as either using or Imports statements) by returning a string array from the GetImportsForProcessingRun method.

The generated code transformation class also includes pre- and post-initialization methods that are called before the TransformText method. You can return code that will be added to those methods from the GetPreInitializationCodeForProcessingRun and GetPostInitializationCodeForProcessingRun methods.

As you debug, remember that using Run Custom Tool does not cause Visual Studio to build your solution. As you make changes to your directive processor, you'll need to build your solution to pick up your latest changes before executing your template. And, again, because T4 locks the assemblies it uses, you might find you have to restart Visual Studio and recompile your code before retesting.

T4 significantly lowers the barriers for integrating code generation into your toolkit. However, as with any other application, you need to consider how you'll architect complete solutions to support maintainability and extensibility. The T4 code-generation process provides several places—each with its own costs and benefits—where you can refactor your code and insert it into the process. No matter which mechanism you use, you'll ensure your T4 solutions are well-architected. ■

Using T4 at Run Time

You can get a glimpse of what the generated transformation class looks like by taking advantage of Preprocessed Text Templates, which let you generate text at run time. Compiling or executing the results of Microsoft Text Template Transformation Toolkit (T4) code generation at run time is probably not something most developers would want to tackle. However, if you need several "similar but different" versions of an XML or an HTML document (or some other text), Preprocessed Text Templates let you use T4 to generate those documents at run time.

As with other T4 solutions, the first step in using T4 at run time is to add a T4 file to your project from the New Item dialog. But, instead of adding a Text Template file, you add a Preprocessed Text Template (also listed in the Visual Studio New Item dialog). A Preprocessed Text Template is identical to a standard T4 Text Template file except that the Custom Tool property is set to TextTemplatingFileProcessor instead of the usual TextTemplatingFileGenerator.

Unlike with a Text Template, the child file containing the generated code from a Preprocessed Text Template doesn't contain the code output by the generated transformation class. Instead, the file holds something that looks very much like one of your generated transformation classes: a class with the same name as the Preprocessed Text Template file with a method called TransformText. Calling that TransformText method at run time returns, as a string, what you'd expect to find in the code file of a T4 template: your generated code. So, for a Preprocessed Text Template file called GenerateHTML, you'd retrieve its generated text using code like this at run time:

```
GenerateHTML HtmlGen = new GenerateHTML();
string html = HtmlGen.TransformText();
```

—P.V.

PETER VOGEL *is a principal in PH&V Information Services. His last book was "Practical Code Generation in .NET" (Addison-Wesley Professional, 2010). PH&V Information Services specializes in facilitating the design of service-based architectures and in integrating .NET technologies into those architectures. In addition to his consulting practice, Vogel wrote Learning Tree International's service-oriented architecture design course, taught worldwide.*

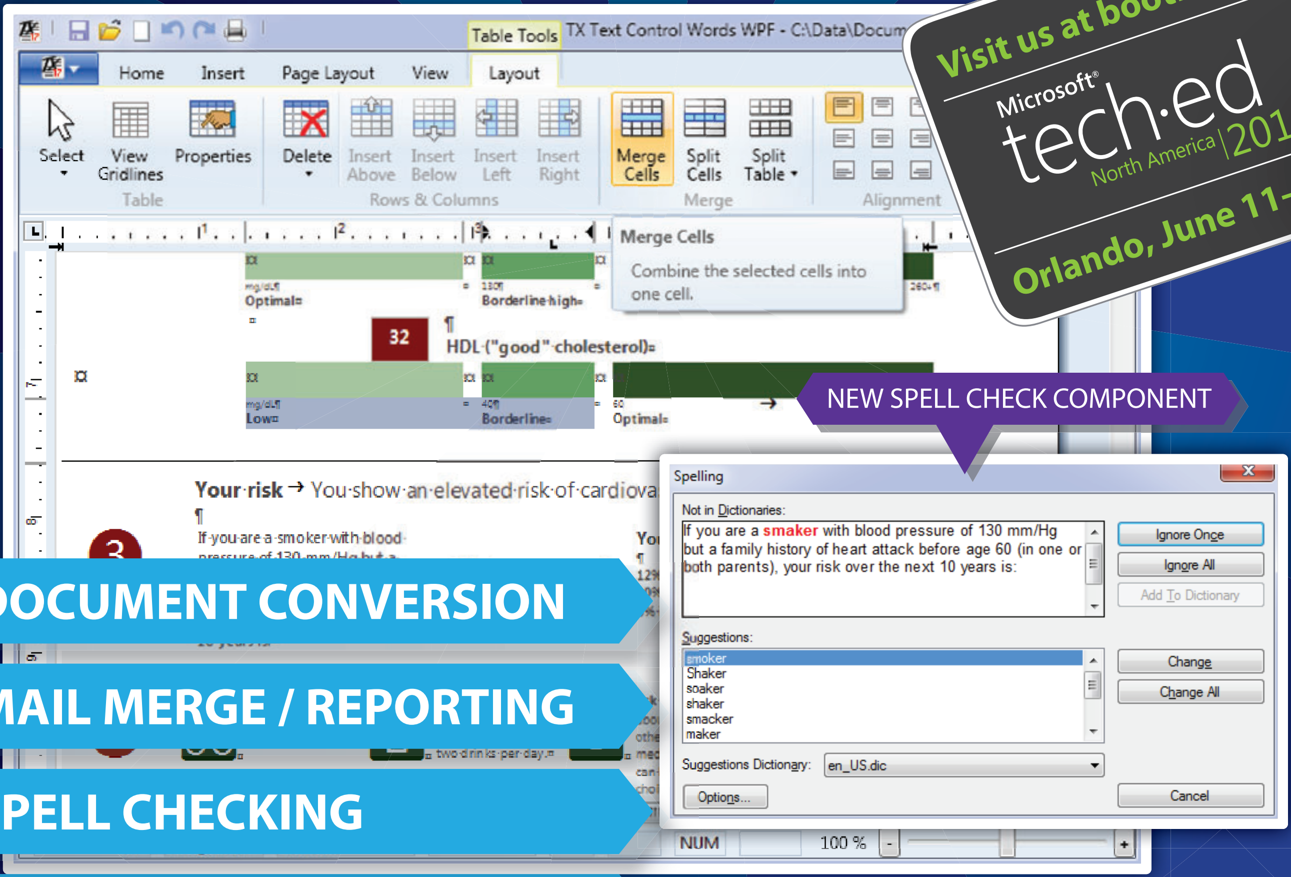
THANKS to the following technical expert for reviewing this article: *Gareth Jones*

WINDOWS FORMS | WPF | ASP.NET | ACTIVEX

WORD PROCESSING COMPONENTS

VERSION 17.0 RELEASED

Visit us at booth #2237
Microsoft® tech·ed North America | 2012
Orlando, June 11-14



If you are a smaker with blood pressure of 130 mm/Hg but a family history of heart attack before age 60 (in one or both parents), your risk over the next 10 years is:

smoker
Shaker
soaker
shaker
smacker
maker

en_US.dic

NEW SPELL CHECK COMPONENT

DOCUMENT CONVERSION

MAIL MERGE / REPORTING

SPELL CHECKING

RICH TEXT EDITING

TX TEXTCONTROL® word processing components

Microsoft® Visual Studio® PARTNER

Word Processing Components
for Windows Forms, WPF & ASP.NET

WWW.TEXTCONTROL.COM

US +1 877-462-4772 (toll-free)
EU +49 421-4270671-0

Multimodal Communication Using Kinect

Leland Holmquest

In the April issue (msdn.microsoft.com/magazine/hh882450) I introduced you to "Lily," a virtual assistant intended to help office workers in their daily tasks. I demonstrated how to use the Microsoft Kinect for Windows SDK to create context-aware dialogue, enabling Lily to listen and respond appropriately with respect to the intentions of the user.

Now I'll take you through the next step in achieving a natural UI by leveraging the Kinect device's skeletal tracking to facilitate user interaction through gestures. Then I'll tie the two concepts together and demonstrate multimodal communication by having Lily's output dependent on not only what gesture was made, but also what spoken command was issued. By combining these two modes of communication, the user comes away with a much richer experience and gets a step closer to ubiquitous computing. The presentation of Lily is in the form of a Windows Presentation Foundation (WPF) application.

This article discusses the Kinect for Windows SDK beta 2, a prerelease technology. All related information is subject to change.

This article discusses:

- This article discusses:
- Initializing Kinect
- Tracking users
- Video and depth streams
- Handling events
- Evaluating project state

Technologies discussed:

Kinect

Code download available at:

code.msdn.microsoft.com/mag201204Kinect

Initializing Kinect

The first step in using the Kinect device is to construct a Runtime, setting various parameters. **Figure 1** shows the configuration I chose for Lily.

When setting up a Kinect device, a number of options are available. First, notice the first line of code in **Figure 1**. The Kinect for Windows SDK beta 2 has a different constructor for the Runtime. By referencing an index (Runtime.Kinects[0];), it's simple to attach multiple Kinect units to the application. In this application I've limited it to a single Kinect device, so by definition the Runtime must be at location [0]. You can iterate through the collection of Runtime.Kinects to handle multiple Kinect units if available. Next, I need to tell the Kinect device what capabilities are going to be used. This is done by passing the desired capabilities into the Initialize method. There are four values from which to choose:

- **UseColor** enables the application to process the color image information.
- **UseDepth** enables the application to make use of the depth image information.
- **UseDepthAndPlayerIndex** enables the application to make use of the depth image information as well as the index generated by the skeleton tracking engine.
- **UseSkeletalTracking** enables the application to use the skeleton tracking data.

Figure 1 **Kinect Runtime Construction**

```
// Initialize Kinect
nui = Runtime.Kinects[0];// new Runtime();
nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex |
  RuntimeOptions.UseDepth | RuntimeOptions.UseColor |
  RuntimeOptions.UseSkeletalTracking);
nuiInitialized = true; nui.SkeletonEngine.TransformSmooth = true;
nui.SkeletonEngine.SmoothParameters = new TransformSmoothParameters
{
  Smoothing = 0.75f,
  Correction = 0.0f,
  Prediction = 0.0f,
  JitterRadius = 0.05f,
  MaxDeviationRadius = 0.04f
};
nui.VideoStream.Open(ImageStreamType.Video, 2,
  ImageResolution.Resolution640x480, ImageType.Color);
nui.DepthStream.Open(ImageStreamType.Depth, 2,
  ImageResolution.Resolution320x240, ImageType.DepthAndPlayerIndex);
```

Passing in these values tells the API what subsystems in the Kinect device are going to be used so the appropriate parts of the multistage pipeline of the Runtime can be started. It's important to note that you can't access capabilities later in the application that aren't declared during the initialization. For example, if the only option selected was RuntimeOptions.UseColor and later using the depth information was required, it wouldn't be available. Therefore, I've passed in all of the values available, indicating that I intend to use the full capabilities of the Kinect device.

Tracking Users

Before discussing the next section in the code, let's look at what the Kinect device is really giving us. When using the skeleton tracking capability, the Kinect device can track up to two active humans interacting with the system. It achieves this by creating a collection of 20 joints and associating an ID with each. **Figure 2** shows what joints are being modeled.

Figure 3 is an image of the joints being captured from two separate users.

In order for a skeleton to become active, the Kinect device must be able to see the user from head to foot. Once a skeleton is active, if a joint goes out of view, the Kinect device will try to interpolate where that part of the skeleton is. If you're going to build Kinect-enabled applications, I strongly encourage you to create a simple application just to watch the skeleton streams and interact with the Kinect device. Make sure you have multiple users participate and set up scenarios where obstructions come between your users and the Kinect device—scenarios that mimic what your application will experience once deployed. This will give you an excellent understanding of how the skeleton tracking works and what it's capable of, as well as what limitations you might want to address. You'll quickly see how amazing the technology is and how creative it can be in the interpolation.

In some scenarios (like the one represented by Project Lily) the speed and choppiness of this interpolation can be distracting and unproductive. Therefore the API exposes the ability to control a level of smoothing. Referring to **Figure 1** again, first use the SkeletonEngine on the Runtime to set the TransformSmooth to true. This tells the Kinect device that you want to affect the smoothness of the data being rendered. Then set the SmoothParameters. Here's a brief description of each of the TransformSmoothParameters:

- **Correction** controls the amount of correction with values ranging from 0 to 1.0 and a default value of .5.
- **JitterRadius** controls the jitter-reduction radius. The value passed in represents the radius desired in meters. The default value is set to 0.05, which translates into 5 cm. Any jitter that goes beyond this radius is clamped to the radius.
- **MaxDeviationRadius** controls the maximum radius (in meters) that corrected positions can deviate from the raw data. The default value is 0.04.
- **Prediction** controls the number of predicted frames.
- **Smoothing** controls the amount of smoothing with a range of 0 to 1.0. The documentation makes a specific point that smoothing has an impact on latency; increasing smoothing increases latency. The default value is 0.5. Setting the value to 0 causes the raw data to be returned.

Video and Depth Streams

You'll want to experiment with these settings in your own application, depending on the requirements that you're fulfilling. The last thing needed for this application is to open the VideoStream and the DepthStream. This facilitates viewing the video images coming from the color camera and the depth images coming from the depth camera, respectively. Later on I'll show you how this gets connected to the WPF application.

The Open method requires four parameters. The first is streamType. It represents the type of stream that's being opened (for example, Video). The second parameter is poolSize. This represents the number of frames that the Runtime is to buffer. The maximum value is 4. The third parameter is resolution, which represents the resolution of the desired images. The values include 80x60, 640x480, 320x240 and 1280x1024 to match your needs. And the last parameter indicates the desired type of image (for example, Color).

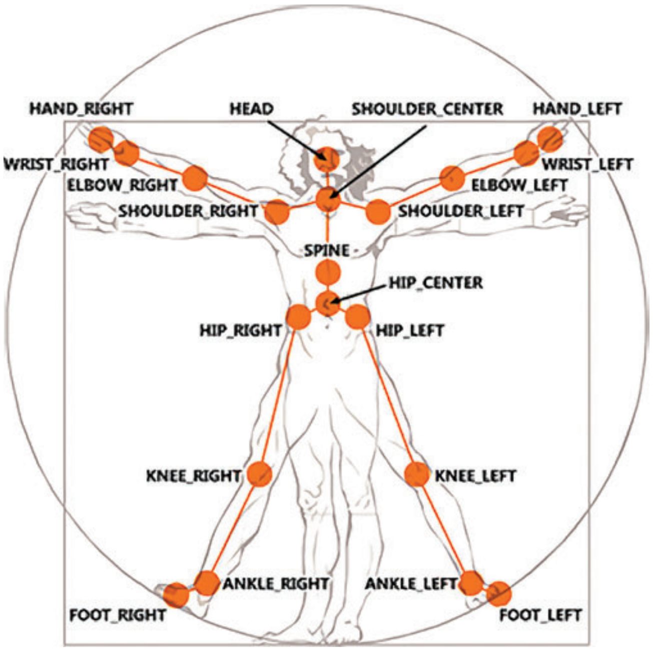


Figure 2 **The 20 Joints that Are Modeled in Kinect**

Kinect Events

With the Runtime successfully initialized, it's time to wire up the events made available from the Runtime to the application. For Lily, the first two events that will be handled are used simply to give the end user a graphical view of the color images and the depth images. First, let's look at the method that's handling the Runtime.VideoFrameReady event. This event passes an ImageFrameReadyEventArgs as its event argument. The nui_VideoFrameReady method is where Lily handles the event, as shown in the following code:

```
void nui_VideoFrameReady(object sender, ImageFrameReadyEventArgs e)
{
  // Pull out the video frame from the eventargs and
  // load it into our image object.
  PlanarImage image = e.ImageFrame.Image;
  BitmapSource source =
    BitmapSource.Create(image.Width, image.Height, 96, 96,
    PixelFormats.Bgr32, null, image.Bits,
    image.Width * image.BytesPerPixel);
  colorImage.Source = source;
}
```

The Kinect for Windows API makes this method simple. The ImageFrameReadyEventArgs contains an ImageFrame.Image. I convert that to a BitmapSource and then pass that BitmapSource to an Image control in the WPF application. The frame coming from the Kinect device's color camera is thus displayed on the application, like what you see in **Figure 3**.

The DepthFrameReady event, which is being handled by nui_DepthFrameReady, is similar but needs a little more work to get a useful presentation. You can look at this method in the code download, which is the same as last month's article (code.msdn.microsoft.com/mag201204Kinect). I didn't create this method myself, but found it used in a number of examples online.

If Lily is tracking the user's hands when that isn't the desired behavior, it quickly becomes tedious and tiring.

The event handler that really starts to get interesting is the nui_SkeletonFrameReady method. This method handles the SkeletonFrameReady event and gets passed in SkeletonFrameReadyEventArgs, as shown in **Figure 4**.

One thing I found necessary to put into this application was that first conditional in **Figure 4**. When the user doesn't want the application to track her hand movements, there are spoken commands that set the trackHands variable, which in turn determines whether the hands are tracked. If trackHands is set to false, the code simply returns out of this method. If Lily is tracking the user's hands when that isn't the desired behavior, it quickly becomes tedious and tiring.

Similarly, if no skeletons are being tracked (either there are no users, or they're out of the view range of the Kinect device) then there's no sense in continuing to evaluate the data, so the code returns out of the method. However, if there *is* a skeleton and the user wants the hands tracked, then the code continues to evaluate. The HoverButton project (bit.ly/nUA2RC) comes with sample code. Most of this method came from those examples. One of the interesting things happening in this method is that the code checks to see which hand on the user is physically higher. It then makes the assumption that the highest hand is the one being used to potentially select a button. The code then goes on to determine whether a button is being hovered over, and renders a "hand" on the screen in the place that's representative of the screen with respect to the location of the user's hand. In other words, as the user moves his hand, a graphical hand is moved around the screen in like fashion. This gives the user a natural interface, no longer bound by the cord of the mouse. The *user* is the controller.

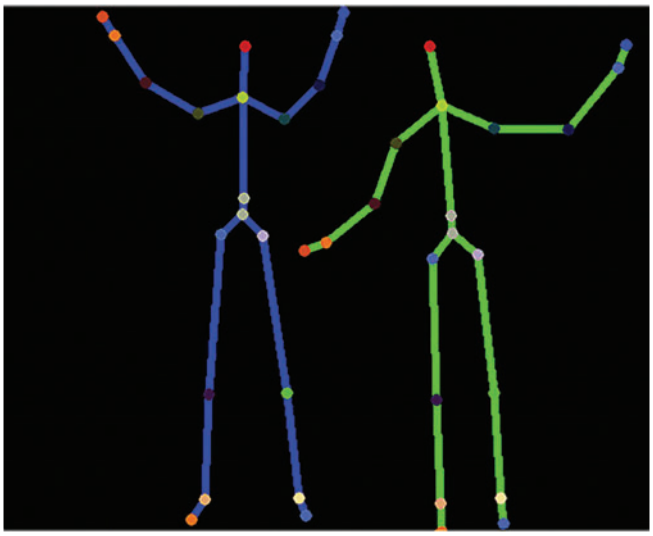


Figure 3 **Two Active Skeletons**

The next item of interest is when the system determines that one of the HoverButtons is clicked. Lily has a total of eight buttons on the screen. Each has an on_click event handler wired in. At this point, I need to cover three special classes: ButtonActionEvaluator, LilyContext and MultiModalReactions.

The action of clicking a button has a corresponding event associated with it, but Lily takes this single action and checks if it can be coupled to a corresponding audio command to evaluate as a multimodal communication that would take on a higher level of meaning. For example, clicking one of the HoverButtons represents the intention of selecting a project. With that information, the only action required by the system is to note that the context, with respect to the project being worked on, has changed. No further action is desired. However, if the user either previously made an unsatisfied request to "open the project plan" or subsequently makes the same request, the application must put these two disparate pieces of data together to create a higher order of meaning (the communication coming from two separate modes makes this multimodal communication) and respond accordingly. To make this all occur in a seamless fashion, the following design was implemented.

The ButtonActionEvaluator class is implemented as a singleton and implements the INotifyPropertyChanged interface. This class also exposes a PropertyChanged event that's handled by the LilyContext class (also a singleton). The following code probably requires a bit of explaining, even though it looks innocuous enough:

```
void buttonActionEvaluator_PropertyChanged(
  object sender, System.ComponentModel.PropertyChangedEventArgs e)
{
  if (MultiModalReactions.ActOnMultiModalInput(
    buttonActionEvaluator.EvaluateCriteria()) ==
    PendingActionResult.ClearPendingAction)
    buttonActionEvaluator.ActionPending = PendingAction.NoneOutstanding;
}
```

Evaluating Lily's State

First, the preceding code calls the EvaluateCriteria method on the buttonActionEvaluator class. This method simply returns a numerical representation for the state as defined by the ActionPending and SelectedButton properties. This is at the heart of how the application is able to infer meaning through the use of multimodal communication. In traditional applications, the desired action is evaluated by looking at the state of a single event or property (for example, button1.clicked). But with Lily, the state being evaluated (from the multimodal perspective) is the combination of two otherwise separate properties. In other words, each property has significance and requires actions independently, but when evaluated together, they take on a new and higher level of meaning.

That numeric representation of the combined state is then passed into the ActOnMultiModalInput method on the MultiModalReactions class. This method implements a large switch statement that handles all of the permutations possible. (This is a rudimentary implementation that was used to illustrate the point. Future iterations of Lily will replace this implementation with more advanced techniques such as state machines and machine learning to enhance the overall experience and usability.) If this method results in the intention of the user being satisfied (for example, the user intends for the system to open the project plan for Project Lily), the return type is PendingActionResult.ClearPendingAction. This leaves the context of the system still in the frame of reference of Project Lily, but there's no action waiting to be executed in the queue. If the user's intention is still unsatisfied, the PendingActionResult.LeavePendingActionInPlace is returned, telling the system that whatever action was taken hasn't yet satisfied the user's intention, and to therefore not clear the pending action.

In the first article I showed how to create grammars that are specific to a given domain or context. The Kinect unit, leveraging the Speech Recognition Engine, used these grammars, loading and unloading them to meet the needs of the user. This created an application that doesn't require the user to stick to a scripted interaction. The user can go in whatever direction she desires and change directions without having to reconfigure the application. This created a natural way of establishing dialogue between the human user and computer application.

Higher Level of Meaning

In this article I demonstrated how to couple actions resulting from context-aware grammars to a user's physical gesturing in the form of selecting buttons by hovering one's hand over a button. Each event (speechDetected and buttonClicked) can be handled individually and independently. But in addition, the two events can be correlated by the system, bringing a higher level of meaning to the events and acting accordingly.

Figure 4 **nui_SkeletonFrameReady**

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
  renderSkeleton(sender, e);
  if (!trackHands)
    return;// If the user doesn't want to use the buttons, simply return.
  if (e.SkeletonFrame.Skeletons.Count() == 0)
    return;// No skeletons, don't bother processing.
  SkeletonFrame skeletonSet = e.SkeletonFrame;
  SkeletonData firstPerson = (from s in skeletonSet.Skeletons
                              where s.TrackingState ==
                              SkeletonTrackingState.Tracked
                              orderby s.UserIndex descending
                              select s).FirstOrDefault();
  if (firstPerson == null)
    return;// If no one is being tracked, no sense in continuing.

  JointsCollection joints = firstPerson.Joints;
  Joint righthand = joints[JointID.HandRight];
  Joint lefthand = joints[JointID.HandLeft];
  // Use the height of the hand to figure out which is being used.
  Joint joinCursorHand = (righthand.Position.Y > lefthand.Position.Y)
    ? righthand
    : lefthand;

  float posX = joinCursorHand.ScaleTo((int)SystemParameters.PrimaryScreenWidth,
    (int)SystemParameters.PrimaryScreenHeight).Position.X;
  float posY = joinCursorHand.ScaleTo((int)SystemParameters.PrimaryScreenWidth,
    (int)SystemParameters.PrimaryScreenHeight).Position.Y;
  Joint scaledCursorJoint = new Joint
  {
    TrackingState = JointTrackingState.Tracked,
    Position = new Microsoft.Research.Kinect.Nui.Vector
    {
      X = posX,
      Y = posY,
      Z = joinCursorHand.Position.Z
    }
  };
  OnButtonLocationChanged(kinectButton, buttons, (int)scaledCursorJoint.Position.X,
    (int)scaledCursorJoint.Position.Y);
}
```

I hope you're as excited about the capabilities that Kinect puts into our hands as I am. I think Microsoft brought us to the edge where human computing interfaces can take leaps forward. As testimony to this, as I developed Lily, there were times when I was testing different components and sections of code. As the application matured and I was able to actually "talk" to Lily, I would find something wrong, switch to my second monitor and start looking up something in the documentation or elsewhere. But I would continue to verbally interact with Lily, asking her to execute tasks or even to shut down. I found that when Lily was unavailable, I became perturbed because the amount of enabling that Lily represented was significant—taking petty tasks off my hands through simple verbal communications.

And incorporating little "tricks" into the dialogue mechanism (for example, random but contextually and syntactically correct responses) made the adoption of the application intuitive and satisfying. Kinect truly makes your body the controller. Where you go with it is limited only by your imagination. What will you Kinect? ■

LELAND HOLMQUEST *is an enterprise strategy consultant at Microsoft. Previously he worked for the Naval Surface Warfare Center Dahlgren. He's working on his Ph.D. in Information Technology at George Mason University.*

THANKS to the following technical expert for reviewing this article: *Mark Schwesinger*

Dive into Neural Networks

An artificial neural network (usually just called a neural network) is an abstraction loosely modeled on biological neurons and synapses. Although neural networks have been studied for decades, many neural network code implementations on the Internet are not, in my opinion, explained very well. In this month's column, I'll explain what artificial neural networks are and present C# code that implements a neural network.

The best way to see where I'm headed is to take a look at **Figure 1** and **Figure 2**. One way of thinking about neural networks is to consider them numerical input-output mechanisms. The neural network in **Figure 1** has three inputs labeled x0, x1 and x2, with values 1.0, 2.0 and 3.0, respectively. The neural network has two outputs labeled y0 and y1, with values 0.72 and -0.88, respectively. The neural network in **Figure 1** has one layer of so-called hidden neurons and can be described as a three-layer, fully connected, feedforward network with three inputs, two outputs and four hidden neurons. Unfortunately, neural network terminology varies quite a bit. In this article, I'll generally—but not always—use the terminology described in the excellent neural network FAQ at bit.ly/wfikTI.

Figure 2 shows the output produced by the demo program presented in this article. The neural network uses both a sigmoid activation function and a tanh activation function. These functions are suggested by the two equations with the Greek letters phi in **Figure 1**. The outputs produced by a neural network depend on the values of a set of numeric weights and biases. In this example, there are a total of 26 weights and biases with values 0.10, 0.20 ... -5.00. After the weight and bias values are loaded into the neural network, the demo program loads the three input values (1.0, 2.0, 3.0) and then performs a series of computations as suggested by the messages about the input-to-hidden sums and the hidden-to-output sums. The demo program concludes by displaying the two output values (0.72, -0.88).

I'll walk you through the program that produced the output shown in **Figure 2**. This column assumes you have intermediate programming skills but doesn't assume you know anything about neural networks. The demo program is coded using the C# language but you should have no trouble refactoring the demo code to another language such as Visual Basic .NET or Python. The program presented in this article is essentially a tutorial and a platform for experimentation; it does not directly solve any practical problem, so I'll explain how you can expand the code to solve meaningful problems. I think you'll find the information quite interesting, and some of the programming techniques can be valuable additions to your coding skill set.

Code download available at code.msdn.microsoft.com/mag201205TestRun.

Modeling a Neural Network

Conceptually, artificial neural networks are modeled on the behavior of real biological neural networks. In **Figure 1** the circles represent neurons where processing occurs and the arrows represent both information flow and numeric values called weights. In many situations, input values are copied directly into input neurons without any weighting and emitted directly without any processing, so the first real action occurs in the hidden layer neurons. Assume that input values 1.0, 2.0 and 3.0 are emitted from the input neurons. If you examine **Figure 1**, you can see an arrow representing a weight value between each of the three input neurons and each of the four hidden neurons. Suppose the three weight arrows shown pointing into the top hidden neuron are named w00, w10 and w20. In this notation the first index represents the index of the source input neuron and the second index represents the index of the destination hidden neuron. Neuron processing occurs in three steps. In the first step, a weighted sum is computed. Suppose w00 = 0.1, w10 = 0.5 and w20 = 0.9. The weighted sum for the top hidden neuron is (1.0)(0.1) + (2.0)(0.5) + (3.0)(0.9) = 3.8. The second processing step is to add a bias value. Suppose the bias value is -2.0; then the adjusted weighted sum becomes 3.8 + (-2.0) = 1.8. The third step is to apply an activation function to the adjusted weighted sum. Suppose the activation function is the

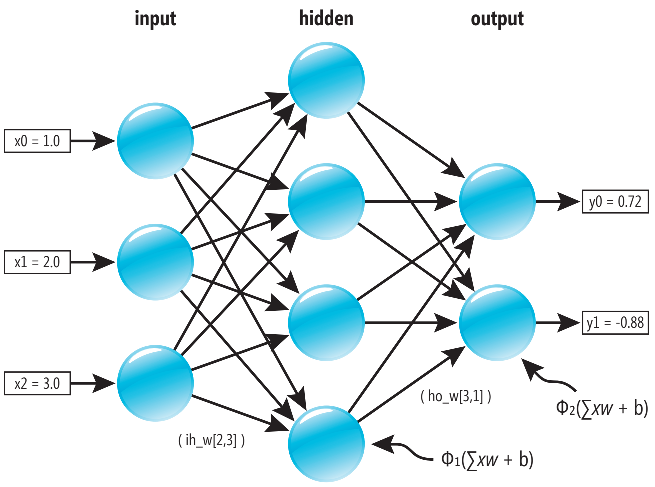
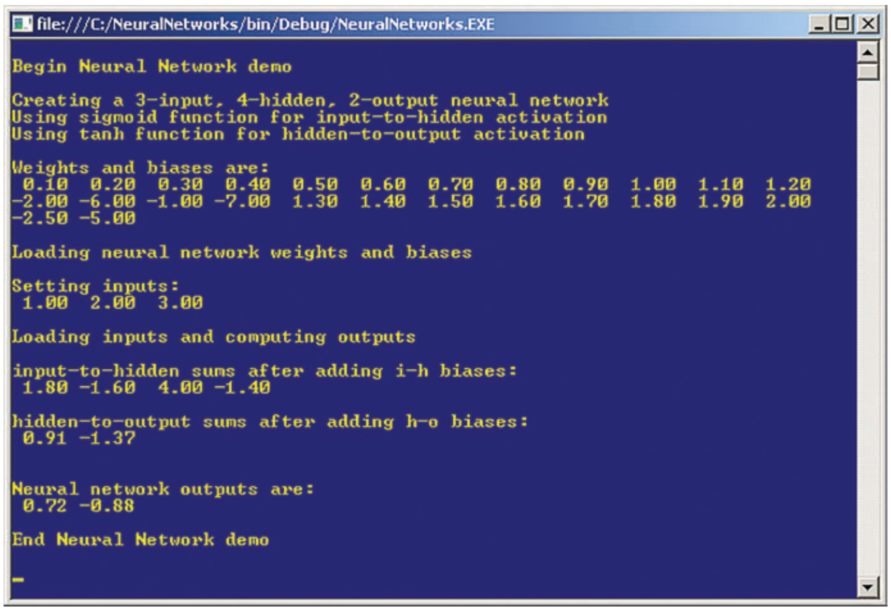


Figure 1 **Neural Network Structure**



```
file:///C:/NeuralNetworks/bin/Debug/NeuralNetworks.EXE

Begin Neural Network demo

Creating a 3-input, 4-hidden, 2-output neural network
Using sigmoid function for input-to-hidden activation
Using tanh function for hidden-to-output activation

Weights and biases are:
 0.10  0.20  0.30  0.40  0.50  0.60  0.70  0.80  0.90  1.00  1.10  1.20
-2.00 -6.00 -1.00 -7.00  1.30  1.40  1.50  1.60  1.70  1.80  1.90  2.00
-2.50 -5.00

Loading neural network weights and biases

Setting inputs:
 1.00  2.00  3.00

Loading inputs and computing outputs

input-to-hidden sums after adding i-h biases:
 1.80 -1.60  4.00 -1.40

hidden-to-output sums after adding h-o biases:
 0.91 -1.37


Neural network outputs are:
 0.72 -0.88

End Neural Network demo

_
```

Figure 2 **Neural Network Demo Program**

sigmoid function defined by 1.0 / (1.0 * Exp(-x)), where Exp represents the exponential function. The output from the hidden neuron becomes 1.0 / (1.0 * Exp(-1.8)) = 0.86. This output then becomes part of the weighted sum input into each of the output layer neurons. In **Figure 1**, this three-step process is suggested by the equation with the Greek letter phi: weighted sums (xw) are computed, a bias (b) is added and an activation function (phi) is applied.

After all hidden neuron values have been computed, output layer neuron values are computed in the same way. The activation function used to compute output neuron values can be the same function used when computing the hidden neuron values, or a different activation function can be used. The demo program shown running in **Figure 2** uses the hyperbolic tangent function as the hidden-to-output activation function. After all output layer neuron values have been computed, in most situations these values are not weighted or processed but are simply emitted as the final output values of the neural network.

Internal Structure

The key to understanding the neural network implementation presented here is to closely examine **Figure 3**, which, at first glance, might appear extremely complicated. But bear with me—the figure is not nearly as complex as it might first appear. **Figure 3** shows a total of eight arrays and two matrices. The first array is labeled this.inputs. This array holds the neural network input values, which are 1.0, 2.0 and 3.0 in this example. Next comes the set of weight values that are used to compute values in the so-called hidden layer. These weights are stored in a 3 x 4 matrix labeled i-h weights where the i-h stands for input-to-hidden. Notice in **Figure 1** that the demo neural network has four hidden neurons. The i-h weights matrix has a number of rows equal to the number of inputs and a number of columns equal to the number of hidden neurons.

The array labeled i-h sums is a scratch array used for computation. Note that the length of the i-h sums array will always be the same as the number of hidden neurons (four, in this example). Next comes an array labeled i-h biases. Neural network biases are additional weights used to compute hidden and output layer neurons. The length of the i-h biases array will be the same as the length of the i-h sums array, which in turn is the same as the number of hidden neurons.

The array labeled i-h outputs is an intermediate result and the values in this array are used as inputs to the next layer. The i-h sums array has length equal to the number of hidden neurons.

Next comes a matrix labeled h-o weights where the h-o stands for hidden-to-output. Here the h-o weights matrix has size 4 x 2 because there are four hidden neurons and two outputs. The h-o sums array, the h-o biases array and the this.outputs array all have lengths equal to the number of outputs (two, in this example).

The array labeled weights at the bottom of **Figure 3** holds all the input-to-hidden and hidden-to-output weights and biases. In this example, the length of the weights array is (3 * 4) + 4 + (4 * 2) + 2 = 26. In general, if Ni is the number of input values, Nh is the number of hidden neurons and No is the number of outputs, then the length of the weights array will be Nw = (Ni * Nh) + Nh + (Nh * No) + No.

Computing the Outputs

After the eight arrays and two matrices described in the previous section have been created, a neural network can compute its output based on its inputs, weights and biases. The first step is to copy input values into the this.inputs array. The next step is to assign values to the weights array. For the purposes of a demonstration you can use any weight values you like. Next, values in the weights array are copied to the i-h weights matrix, the i-h biases array, the h-o weights matrix and the h-o biases array. **Figure 3** should make this relationship clear.

The values in the i-h sums array are computed in two steps. The first step is to compute the weighted sums by multiplying the values in the inputs array by the values in the appropriate column of the i-h weights matrix. For example, the weighted sum for hidden neuron [3] (where I'm using zero-based indexing) uses each input value and the values in column [3] of the i-h weights matrix: (1.0)(0.4) + (2.0)(0.8) + (3.0)(1.2) = 5.6. The second step when computing i-h sum values is to add each bias value to the current i-h sum value. For example, because i-h biases [3] has value -7.0, the value of i-h sums [3] becomes 5.6 + (-7.0) = -1.4.

After all the values in the i-h sums array have been calculated, the input-to-hidden activation function is applied to those sums to produce the input-to-hidden output values. There are many possible activation functions. The simplest activation function is called the step function, which simply returns 1.0 for any input value greater than zero and returns 0.0 for any input value less than or equal to

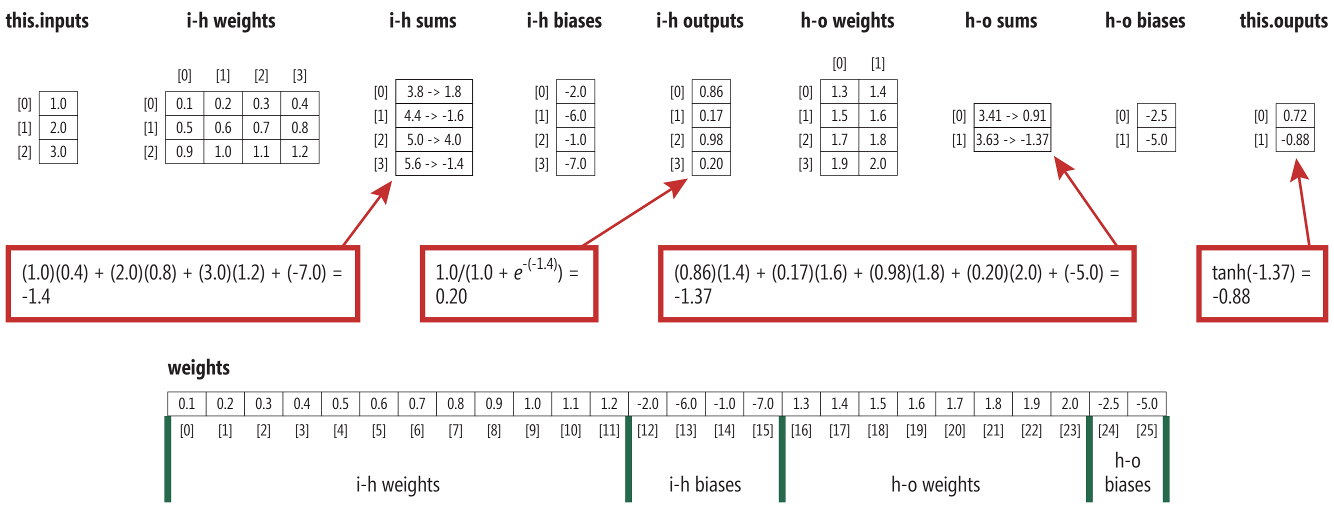


Figure 3 **Neural Network Internal Structure**

zero. Another common activation function, and the one used in this article, is the sigmoid function, which is defined as f(x) = 1.0 / (1.0 * Exp(-x)). The graph of the sigmoid function is shown in **Figure 4**.

Notice the sigmoid function returns a value in the range strictly greater than zero and strictly less than one. In this example, if the value for i-h sums [3] after the bias value has been added is -1.4, then the value of i-h outputs [3] becomes 1.0 / (1.0 * Exp(-(-1.4))) = 0.20.

After all the input-to-hidden output neuron values have been computed, those values serve as the inputs for the hidden-to-output layer neuron computations. These computations work in the same way as the input-to-hidden computations: preliminary weighted sums are calculated, biases are added and then an activation function is applied. In this example I use the hyperbolic tangent function, abbreviated as tanh, for the hidden-to-output activation function. The tanh function is closely related to the sigmoid function. The graph of the tanh function has an S-shaped curve similar to the sigmoid function, but tanh returns a value in the range (-1,1) instead of in the range (0,1).

Combining Weights and Biases

All of the neural network implementations I've seen on the Internet don't maintain separate weight and bias arrays, but instead combine weights and biases into the weights matrix. How is this possible? Recall that the computation of the value of input-to-hidden neuron [3] resembled (i0 * w03) + (i1 * w13) + (i2 * w23) + b3, where i0 is input value [0], w03 is the weight for input [0] and neuron [3], and b3 is the bias value for hidden neuron [3]. If you create an additional, fake input [4] that has a dummy value of 1.0, and an additional row of weights that hold the bias values, then the previously described computation becomes: (i0 * w03) + (i1 * w13) + (i2 * w23) + (i3 * w33), where i3 is the dummy 1.0 input value and w33 is the bias. The argument is that this approach simplifies the neural network model. I disagree. In my opinion, combining weights and biases makes a neural network model more difficult to understand and more error-prone to implement. However, apparently I'm the only author who seems to have this opinion, so you should make your own design decision.

Implementation

I implemented the neural network shown in **Figures 1**, **2** and **3** using Visual Studio 2010. I created a C# console application named NeuralNetworks. In the Solution Explorer window I right-clicked on file Program.cs and renamed it to NeuralNetworksProgram.cs, which also changed the template-generated class name to NeuralNetworksProgram. The overall program structure, with most WriteLine statements removed, is shown in **Figure 5**.

I deleted all the template-generated using statements except for the one referencing the System namespace. In the Main function, after displaying a begin message, I instantiate a NeuralNetwork object named nn with three inputs, four hidden neurons and two outputs. Next, I assign 26 arbitrary weights and biases to an array named weights. I load the weights into the neural network object using a method named SetWeights. I assign values 1.0, 2.0 and 3.0 to an array named xValues. I use method ComputeOutputs to load the input values into the neural network and determine the resulting outputs, which I fetch into an array named yValues. The demo concludes by displaying the output values.

The NeuralNetwork Class

The NeuralNetwork class definition starts:

```
class NeuralNetwork
{
  private int numInput;
  private int numHidden;
  private int numOutput;
...
```

As explained in the previous sections, the structure of a neural network is determined by the number of input values, the number of hidden layer neurons and the number of output values. The class definition continues as:

```
private double[] inputs;
private double[][] ihWeights; // input-to-hidden
private double[] ihSums;
private double[] ihBiases;
private double[] ihOutputs;
private double[][] hoWeights;  // hidden-to-output
private double[] hoSums;
private double[] hoBiases;
private double[] outputs;
...
```

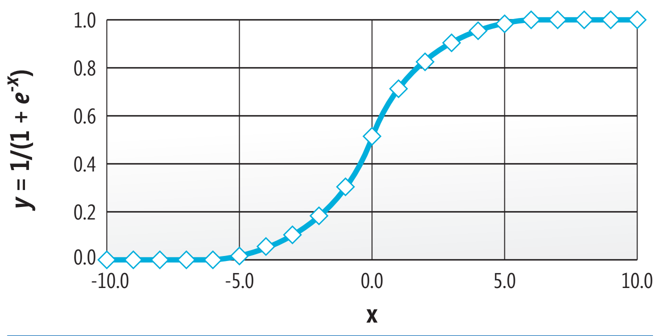


Figure 4 **The Sigmoid Function**

These seven arrays and two matrices correspond to the ones shown in **Figure 3**. I use an ih prefix for input-to-hidden data and an ho prefix for hidden-to-output data. Recall that the values in the ihOutputs array serve as the inputs for the output layer computations, so naming this array precisely is a bit troublesome.

Figure 6 shows how the NeuralNetwork class constructor is defined.

After copying the input parameter values numInput, numHidden and numOutput into their respective class fields, each of the nine member arrays and matrices are allocated with the sizes I explained earlier. I implement matrices as arrays of arrays rather than using the C# multidimensional array type so that you can more easily refactor my code to a language that doesn't support multidimensional array types. Because each row of my matrices must be allocated, it's convenient to use a helper method such as MakeMatrix.

Figure 5 **Neural Network Program Structure**

```
using System;
namespace NeuralNetworks
{
  class NeuralNetworksProgram
  {
    static void Main(string[] args)
    {
      try
      {
        Console.WriteLine("\nBegin Neural Network demo\n");
        NeuralNetwork nn = new NeuralNetwork(3, 4, 2);

        double[] weights = new double[] {
          0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2,
          -2.0, -6.0, -1.0, -7.0,
          1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0,
          -2.5, -5.0 };

        nn.SetWeights(weights);
        double[] xValues = new double[] { 1.0, 2.0, 3.0 };
        double[] yValues = nn.ComputeOutputs(xValues);
        Helpers.ShowVector(yValues);
        Console.WriteLine("End Neural Network demo\n");
      }
      catch (Exception ex)
      {
        Console.WriteLine("Fatal: " + ex.Message);
      }
    }
  }

  class NeuralNetwork
  {
    // Class members here
    public NeuralNetwork(int numInput, int numHidden, int numOutput) { ... }
    public void SetWeights(double[] weights) { ... }
    public double[] ComputeOutputs(double[] xValues) { ... }
    private static double SigmoidFunction(double x) { ... }
    private static double HyperTanFunction(double x) { ... }
  }

  public class Helpers
  {
    public static double[][] MakeMatrix(int rows, int cols) { ... }
    public static void ShowVector(double[] vector) { ... }
    public static void ShowMatrix(double[][] matrix, int numRows) { ... }
  }
} // ns
```

The SetWeights method accepts an array of weights and bias values and populates ihWeights, ihBiases, hoWeights and hoBiases. The method begins like this:

```
public void SetWeights(double[] weights)
{
  int numWeights = (numInput * numHidden) +
    (numHidden * numOutput) + numHidden + numOutput;
  if (weights.Length != numWeights)
    throw new Exception("xxxxxx");
  int k = 0;
...
```

As explained earlier, the total number of weights and biases, Nw, in a fully connected feedforward neural network is (Ni * Nh) + (Nh * No) + Nh + No. I do a simple check to see if the weights array parameter has the correct length. Here, "xxxxxx" is a stand-in for a descriptive error message. Next, I initialize an index variable k to the beginning of the weights array parameter. Method SetWeights concludes:

```
for (int i = 0; i < numInput; ++i)
  for (int j = 0; j < numHidden; ++j)
    ihWeights[i][j] = weights[k++];
for (int i = 0; i < numHidden; ++i)
  ihBiases[i] = weights[k++];
for (int i = 0; i < numHidden; ++i)
  for (int j = 0; j < numOutput; ++j)
    hoWeights[i][j] = weights[k++];
for (int i = 0; i < numOutput; ++i)
  hoBiases[i] = weights[k++]
}
```

Each value in the weights array parameter is copied sequentially into ihWeights, ihBiases, hoWeights and hoBiases. Notice no values are copied into ihSums or hoSums because those two scratch arrays are used for computation.

Computing the Outputs

The heart of the NeuralNetwork class is method ComputeOutputs. The method is surprisingly short and simple and begins:

```
public double[] ComputeOutputs(double[] xValues)
{
  if (xValues.Length != numInput)
    throw new Exception("xxxxxx");
  for (int i = 0; i < numHidden; ++i)
    ihSums[i] = 0.0;
  for (int i = 0; i < numOutput; ++i)
    hoSums[i] = 0.0;
...
```

First I check to see if the length of the input x-values array is the correct size for the NeuralNetwork object. Then I zero out the ihSums and hoSums arrays. If ComputeOutputs is called only once, then this explicit initialization is not necessary, but if ComputeOutputs is called more than once—because ihSums and hoSums are accumulated values—the explicit initialization is absolutely necessary. An alternative design approach is to not declare and allocate ihSums and hoSums as class members, but instead make them local to the ComputeOutputs method. Method ComputeOutputs continues:

```
for (int i = 0; i < xValues.Length; ++i)
  this.inputs[i] = xValues[i];
for (int j = 0; j < numHidden; ++j)
  for (int i = 0; i < numInput; ++i)
    ihSums[j] += this.inputs[i] * ihWeights[i][j];
...
```

The values in the xValues array parameter are copied to the class inputs array member. In some neural network scenarios, input parameter values are normalized, for example by performing a linear transform so that all inputs are scaled between -1.0 and +1.0, but here no normalization is performed. Next, a nested loop computes the weighted sums as shown in **Figures 1** and **3**. Notice that in order to index ihWeights in standard form where index i is the row index and index j is the column index, it's necessary to have j in the outer loop. Method ComputeOutputs continues:

```
for (int i = 0; i < numHidden; ++i)
  ihSums[i] += ihBiases[i];
for (int i = 0; i < numHidden; ++i)
  ihOutputs[i] = SigmoidFunction(ihSums[i]);
...
```

Each weighted sum is modified by adding the appropriate bias value. At this point, to produce the output shown in **Figure 2**, I used method Helpers.ShowVector to display the current values in the ihSums array. Next, I apply the sigmoid function to each of the values in ihSums and assign the results to array ihOutputs. I'll present the code for method SigmoidFunction shortly. Method ComputeOutputs continues:

```
for (int j = 0; j < numOutput; ++j)
  for (int i = 0; i < numHidden; ++i)
    hoSums[j] += ihOutputs[i] * hoWeights[i][j];
for (int i = 0; i < numOutput; ++i)
  hoSums[i] += hoBiases[i];
...
```

I use the just-computed values in ihOutputs and the weights in hoWeights to compute values into hoSums, then I add the appropriate hidden-to-output bias values. Again, to produce the output shown in **Figure 2**, I called Helpers.ShowVector. Method ComputeOutputs finishes:

```
  for (int i = 0; i < numOutput; ++i)
    this.outputs[i] = HyperTanFunction(hoSums[i]);
  double[] result = new double[numOutput];
  this.outputs.CopyTo(result, 0);
  return result;
}
```

I apply method HyperTanFunction to the hoSums to generate the final outputs into class array private member outputs. I copy those outputs to a local result array and use that array as a return value. An alternative design choice would be to implement ComputeOutputs without a return value, but implement a public method GetOutputs so that the outputs of the neural network object could be retrieved.

The Activation Functions and Helper Methods

Here's the code for the sigmoid function used to compute the input-to-hidden outputs:

```
private static double SigmoidFunction(double x)
{
  if (x < -45.0) return 0.0;
  else if (x > 45.0) return 1.0;
  else return 1.0 / (1.0 + Math.Exp(-x));
}
```

Because some implementations of the Math.Exp function can produce arithmetic overflow, checking the value of the input parameter is usually performed. The code for the tanh function used to compute the hidden-to-output results is:

```
private static double HyperTanFunction(double x)
{
  if (x < -10.0) return -1.0;
  else if (x > 10.0) return 1.0;
  else return Math.Tanh(x);
}
```

Figure 6 **The NeuralNetwork Class Constructor**

```
public NeuralNetwork(int numInput, int numHidden, int numOutput)
{
  this.numInput = numInput;
  this.numHidden = numHidden;
  this.numOutput = numOutput;

  inputs = new double[numInput];
  ihWeights = Helpers.MakeMatrix(numInput, numHidden);
  ihSums = new double[numHidden];
  ihBiases = new double[numHidden];
  ihOutputs = new double[numHidden];
  hoWeights = Helpers.MakeMatrix(numHidden, numOutput);
  hoSums = new double[numOutput];
  hoBiases = new double[numOutput];
  outputs = new double[numOutput];
}
```

The hyperbolic tangent function returns values between -1 and +1, so arithmetic overflow is not a problem. Here the input value is checked merely to improve performance.

The static utility methods in class Helpers are just coding conveniences. The MakeMatrix method used to allocate matrices in the NeuralNetwork constructor allocates each row of a matrix implemented as an array of arrays:

```
public static double[][] MakeMatrix(int rows, int cols)
{
  double[][] result = new double[rows][];
  for (int i = 0; i < rows; ++i)
    result[i] = new double[cols];
  return result;
}
```

Methods ShowVector and ShowMatrix display the values in an array or matrix to the console. You can see the code for these two methods in the code download that accompanies this article (available at code.msdn.microsoft.com/mag201205TestRun).

Next Steps

The code presented here should give you a solid basis for understanding and experimenting with neural networks. You might want to examine the effects of using different activation functions and varying the number of inputs, outputs and hidden layer neurons. You can modify the neural network by making it partially connected, where some neurons are not logically connected to neurons in the next layer. The neural network presented in this article has one hidden layer. It's possible to create more complex neural networks that have two or even more hidden layers, and you might want to extend the code presented here to implement such a neural network.

Neural networks can be used to solve a variety of practical problems, including classification problems. In order to solve such problems there are several challenges. For example, you must know how to encode non-numeric data and how to train a neural network to find the best set of weights and biases. I will present an example of using neural networks for classification in a future article. ■

Dr. James McCaffrey *works for Volt Information Sciences Inc., where he manages technical training for software engineers working at Microsoft's Redmond, Wash., campus. He has worked on several Microsoft products including Internet Explorer and MSN Search. He's the author of ".NET Test Automation Recipes" (Apress, 2006), and can be reached at jammc@microsoft.com.*

Thanks to the following Microsoft technical experts for reviewing this article: *Dan Liebling and Anne Loomis Thompson*

```
40, 0); AddEAN13Sup5(page.Elements, 480, 0); AddUPCVersionA(page.Elements, 0, 135); AddUPCVersionASup2(page.Elements, 240, 135); AddUPCVersionASup5(page.Elements, 480, 135); AddEAN8(page.Element
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 - bar
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
vices; 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
subUnordere                                                                                              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
boBox("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

Talk to Me, Part 3: Meet the Therapist

In the first part of this series (msdn.microsoft.com/magazine/hh781028), I built a simple voice-input system over the phone using the Tropo cloud-hosted voice/SMS system. It wasn't too complicated, but it showed how to use the Tropo scripting API, hosted on the Tropo servers, to receive phone calls, present a menu, gather response input and so on.

In the second column (msdn.microsoft.com/magazine/hh852597), I took a step sideways and talked about Feliza, a "chat-bot" in the spirit of the original "ELIZA" program, designed to take user text input and respond to it in a manner similar to what we might hear while reclining on a psychologist's couch. Again, "she" wasn't all that sophisticated, but Feliza got the point across, and more important, demonstrated how easily the system could be extended to get a lot closer to passing the Turing test.

It seems natural, then, to take these two pieces and weld them together: Let Tropo gather the voice or SMS input from the user, feed it to Feliza, let her calculate a deep, thoughtful response, send it back to Tropo and have Tropo in turn feed it back to the user. Unfortunately, a significant disconnect prevents that from being as easy as it sounds. Because we're using the Tropo Scripting API, our Tropo app is hosted on its servers, and Tropo isn't opening up its servers to host an ASP.NET app, much less our custom Feliza binaries (which, as of last column, are just a set of Microsoft .NET Framework DLLs).

Fortunately, Tropo realized that being able to do voice and SMS by themselves wasn't going to really cut it among the business-savvy developer crowd, and it offers the same kind of voice/SMS access, but over HTTP/REST-like channels. In other words, Tropo will take the incoming voice or SMS input, pass it to a URL of your choice, then capture the response and … well, do whatever the response tells it to (see **Figure 1**).

True, this adds another layer of network communication to the whole system, with all the failover and performance concerns that another network round-trip entails. But it also means that we can capture the input and store it on any server of our choice, which could very well be a significant concern for certain applications—security, database access and so on.

So let's take another step sideways and figure out how Tropo does this little HTTP dance.

Hello, Tropo ... from My Domain

The place to begin is with a simple "Hello world"-style access. Tropo, like many Internet APIs, uses HTTP as the communication channel and JSON as the serialized format of the data being sent.

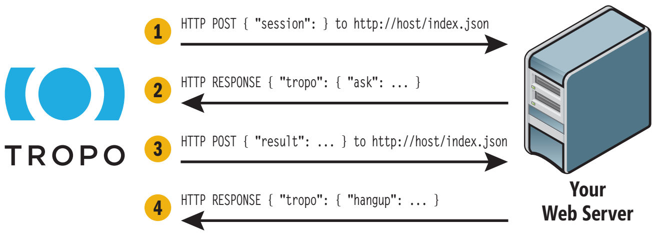

1 HTTP POST { "session": } to http://host/index.json
2 HTTP RESPONSE { "tropo": { "ask": ... }
3 HTTP POST { "result": ... } to http://host/index.json
4 HTTP RESPONSE { "tropo": { "hangup": ... }
TROPO
Your Web Server

Figure 1 **Tropo-Hosted API Call Flow**

So the easiest thing to do is build a simple, static JSON object for Tropo to request when a phone number is called, saying "Hello" to the caller. The JSON for doing that looks like this:

```
{
  "tropo": [
    {
      "say": {
        "value":"Hello, Tropo, from my host!"
      }
    }
  ]
}
```

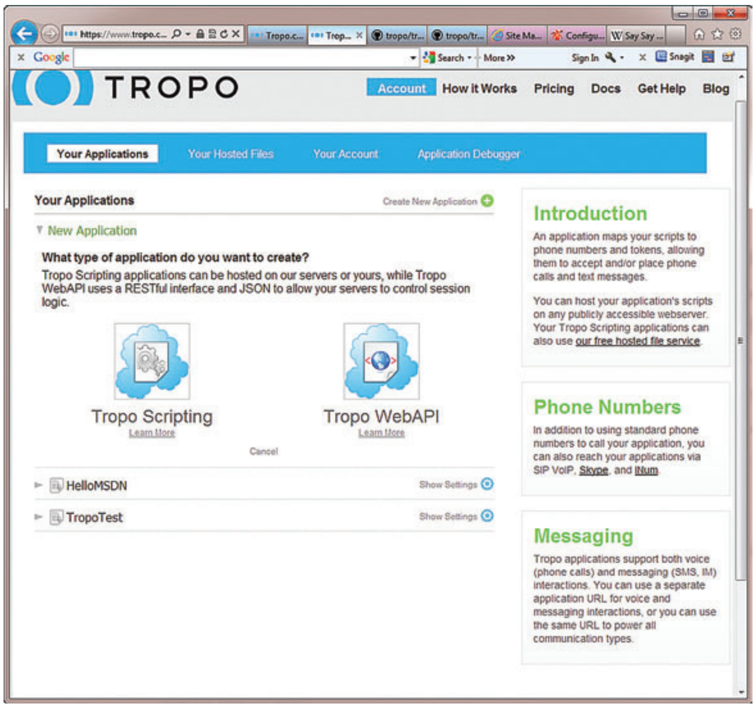
On the surface, the structure is fairly simple. The JSON object is a single-field object, the field "tropo" storing an array of objects that each tell Tropo what to do; in this case, it's a single "say" command, using the Tropo text-to-speech engine to say, "Hello, Tropo, from my host!" But Tropo needs to know how to find this JSON object, which means we need to create and configure a new Tropo application, and we need a server that Tropo can find (meaning it probably can't be a developer laptop hiding behind a firewall). That second point is easily fixed via a quick trip to your favorite ASP.NET hosting provider (I used WinHost—its Basic plan is perfect for this). The first requires a trip back to the Tropo control panel.

This time, when creating a new application, choose "Tropo WebAPI" instead of "Tropo Scripting" (see **Figure 2**), and give it the URL by which to find that particular JSON file; in my case, I created feliza.org (in anticipation of the steps after this) and dropped it off the root of the site. Fully configured, it looks like **Figure 3**.

Although Tropo was happy to hook up a Skype and Session Initiation Protocol (SIP) number for us, we still have to hook up a standard phone number manually. I did that while you weren't looking, and that number is 425-247-3096, in case you want to take a moment and dial the server.

And that's it! Sort of.

If you've been building your own Tropo service alongside me, you're not getting any kind of response from the phone when dialing in. When this is the case, Tropo provides an Application



Your Applications | Your Hosted Files | Your Account | Application Debugger

Your Applications — Create New Application

New Application

What type of application do you want to create?
Tropo Scripting applications can be hosted on our servers or yours, while Tropo WebAPI uses a RESTful interface and JSON to allow your servers to control session logic.

Tropo Scripting — Learn More | Tropo WebAPI — Learn More | Cancel

HelloMSDN — Show Settings
TropoTest — Show Settings

Introduction
An application maps your scripts to phone numbers and tokens, allowing them to accept and/or place phone calls and text messages.

You can host your application's scripts on any publicly accessible webserver. Your Tropo Scripting applications can also use our free hosted file service.

Phone Numbers
In addition to using standard phone numbers to call your application, you can also reach your applications via SIP VoIP, Skype, and INum.

Messaging
Tropo applications support both voice (phone calls) and messaging (SMS, IM) interactions. You can use a separate application URL for voice and messaging interactions, or you can use the same URL to power all communication types.

Figure 2 **The Application Wizard**

Debugger to be able to see the logs from your Tropo app. (Look in the blue bar at the top of the page.) When looking at the log, we see something like the following: "Received non-2XX status code on Tropo-Thread-8d60bf40bc3409843b52f30f929f641c [url=http://www.feliza.org/helloworld.json, code=405]."

Yep, Tropo got an HTTP error. Specifically, it got a "405" error, which (for those who haven't memorized the HTTP spec yet) translates to "Method not supported."

To be honest, calling Tropo a REST service is something of a misnomer, because it doesn't really follow one of the cardinal rules of REST: the HTTP verb describes the action on the resource. Tropo doesn't really care about the verb; it just POSTs everything. And that's why the host is responding (correctly) to the HTTP POST request, because a static page isn't POSTable. Oy.

Fortunately, we know a technology that fixes that pretty easily. At this point, we create an ASP.NET app (an Empty one is fine), and give it a routing that takes "/helloworld.json" and maps it to a simple Controller, as shown in the following code (with much non-relevant code omitted):

```
namespace TropoApp
{
  public class MvcApplication : System.Web.HttpApplication
  {
    public static void RegisterRoutes(RouteCollection routes)
    {
      routes.MapRoute("HelloWorld", "helloworld.json",
        new { controller = "HelloWorld", action = "Index" });
    }
  }
}
```

… which in turn just returns the static JSON for our HelloWorld, as shown here (with much non-relevant code omitted):

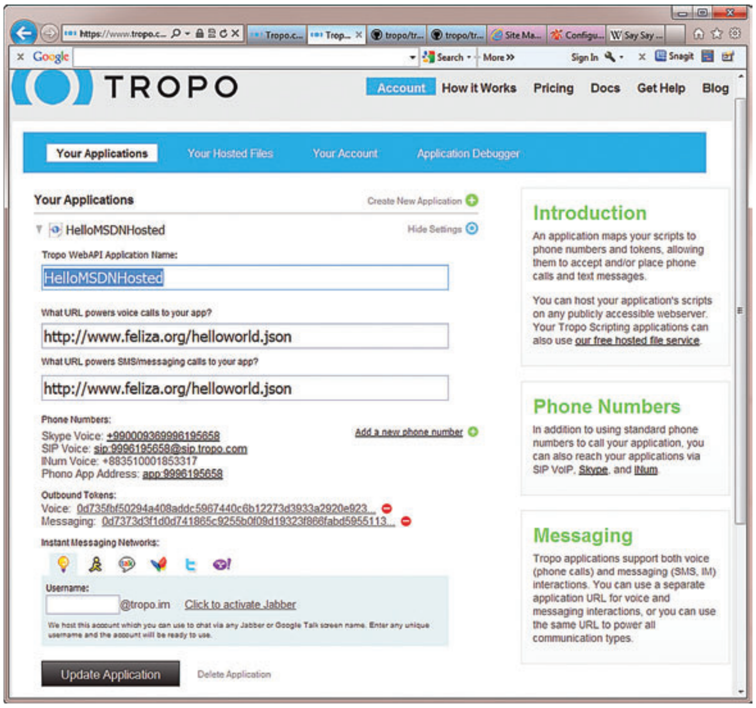
```
namespace TropoApp.Controllers
{
  public class HelloWorldController : Controller
  {
    public const string helloworldJSON =
      "{ \"tropo\":[{\"say\":{\"value\":\"Hello, Tropo,\" +
      " from my host!\"}}]}";

    [AcceptVerbs("GET", "POST")]
    public string Index() {
      return helloworldJSON;
    }
  }
}
```

Push this up to the server, and we're golden.

Say, Say, Say …

If the "say" in the JSON tickles your memory just a little bit, it's because we ran into it during the earlier exploration of the Tropo Scripting API. Back then, it was a method we called, passing in a series of name/value pairs (in true JavaScript fashion) of parameters describing how to customize the spoken output. Here, because we don't have the ability to call APIs on the server—remember, this JSON file is hosted on my server, not the Tropo cloud—we have to describe it in a structural form instead. So, if we want a different voice talking to the user, we need to specify that as a field in the "say" object:

```
{
  "tropo":[
    {
      "say":
      {
        "value":"Hello, Tropo, from my host!",
        "voice":"Grace"
      }
    }
  ]
}
```



Your Applications | Your Hosted Files | Your Account | Application Debugger

Your Applications — Create New Application

HelloMSDNHosted — Hide Settings

Tropo WebAPI Application Name:
HelloMSDNHosted

What URL powers voice calls to your app?
http://www.feliza.org/helloworld.json

What URL powers SMS/messaging calls to your app?
http://www.feliza.org/helloworld.json

Phone Numbers: — Add a new phone number
Skype Voice: +990009369996195658
SIP Voice: sip:9996195658@sip.tropo.com
INum Voice: +883510001853317
Phono App Address: app:9996195658

Outbound Tokens:
Voice: 0d735fbf50294a408addc5967440c6b12273d3933a2920e923...
Messaging: 0d7373d3f1d0d741865c9255b0f09d19323f868fabd5955113...

Instant Messaging Networks:
Username: @tropo.im Click to activate Jabber
We host this account which you can use to chat via any Jabber or Google Talk screen name. Enter any unique username and the account will be ready to use.

Update Application | Delete Application

Introduction
An application maps your scripts to phone numbers and tokens, allowing them to accept and/or place phone calls and text messages.

You can host your application's scripts on any publicly accessible webserver. Your Tropo Scripting applications can also use our free hosted file service.

Phone Numbers
In addition to using standard phone numbers to call your application, you can also reach your applications via SIP VoIP, Skype, and INum.

Messaging
Tropo applications support both voice (phone calls) and messaging (SMS, IM) interactions. You can use a separate application URL for voice and messaging interactions, or you can use the same URL to power all communication types.

Figure 3 **The Configured Application**

Figure 4 **Using .NET Framework Object-to-JSON Bindings**

```
public static object helloworld =
  new { tropo =
    new[] {
      new {
        say = new {
          value = "Hello, Tropo, from my host!",
          voice = "Grace"
        }
      }
    }
};

[AcceptVerbs("POST")]
public JsonResult Index()
{
  return Json(helloworld);
}
```

Now, Grace (who's described as "Australian English") will greet us in the name of Tropo. The full details of "say" are described in the Tropo API docs on its Web site, as are all the JSON objects being passed back and forth.

Here's where using ASP.NET really shines: Rather than try to build up these strings of JSON in the code, we can use the implicit object-to-JSON bindings in ASP.NET to make it easy to slam out these JSON objects (see **Figure 4**).

The JSON sent must have its fields and values quoted using the double quotes, as opposed to the normal JavaScript "it can be either" single quote or double quote. Using the object-to-JSON bindings makes all of that entirely irrelevant to the application developer. Nice. (Note: Tropo also provides a client library for C# that abstracts away much of the JSON stuff, but I'm focusing on REST calls "by hand" because this also helps show how to do the same kind of thing with ASP.NET MVC in general—see bit.ly/bMMJDv for details.)

Calling Tropo a REST service is something of a misnomer.

Listen to the Sound ...

The point of Feliza isn't just to spew out random cookie-cutter bits of psychological tripe, though. She needs to hear the user's spoken input, analyze that and then spew out random cookie-cutter bits of psychological tripe. In order to do that, we have to be able to process the incoming POSTed JSON object that Tropo will send us. Doing so is relatively easy, given that it's a JSON object (and described at bit.ly/yV5ect concerning the "ask" structure, which will say something, then pause and wait for input) and that ASP.NET MVC has some nice auto-JSON-to-object bindings for doing this. So, for example, to post a question to the user and have it drive to a different JSON result, we'd want an "ask" like that in **Figure 5** (as seen in the Tropo docs).

As the parameters imply, this "ask" will time out in 30 seconds, then bind the results (which must be five digits) into a parameter called "acctNum" in the subsequent JSON response POSTed back, which will be sent to the "accountDescribe.json" endpoint. If the account number is incomplete, Tropo will POST to "accountIncomplete.json" and so on.

Figure 5 **An "ask" Example**

```
{
  "tropo": [
    {
      "ask": {
        "say": [
          {
            "value": "Please say your account number"
          }
        ],
        "required": true,
        "timeout": 30,
        "name": "acctNum",
        "choices": {
          "value": "[5 DIGITS]"
        }
      }
    },
    {
      "on":{
        "next":"/accountDescribe.json",
        "event":"continue"
      }
    },
    {
      "on":{
        "next":"/accountIncomplete.json",
        "event":"incomplete"
      }
    }
  ]
}
```

There's only one problem with the system as it's written currently: If we change the input type (in the "choices" field) from "[5 DIGITS]" to "[ANY]" (which is what Feliza would want, after all—she wants users to be able to say anything they want), Tropo tells us in the documentation for "ask" that trying to capture "[ANY]" kinds of input over the voice channel is disallowed. That puts the kibosh on using voice to talk to Feliza. In almost any other application scenario, this wouldn't be a problem. Usually voice input will need to be constrained to a small set of inputs, or else we'll need a tremendous amount of accuracy in transforming the speech to text. Tropo can record the voice channel and store it as an MP3 file for offline analysis, but Tropo offers us another alternative for open-ended text input.

ASP.NET Talking to F#

We've wired Tropo up to our Web site, but Feliza still sits in her F# DLLs, unconnected. We can now start to wire up the Feliza F# binaries against the incoming input, but that's going to require ASP.NET to talk to F#, an exercise that's relatively simple but not always obvious. The ASP.NET site is also going to need to emit custom JSON responses back, so rather than leave the job half-finished, we'll finish off Feliza next time—and look at some ways to potentially extend the system even further.

Happy coding! ■

TED NEWARD *is an architectural consultant with Neudesic LLC. He's written more than 100 articles, is a C# MVP and INETA speaker and has authored and coauthored a dozen books, including the recently released "Professional F# 2.0" (Wrox, 2010). He consults and mentors regularly. Reach him at ted@tedneward.com if you're interested in having him come work with your team, or read his blog at blogs.tedneward.com.*

THANKS to the following technical expert for reviewing this article: *Adam Kalsey*

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

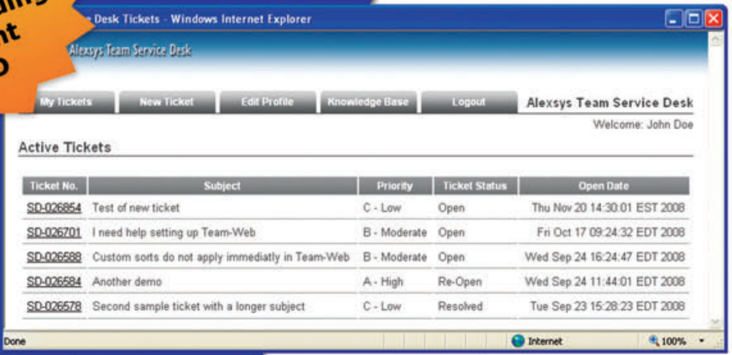
Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD

Alexsys Team



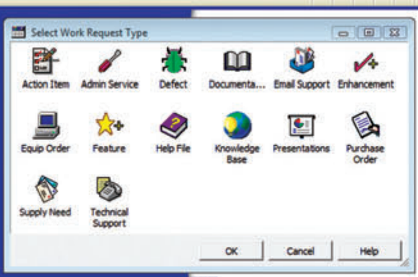
Desk Tickets - Windows Internet Explorer
Alexsys Team Service Desk
My Tickets
New Ticket
Edit Profile
Knowledge Base
Logout
Alexsys Team Service Desk
Welcome: John Doe
Done
Internet
100%

Active Tickets

Ticket No.	Subject	Priority	Ticket Status	Open Date
SD-026854	Test of new ticket	C - Low	Open	Thu Nov 20 14:30:01 EST 2008
SD-026701	I need help setting up Team-Web	B - Moderate	Open	Fri Oct 17 09:24:32 EDT 2008
SD-026588	Custom sorts do not apply immediatly in Team-Web	B - Moderate	Open	Wed Sep 24 16:24:47 EDT 2008
SD-026584	Another demo	A - High	Re-Open	Wed Sep 24 11:44:01 EDT 2008
SD-026578	Second sample ticket with a longer subject	C - Low	Resolved	Tue Sep 23 15:28:23 EDT 2008

New in Team 2.11

- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment



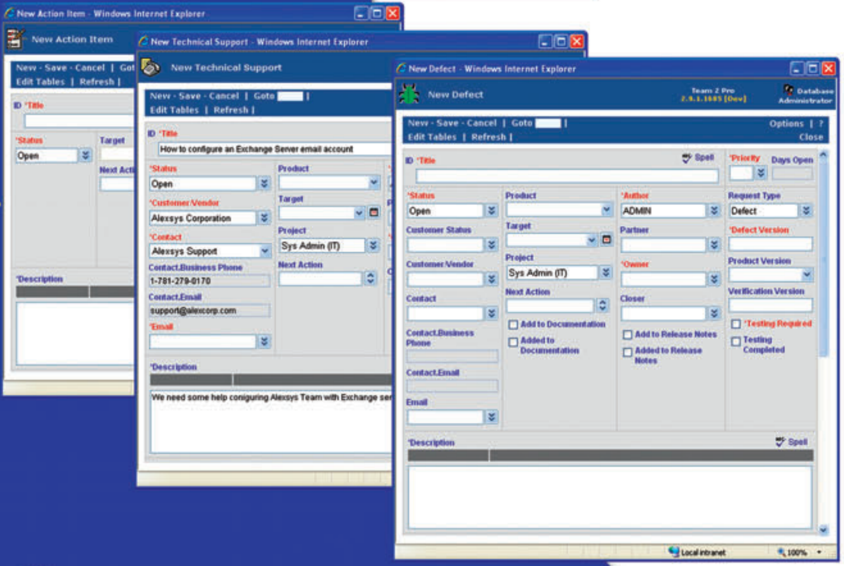
Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration

ZDNet Editors' Pick



New Action Item - Windows Internet Explorer
New Technical Support - Windows Internet Explorer
New Defect - Windows Internet Explorer
New - Save - Cancel | Goto
Edit Tables | Refresh |
Options | ?
Close
Spell

Title: How to configure an Exchange Server email account
Status: Open
Customer/Vendor: Alexsys Corporation
Contact: Alexsys Support
Contact.Business Phone: 1-781-279-0170
Contact.Email: support@alexcorp.com
Project: Sys Admin (IT)
Description: We need some help coniguring Alexsys Team with Exchange ser

Status: Open
Author: ADMIN
Request Type: Defect
Project: Sys Admin (IT)

Free Trial and Single User FreePack™ available at www.alexcorp.com.
FreePack™ includes a free single user Team Pro and Team-Web license.
Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.

Advanced JsRender Templating Features

Templates are powerful, but sometimes you need more than the standard features a templating engine supplies out of the box. You might want to convert data, define a custom helper function or create your own tag. The good news is that you can use the core features of JsRender to do all of this and more.

My April column (msdn.microsoft.com/magazine/hh882454) explored the fundamental features of the JsRender templating library. This column continues the exploration of JsRender in more scenarios such as rendering external templates, changing context with the {{for}} tag and using complex expressions. I'll also demonstrate how to use some of the more powerful JsRender features, including creating custom tags, converters and context helpers, and allowing custom code. All code samples can be downloaded from code.msdn.microsoft.com/mag201205ClientInsight, and JsRender can be downloaded from bit.ly/ywSoNu.

{{for}} Variations

There are several ways the {{for}} tag can be an ideal solution. In my previous column I demonstrated how the {{for}} tag can help iterate through arrays using a block and how it can iterate through multiple objects at once:

```
<!-- looping {{for}} -->
{{for students}}
{{/for}}

<!--  combo iterators {{for}} -->
{{for teachers students staff}}
{{/for}}
```

Data is rarely flat, which is why diving into and out of object hierarchies is an important feature for templates.

The {{for}} (or any block tag) can be converted from a block tag (with content) to a self-closing tag by replacing the block content with an external template, which you point to declaratively as a *tmpl* property. The tag will then render the external template in place of inline content.

Code download available at code.msdn.microsoft.com/mag201205ClientInsight.

This makes it easy to adopt a modular approach to templates where you can reuse template markup in different places and organize and compose templates:

```
<!--  self closing {{for}} -->
{{for lineItems tmpl="#lineItemsDetailTmpl" /}}
```

Code reuse is one of the big advantages of using templates.

Data is rarely flat, which is why diving into and out of object hierarchies is an important feature for templates. I demonstrated the core techniques to dive into an object hierarchy in my previous column using dot notation and square brackets, but you can also use the {{for}} tag to help reduce code. This becomes more apparent when you have an object structure where you're diving into an object hierarchy and need to render a set of properties from a child object. For example, when rendering a person object's address, you might write the template in the following way, where the "address" term in the path is repeated several times:

```
<div>{{:address.street1}}</div>
<div>{{:address.street2}}</div>
<div>{{:address.city}}, {{:address.state}} {{:address.postalCode}}</div>
```

The {{for}} can make code for rendering an address much simpler by eliminating the need to repeat the address object, as shown here:

```
<!--  "with" {{for}} -->
{{for address}}
  <div>{{:street1}}</div>
  <div>{{:street2}}</div>
  <div>{{:city}}, {{:state}} {{:postalCode}}</div>
{{/for}}
```

The {{for}} is operating on the address property, which is a single object with properties, not an array of objects. If the address is truthy (it contains some non-falsey value), the contents of the {{for}} block will be rendered. The {{for}} also changes the current data context from the person object to the address object; thus it acts like a "with" command that many libraries and languages have. So in the preceding example, the {{for}} tag changes the data context to the address and then renders the contents of the templates once (because there's only one address). If the person doesn't have an address (the address property is null or undefined), the contents won't be rendered at all. This makes the {{for}} block great for containing templates that should only be displayed in certain circumstances. This example (from the file 08-for-variations.html in

Figure 1 **Code for Rendering an External Template**

```
my.utils = (function () {
  var
    formatTemplatePath = function (name) {
      return "/templates/_" + name + ".tmpl.html";
    },
    renderTemplate = function (tmplName, targetSelector, data) {
      var file = formatTemplatePath(tmplName);
      $.get(file, null, function (template) {
        var tmpl = $.templates(template);
        var htmlString = tmpl.render(data);
        if (targetSelector) {
          $(targetSelector).html(htmlString);
        }
        return htmlString;
          });
        };
    return {
      formatTemplatePath: formatTemplatePath,
        renderExternalTemplate: renderTemplate
    };
})()
```

the accompanying code download) demonstrates how the sample uses the {{for}} to display pricing info if it exists:

```
{{for pricing}}
  <div class="text">${{:salePrice}}</div>
  {{if fullPrice !== salePrice}}
    <div class="text highlightText">PRICED TO SELL!</div>
  {{/if}}
{{/for}}
```

External Templates

Code reuse is one of the big advantages of using templates. If a template is defined inside of a <script> tag in the same page that it's used, then the template isn't as reusable as it could be. Templates that should be accessible from multiple pages can be created in their own files and retrieved as needed. JavaScript and jQuery make it easy to retrieve the template from an external file, and JsRender makes it easy to render it.

One convention I like to use with external templates is to prefix the file name with an underscore, which is a common naming convention for partial views. I also prefer to suffix all template files with .tmpl.html. The .tmpl denotes that it's a template and the .html extension simply makes it easier for development tools such as Visual Studio to recognize that the template contains HTML. **Figure 1** shows the rendering of an external template.

One way to retrieve the template from an external file is to write a utility function that the JavaScript in a Web app can call. Notice in **Figure 1** that the renderExternalTemplate function on the my.utils object first retrieves the template using the $.get function. When the call completes, the JsRender template is created using the $.templates function from the contents of the response. Finally, the template is rendered using the template's render function and the resulting HTML is displayed in the target. This code could be called using the following code where the template name, the DOM target and the data context are passed to the custom renderExternalTemplates function:

```
my.utils.renderExternalTemplate("medMovie", "#movieContainer", my.vm);
```

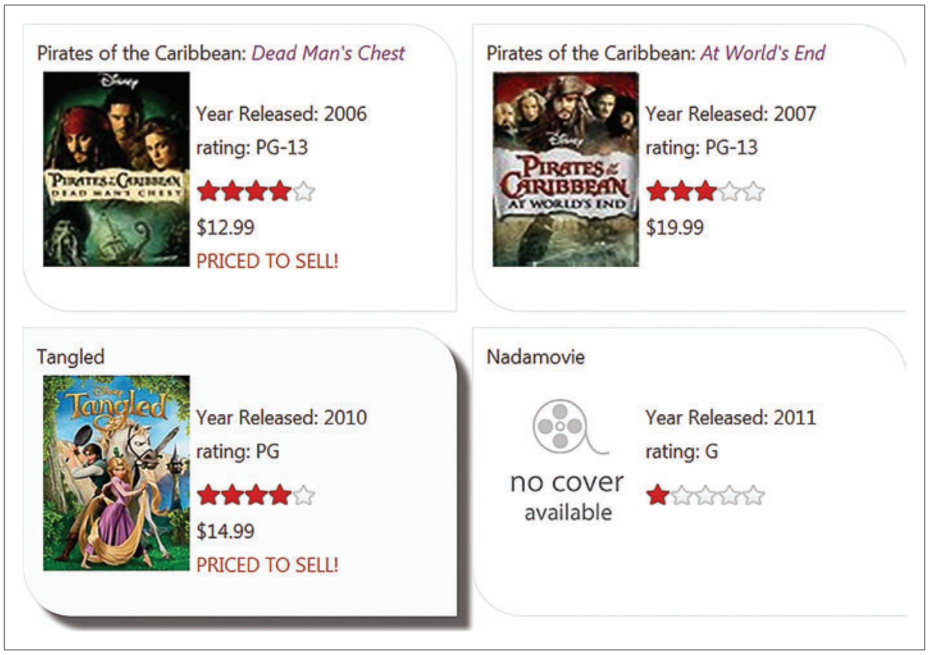
The external template for this sample is in the _medMovie.tmpl.html sample file and contains just the HTML and JsRender tags. It isn't wrapped with a <script> tag. I prefer this technique for external templates because the development environment will recognize that the contents are HTML, which makes writing the code less error prone because IntelliSense works out of the box. However, the file could contain multiple templates, with each template being wrapped in a <script> tag and given an id to uniquely identify it. This is just another way to handle external templates. The final result is shown in **Figure 2**.

JsRender supports expression evaluation but not assignment of the expression, nor the running of random code.

View Paths

JsRender provides several special view paths that make it easy to access the current view object. #view provides access to the current view, #data provides access to the current data context for the view, #parent walks up the object hierarchy and #index returns an index property:

```
<div>{{:#data.section}}</div>
<div>{{:#parent.parent.data.number}}</div>
<div>{{:#parent.parent.parent.parent.data.name}}</div>
<div>{{:#view.data.section}}</div>
```



Pirates of the Caribbean: *Dead Man's Chest*
Year Released: 2006
rating: PG-13
★★★★☆
$12.99
PRICED TO SELL!

Pirates of the Caribbean: *At World's End*
Year Released: 2007
rating: PG-13
★★★☆☆
$19.99

Tangled
Year Released: 2010
rating: PG
★★★★☆
$14.99
PRICED TO SELL!

Nadamovie
no cover available
Year Released: 2011
rating: G
★☆☆☆☆

Figure 2 **The Result of Rendering an External Template**

Figure 3 **Common Expressions in JsRender**

Expression	Example	Comments
+	{{ :a + b }}	Addition
-	{{ :a - b }}	Subtraction
*	{{ :a * b }}	Multiplication
/	{{ :a / b }}	Division
||	{{ :a || b }}	Logical *or*
&&	{{ :a && b }}	Logical *and*
!	{{ :!a }}	Negation
? :	{{ :a === 1 ? b * 2: c * 2 }}	Tertiary expression
( )	{{ :(a||-1) + (b||-1) }}	Ordering evaluation using parentheses
%	{{ :a % b }}	Modulus operation
<= and >= and < and >	{{ :a <= b }}	Comparison operations
=== and !==	{{ :a === b }}	Equality and inequality

When using the view paths (other than #view), they're operating on the current view already. In other words, the following are equivalent:

```
#data
#view.data
```

The view paths are helpful when navigating object hierarchies such as customers with orders with order details, or movies in warehouses in storage locations (as shown in the code download sample file 11-view-paths.html).

Expressions

Common expressions are an essential part of logic and can be useful when deciding how to render a template. JsRender provides support for common expressions including (but not limited to) those shown in **Figure 3**.

JsRender supports expression evaluation but not assignment of the expression, nor the running of random code. This prevents expressions that could otherwise perform variable assignments or perform operations such as opening an alert window. The intention of expressions is to evaluate an expression and either render the result, take action based on the result or use the result in another operation.

For example, performing {{:a++}} with JsRender would result in an error because it attempts to increment the *a* variable. Also, performing {{:alert('hello')}} results in an error because it tries to call a function, #view.data.alert, which doesn't exist.

Registering Custom Tags

JsRender offers several powerful extensibility points such as custom tags, converters, helper functions and template parameters. The syntax for calling each of these is shown here:

```
{{myConverter:name}}

{{myTag name}}

{{:~myHelper(name)}}

{{:~myParameter}}
```

Each of these serves different purposes; however, they can overlap a bit depending on the situation. Before showing how to choose between them, it's important to understand what each one does and how to define them.

Custom tags are ideal when something needs to be rendered that has "control-like" features and can be self-contained. For example, star ratings could be rendered simply as a number using data, like this:

```
{{:rating}}
```

However, it might be better to use JavaScript logic to render the star ratings using CSS and a series of empty and filled star images:

```
{{createStars averageRating max=5/}}
```

The logic for creating the stars could (and should) be separated from the presentation. JsRender provides a way to create a custom tag that wraps this functionality. The code in **Figure 4** defines a custom tag named createStars and registers it with JsRender so it can be used in any page that loads this script. Using this custom tag requires that its JavaScript file, jsrender.tag.js in the sample code, is included in the page.

Custom tags are ideal when something needs to be rendered that has "control-like" features and can be self-contained.

Custom tags can have declarative properties such as the max=5 property of {{createStars}} shown earlier. They're accessed in the code through this.props. For example, the following code registers a custom tag named sort that accepts an array (if the property named reverse is set to true, {{sort array reverse=true/}}, the array is returned in reverse order):

```
$.views.tags({
sort: function(array){
  var ret = "";
  if (this.props.reverse) {
    for (var i = array.length; i; i--) {
      ret += this.tmpl.render(array[i - 1]);
    }
  } else {
      ret += this.tmpl.render(array);
  }
  return ret;
}}
```

Figure 4 **Creating a Custom Tag**

```
$.views.tags({
  createStars: function (rating) {
    var ratingArray = [], defaultMax = 5;
    var max = this.props.max || defaultMax;
    for (var i = 1; i <= max; i++) {
      ratingArray.push(i <= rating ? "rating fullStar" : "rating emptyStar");
    }
    var htmlString = "";
    if (this.tmpl) {
      // Use the content or the template passed in with the template property.
      htmlString = this. renderContent(ratingArray);
    } else {
        // Use the compiled named template.
        htmlString = $.render.compiledRatingTmpl(ratingArray);
    }
    return htmlString;
  }
```

Figure 5 **A Decision Tree to Choose the Right Helper**

```
if (youPlanToReuse) {
  if (simpleConversion && !parameters){
    // Register a converter.
  }
  else if (itFeelsLikeAControl && canBeSelfContained){
    // Register a custom tag.
  }
  else{
    // Register a helper function.
  }
}
else {
  // Pass in a helper function with options for a template.
}
```

A good rule of thumb is to use a custom tag when you need to render something a bit more involved (like a createStars or sort tag) and it could be reused. Custom tags are less ideal for one-off scenarios.

Converters

While custom tags are ideal for creating content, converters are better suited for the simple task of converting a source value to a different value. Converters can change source values (such as a Boolean value of true or false) to something completely different (such as the color green or red, respectively). For example, the following code will use the priceAlert converter to return a string containing a price alert based on the salePrice value:

```
<div class="text highlightText">{{priceAlert:salePrice}}</div>
```

JsRender offers several options to create powerful templates with converters, custom tags and helper functions, but it's important to know under which scenario each should be used.

Converters are great for changing URLs too, as shown here:

```
<img src="{{ensureUrl:boxArt.smallUrl}}" class="rightAlign"/>
```

In the following sample the ensureUrl converter should convert the boxArt.smallUrl value to a qualified URL (both of these converters are used in the file 12-converters.html and are registered in jsrender.helpers.js using the JsRender $.views.converters function):

```
$.views.converters({
  ensureUrl: function (value) {
    return (value ? value : "/images/icon-nocover.png");
  },
  priceAlert: function (value) {0
    return (value < 10 ? "1 Day Special!" : "Sale Price");
  }
});
```

Converters are intended for non-parameterized conversion of data to a rendered value. If the scenario calls for parameters, then a helper function or a custom tag is better suited than a converter. As we've seen previously, custom tags allow for named parameters, so the createStars tag could have parameters to define the size of the stars, their colors, CSS classes to apply to them and so on. The key point here is that converters are for simple conversions, while custom tags are for more involved turnkey rendering.

Figure 6 **Allowing Code in a Template**

```
<script id="myTmpl" type="text/x-jsrender">
  <tr>
    <td>{{:name}}</td>
    <td>
      {{for languages}}
        {{:#data}}{{*
          if ( view.index === view.parent.data.length - 2 ) {
        }} and {{*
          } else if ( view.index < view.parent.data.length - 2 ) {
        }}, {{* } }}
      {{/for}}
    </td>
  </tr>
</script>
```

Helper Functions and Template Parameters

You can pass in helper functions or parameters for use during template rendering in a couple of ways. One is to register them using $.views.helpers, in a similar way to registering tags or converters:

```
$.views.helpers({
  todaysPrices: { unitPrice: 23.40 },
  extPrice:function(unitPrice, qty){
    return unitPrice * qty;
  }
});
```

This will make them available to all templates in the application. Another way is to pass them in as options in the call to render:

```
$.render.myTemplate( data, {
  todaysPrices: { unitPrice: 23.40 },
  extPrice:function(unitPrice, qty){
    return unitPrice * qty;
  }
});
```

This code makes them available only in the context of that particular template-rendering call. Either way, the helpers can be accessed from within the template by prefixing the parameter or function name (or path) with "~":

```
{{: ~extPrice(~todaysPrices.unitPrice, qty) }}
```

Helper functions can do just about anything, including convert data, perform calculations, run app logic, return arrays or objects, or even return a template.

For example, a helper function named getGuitars could be created to search through an array of products and find all guitar products. It might also accept a parameter for the type of guitar. The result could then be used to render a single value or to iterate through the resulting array (because helper functions can return anything). The following code might get an array of all products that are acoustic guitars and iterate on them using a {{for}} block:

```
{{for ~getGuitars('acoustic')}} ... {{/for}}
```

Helper functions can also call other helper functions, such as calculating a total price using an array of an order's line items and applying discount rates and tax rates:

```
{{:~totalPrice(~extendedPrice(lineItems, discount), taxRate}}
```

Helper functions that are accessible to multiple templates are defined by passing an object literal containing the helper functions to the JsRender $.views.helpers function. In the following example, the concat function is defined to concatenate multiple arguments:

```
$.views.helpers({
  concat:function concat() {
    return "".concat.apply( "", arguments );
  }
})
```

The concat helper function can be invoked by using {{:~concat(first, age, last)}}. Assuming the values for first, middle and last are accessible and are John, 25 and Doe, respectively, the value John25Doe would be rendered.

Helper Functions for Unique Scenarios

You might run into a situation where you want to use a helper function for a specific template, but not reuse it in other templates. For example, a shopping cart template might require a calculation that's unique to that template. A helper function could perform the calculation, but there's no need to make it accessible to all templates. JsRender supports this scenario with the second approach mentioned earlier—passing the function in with the options in a render call:

```
$.render.shoppingCartTemplate( data, {
  todaysPrices: { unitPrice: 23.40 },
  extPrice:function(unitPrice, qty){
    return unitPrice * qty;
  }
});
```

In this case the shopping cart template is rendered, and the helper functions and template parameters it needs for its calculation are supplied directly with the render call. The key here is that, in this case, the helper function only exists during the rendering of this specific template.

JsRender allows code to be embedded, but I recommend you do this only when all else fails as the code can be difficult to maintain because it mixes presentation and behavior.

Which to Use?

JsRender offers several options to create powerful templates with converters, custom tags and helper functions, but it's important to know under which scenario each should be used. A good rule of thumb is to use the decision tree shown in **Figure 5**, which outlines how to decide which of these features to use.

If the function is only to be used once, there's no need to create the overhead of making it accessible throughout the entire application. This is the ideal situation for a "one time" helper function that's passed in when needed.

Allow Code

Situations might arise where it's easier to write custom code inside the template. JsRender allows code to be embedded, but I recommend you do this only when all else fails as the code can be difficult to maintain because it mixes presentation and behavior.

Code can be embedded inside a template by wrapping the code with a block prefixed with an asterisk {{* }} and setting allowCode to true. For example, the template named myTmpl (shown in **Figure 6**) embeds code to evaluate the appropriate places to render a command or the word "and" in a series of languages. The full sample can be found in the file 13-allowcode.html. The logic isn't that complicated, yet the code can be difficult to read in the template.

JsRender won't allow the code to be executed unless the allowCode property is set to true (default is false). The following code defines the compiled template named movieTmpl, assigns it the markup from the script tag shown in **Figure 6** and indicates that it should allowCode in the template:

```
$.templates("movieTmpl", {
  markup: "#myTmpl",
  allowCode: true
});

$("#movieRows").html(
  $.render.movieTmpl(my.vm.movies)
);
```

Once the template is created, it's then rendered. The allowCode feature can lead to code that's difficult to read, and in some cases a helper function can do the job. For instance, the example in **Figure 6** uses the allowCode feature of JsRender to add commas and the word "and" where needed. However, this could also be done by creating a helper function:

```
$.views.helpers({
  languagesSeparator: function () {
    var view = this;
    var text = "";
    if (view.index === view.parent.data.length - 2) {
      text = " and";
    } else if (view.index < view.parent.data.length - 2) {
      text = ",";
    }
    return text;
  }
})
```

This languagesSeparator helper function is called by prefixing its name with "~." This makes the template code that calls the helper much easier to read, as shown here:

```
{{for languages}}
  {{:#data}}{{:~languagesSeparator()}}
{{/for}}
```

Moving the logic to a helper function removed behavior from the template and moved it into JavaScript, which follows good separation patterns.

Performance and Flexibility

JsRender offer a variety of features that go well beyond rendering property values, including support for complex expressions, iterating and changing context using the {{for}} tag and view paths for navigating context. It also provides the means to extend its features by adding custom tags, converters and helpers as needed. These features and the pure string-based approach to templating help JsRender benefit from great performance and make it very flexible. ■

JOHN PAPA *is a former evangelist for Microsoft on the Silverlight and Windows 8 teams, where he hosted the popular "Silverlight TV" show. He has presented globally at keynotes and sessions for the BUILD, MIX, PDC, TechEd, Visual Studio Live! and DevConnections events. Papa is also a Microsoft Regional Director, a columnist for* Visual Studio Magazine *(Papa's Perspective) and an author of training videos with Pluralsight. Follow him on Twitter at twitter.com/john_papa.*

THANKS to the following technical expert for reviewing this article: *Boris Moore*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

CODE IN THE SUNSHINE!

December 10-14, 2012
Royal Pacific Resort,
Orlando, FL

Visual Studio Live! is back in Orlando, bringing attendees everything they love about this event: hard-hitting, practical .NET Developer training from the industry's best speakers and Microsoft insiders. But this year, there's an extra bonus – THREE extra co-located events that you can attend for free: SharePoint Live!, SQL Server Live! and Cloud & Virtualization Live!. **vslive.com/orlando**

Customize an agenda to suit YOUR needs – for the same low price, you can attend just one Live! event or sample all four co-located conferences.

SharePoint LIVE!
TRAINING FOR COLLABORATION

Build. Develop. Implement. Manage.
Leading-edge knowledge and training for SharePoint administrators, developers, and planners who must customize, deploy and maintain SharePoint Server and SharePoint Foundation to maximize the business value. **splive360.com**

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

Bringing SQL Server to Your World
IT Professionals, DBAs and Developers – across a breadth of experience and organizations – come together for comprehensive education and knowledge share for SQL Server database management, performance tuning and troubleshooting. **sqllive360.com**

Cloud & Virtualization LIVE!
THE FUTURE OF COMPUTING

Cloud and Virtualization for the Real World
The event for IT Professionals, Systems Administrators, Developers and Consultants to develop their skill sets in evaluating, deploying and optimizing cloud and virtual-based environments. **virtlive360.com**

LIVE! 360 IT EVENTS WITH PERSPECTIVE

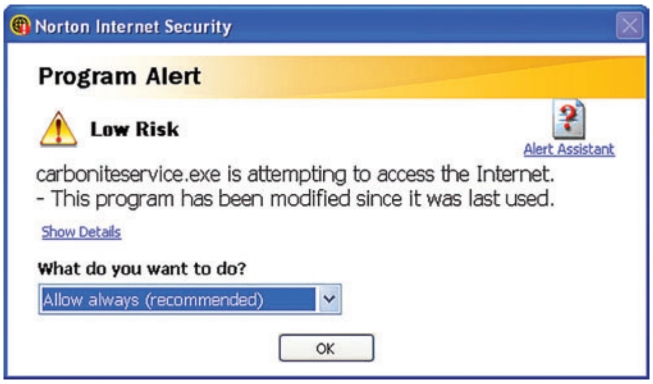
Live360events.com

PRODUCED BY
1105 MEDIA INC

The Myth of Informed Consent

"Dave! Dave, come here! My computer is acting weird again!" I hate when my wife calls me from the other end of the house like that. I just know something bad has come up. Something beyond the capability of my daughters—now 9 and 11—and if a child can't fix it, you know it's serious.

She was reacting to a dialog box displayed by Norton Internet Security. Shown in **Figure 1**, the dialog box read, in part: "carboniteservice.exe is attempting to access the Internet. This program has been modified since it was last used." It then went on to ask if the program should be allowed to access the Internet.


Norton Internet Security

Program Alert

Low Risk

carboniteservice.exe is attempting to access the Internet.
- This program has been modified since it was last used.

What do you want to do?

Allow always (recommended)

Figure 1 **"Low Risk"? Who knows?**

What kind of silliness is this? If all the brainpower at Norton can't figure out whether this application should be allowed to access the Internet, how the hell is my wife ever going to?

For that matter, how would you or I, computer professionals that we claim to be, go about figuring that out? The name of the process means nothing at all. Even if we stipulate that Norton is indicating the correct Carbonite process that we installed, how do we know that Carbonite has been properly updated rather than hijacked by a bad guy, a common attack mode?

We don't, and we shouldn't be asked to. That's why we buy Norton, to access the top brains in the computer security business. Accepting money for a product called "Internet Security" means knowing how to handle these common situations. If the risk is low, then Norton shouldn't be bugging me. And if it's not low, Norton shouldn't be saying it is.

Uninformed Consent

What does Norton think it's doing? I spoke at a conference some time ago, next door to an unrelated computer security meeting. When I slid over during a break to scarf their free beer (we only had juice), I met a guy wearing a Norton badge and jumped on him about this dialog box. He said it makes perfect sense to the company: "We're getting the user's informed consent."

Sorry, that doesn't cut it. Wikipedia defines "informed consent" as consent given "… based upon a clear appreciation and understanding of the facts, implications and future consequences of an action." Ordinary users can't do this, and neither can computer professionals who are not security specialists. Informed consent is impossible in this type of situation.

I opened myself another beer and handed one to the Norton guy, as his meeting was paying for them. He wasn't giving up. "It's like the doctor, who tells you the risks and lets you decide," he said.

No it isn't. Norton throwing this box in a user's face is like an airline asking a passenger if he thinks the weather is safe for flying. The passenger is not competent to make such a judgment. That decision rests entirely on trained and licensed professionals who hold responsibility for transporting passengers safely. That model works well for air travel (zero fatalities on mainline U.S. carriers in the last decade, see bit.ly/GF0cs1), and we should be working the same way.

The main reason I think we're seeing this box is lawyers. Norton's lawyers told the developers, in effect, "If you're not sure, then just ask the user, and you're off the hook. Then if it breaks, it's the user's own fault."

Not to my mind, it isn't. If I were on a jury and the defense tried using this excuse in a trial, I'd not only throw the defendant in jail, I'd add extra punishment for weaseling instead of standing straight and saying, "Sorry, we messed up, here's how we'll fix it." He's probably the kind of guy who refers to bugs as issues. (See "Weasel Words" in the September 2010 issue: msdn.microsoft.com/magazine/ff955613.)

We developers are the experts, and users depend on us. We cannot abdicate our responsibility by asking for guidance from someone who cannot possibly know. Informed consent in computing is a myth, and companies that claim it as an excuse for their malpractice are weasels. Stop it. Now. ■

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. Contact him at rollthunder.com.*

GrapeCity PowerTools

Reporting at its best!

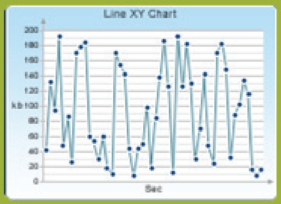
Over 15 years experience in creating the leading .NET Reporting Tool for Silverlight, Windows Forms, ASP.NET, and Windows Azure.

"I've used ActiveReports for a long time...It was easy to use right out of the box, really easy to configure, easy to pull data into.''

Todd Miranda
President
NxtDimension Solutions

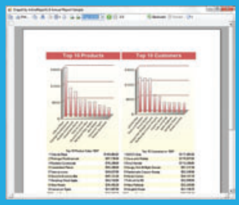
"We can actually handle thousands and thousands of customers and build very, very large hundred-page reports."

Kevin Grohoske
Director of Software Engineering
Bit-Wizards



"It just works!"

Carl Franklin
Host, .NET Rocks!



"For all of our Windows applications we always used Active Reports. It's easy, friendly and simple to use for developers."

Dave Noderer
CEO
Computer Ways, Inc.

"There's nothing like it right now in the market.''

Dani Diaz
Developer Evangelist
Microsoft

Download your FREE Trial Today!

GCPowerTools.com/ActiveReports

ActiveReports

GCPowerTools.com • GvTv.GCPowerTools.com