THE MICROSOFT JOURNAL FOR DEVELOPERS

**FEBRUARY 2014** VOL 29 NO 2

# msdn magazine

Exploring the .NET Framework 4.5.1........32

## Create mobile apps with HTML5, JavaScript and Visual Studio

**DevExtreme Mobile** is a single page application (SPA) framework for your next Windows Phone, iOS and Android application, ready for online publication or packaged as a store-ready native app using Apache Cordova (PhoneGap). With DevExtreme, you can target today's most popular mobile devices with a single codebase and create interactive solutions that will amaze.

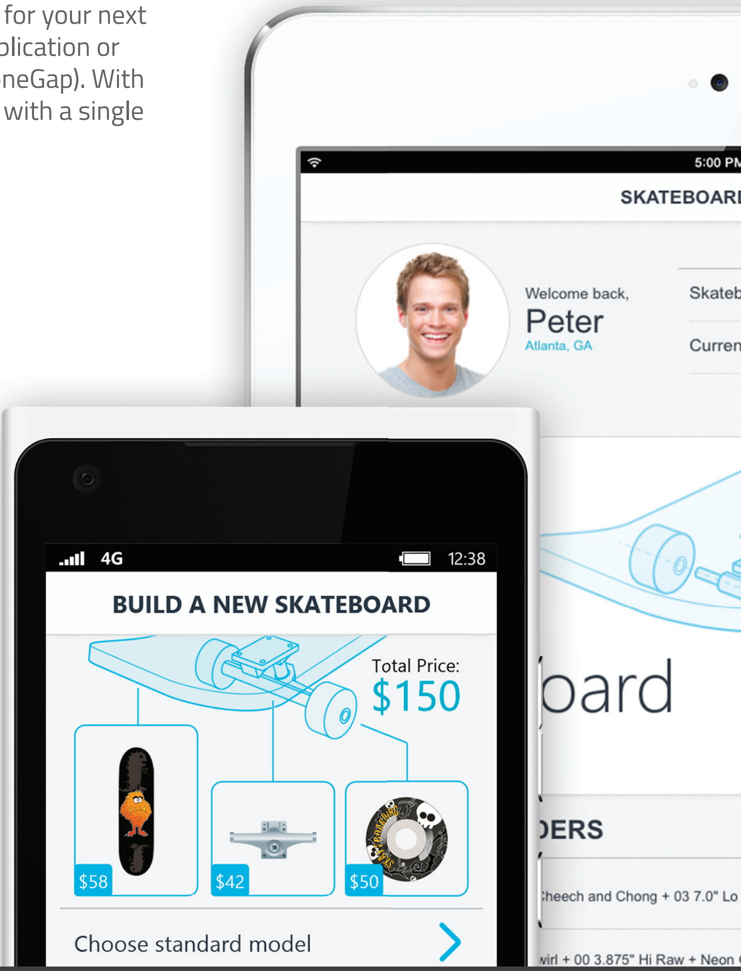
Get started today...

- Leverage your existing Visual Studio expertise.
- Build a real app, not just a web page.
- Deliver a native UI and experience on all supported devices.
- Use over 30 built-in touch optimized widgets.

HTML CSS JS

Learn more and download your free trial

devexpress.com/mobile



DevExpress™

| Rating | Status | Amount |
|---|---|---|
| Hot | Open | $270 |
| Hot | Open | $816 |
|  | Open | $3,658 |
|  | Open | $835 |
|  | Open | $2,828 |
|  | Open | $4,931 |
|  | Open | $120 |
|  | Open | $1,123 |
|  | Open | $2,020 |

| TOP 5 CUSTOMERS | SALES AMOUNT |
|---|---|
| Ernst Handel | $14,193.68 |
| HILARIÓN-Abastos | $5,797.28 |
| Folk och fä HB | $3,019.80 |
| LINO-Delicateses | $2,583.75 |
| Suprêmes délices | $1,209.00 |

# Your next great app starts here.

From interactive Desktop applications, to immersive Web and Mobile solutions, development tools built to meet your needs today and ensure your continued success tomorrow. Download your free 30-day trial today and Experience the DevExpress Difference.

DevExpress™

Learn more and download your free trial at **devexpress.com/try**

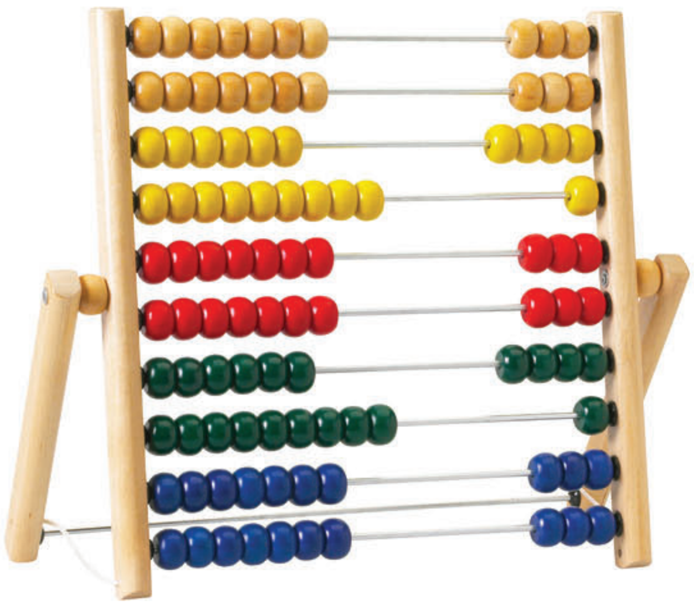
THE MICROSOFT JOURNAL FOR DEVELOPERS

FEBRUARY 2014 VOL 29 NO 2

# msdn magazine

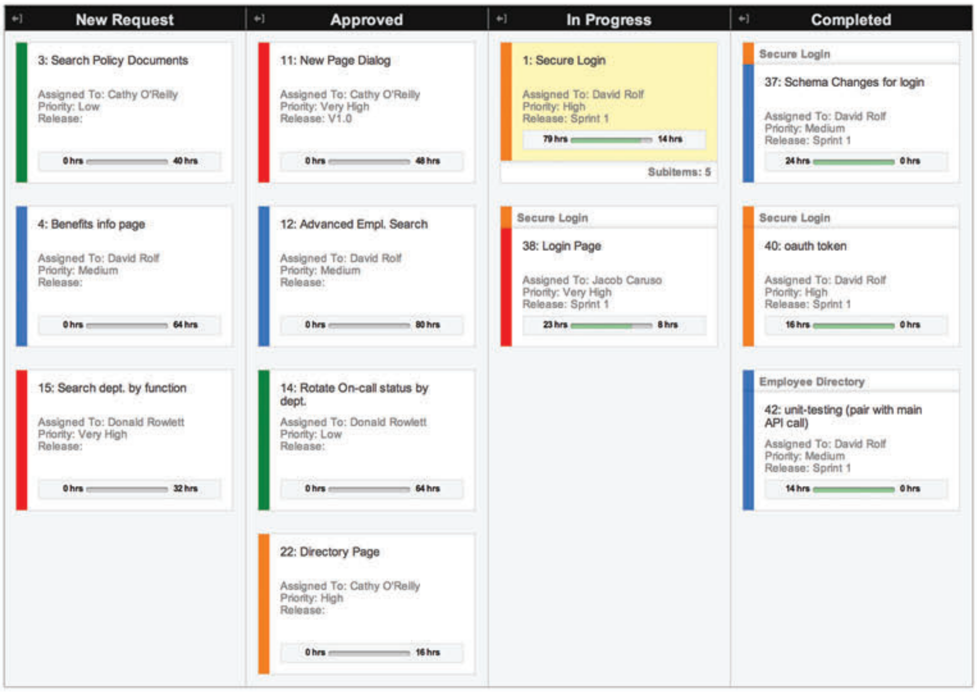
Exploring the .NET Framework 4.5.1........32

Microsoft



VS.

VS.



**New Request**

3: Search Policy Documents
Assigned To: Cathy O'Reilly
Priority: Low
Release:
0 hrs 40 hrs

4: Benefits info page
Assigned To: David Rolf
Priority: Medium
Release:
0 hrs 64 hrs

15: Search dept. by function
Assigned To: Donald Rowlett
Priority: Very High
Release:
0 hrs 32 hrs

**Approved**

11: New Page Dialog
Assigned To: Cathy O'Reilly
Priority: Very High
Release: V1.0
0 hrs 48 hrs

12: Advanced Empl. Search
Assigned To: David Rolf
Priority: Medium
Release:
0 hrs 80 hrs

14: Rotate On-call status by dept.
Assigned To: Donald Rowlett
Priority: Low
Release:
0 hrs 64 hrs

22: Directory Page
Assigned To: Cathy O'Reilly
Priority: High
Release:
0 hrs 16 hrs

**In Progress**

1: Secure Login
Assigned To: David Rolf
Priority: High
Release: Sprint 1
79 hrs 14 hrs
Subitems: 5

Secure Login
38: Login Page
Assigned To: Jacob Caruso
Priority: Very High
Release: Sprint 1
23 hrs 8 hrs

**Completed**

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

Secure Login
40: oauth token
Assigned To: David Rolf
Priority: High
Release: Sprint 1
16 hrs 0 hrs

Employee Directory
42: unit-testing (pair with main API call)
Assigned To: David Rolf
Priority: Medium
Release: Sprint 1
14 hrs 0 hrs

OnTime UPGRADE TO THE #1 SELLING SCRUM SOFTWARE

Technology doesn't have to make things more complicated. Step up your game and learn what real productivity feels like at OnTimeNow.com/MSDN.

dtSearch®

**Instantly Search Terabytes of Text**

25+ fielded and full-text search types

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

Supports databases as well as static and dynamic websites

Highlights hits in all of the above

APIs for .NET, Java, C++, SQL, etc.

64-bit and 32-bit; Win and Linux

"lightning fast" Redmond Magazine

"covers all data sources" eWeek

"results in less than a second" InfoWorld

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

*dtSearch products:*

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

Document filters also available for separate licensing

*Ask about fully-functional evaluations*

The Smart Choice for Text Retrieval® since 1991

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

# msdn magazine

**FEBRUARY 2014** VOLUME 29 NUMBER 2

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

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

**SCOTT SHULTZ** Creative Director
**JOSHUA GOULD** Art Director

**SENIOR CONTRIBUTING EDITOR** Dr. James McCaffrey
**CONTRIBUTING EDITORS** Rachel Appel, Dino Esposito, Kenny Kerr, Julie Lerman, Ted Neward, Charles Petzold, David S. Platt, Bruno Terkaly, Ricardo Villalobos

RedmondMediaGroup

**Henry Allain** President, Redmond Media Group
**Michele Imgrund** Sr. Director of Marketing & Audience Engagement
**Tracy Cook** Director of Online Marketing
**Irene Fincher** Audience Development Manager

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

**Dan LaBianca** Vice President, Group Publisher
**Chris Kourtoglou** Regional Sales Manager
**Danna Vedder** Regional Sales Manager/Microsoft Account Manager
**David Seymour** Director, Print & Online 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.

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

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

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

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

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

Microsoft

BPA WORLDWIDE

Printed in the USA

## DOCUMENT

- *OCR, Barcode & Forms Recognition*
- *PDF Read, Write & Edit*
- *Cleanup and Preprocessing*
- *Annotation and Markup*

## MEDICAL

- *DICOM*
- *PACS*
- *Medical Workstation*
- *Image Processing*

## MULTIMEDIA

- *Playback, Capture & Conversion*
- *MPEG-2 Transport Stream*
- *Distributed Transcoding*
- *DVR*

## IMAGING

- *Raster and Vector Imaging*
- *Viewers*
- *Imaging Processing*
- *150+ Formats*

LEADTOOLS comprehensive line of imaging SDKs have all the document, medical, imaging and multimedia technology you need to develop powerful applications on any platform including .NET, Windows API, WinRT, iOS, OS X, Android, HTML5 and more.

DOWNLOAD OUR 60 DAY EVALUATION
**WWW.LEADTOOLS.COM**

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

SALES@LEADTOOLS.COM
**800.637.1840**

# Conference Calling

When I first got into the technology and IT journalism racket as an editor at *PCWorld* magazine way back in 1992, computer conferences were king. Huge, big-tent confabs like Comdex in Las Vegas and PC Expo in New York drew tens—if not hundreds—of thousands of attendees. Flights were booked solid for days. Cab lines in Vegas snaked around hotel entrances and often forced people to trudge the astonishingly long blocks between venues along the Strip.

The editorial planning for these events was robust. We'd hold multiple full-staff meetings in the weeks ahead of these shows, assigning beats and lining up coverage responsibilities for the team of editors sent to cover keynotes and announcements. It was an era of conference gigantism. Like the dinosaurs that roamed the earth more than 68 million years ago, these industry-wide events were huge, powerful ... and extraordinarily vulnerable.

I began thinking about this when Microsoft announced that it would be holding another Build developer conference this year, returning to the Moscone Center in San Francisco from April 2-4. Build, of course, has its roots in the old Microsoft Professional Developers Conference (PDC) series, which itself launched the same year I arrived at *PCWorld*. No developer conference would ever approach the size and scale of a general computing event like Comdex, but those early PDCs could pack them in. The inaugural 1992 event, which launched the Win32 API and introduced first mention of Windows 95 by its code name "Chicago," was attended by about 5,000 people. Later events would draw 8,000 or more.

There is certainly value in big-tent, destination events—see the ongoing success of giant shows such as the Computer Electronics Show (CES) or CeBIT in Germany—but the IT/computing industry in North America has seen nothing like Comdex or PC Expo since they both waned in the early 2000s. The great lizards of the past have been supplanted by smaller, more nimble mammals better designed to endure a global cold snap and adjust to changing environments. The annual Microsoft TechEd North America conference, considered a large IT/computing gathering, draws an estimated 10,000 or so attendees. Build attendance, meanwhile, is strictly gated. Last year, registration for the Build 2013 conference sold out within hours.

Or consider the Live! 360 DEV conference, which takes place in Las Vegas next month (March 10-14). I've been active with the Visual Studio Live! conference going back to my days as editor in chief of *Visual Studio Magazine*, and now that the event is part of the expanded Live! 360 conference program, I continue to consult with the team to this day. The Live! 360 organizers place an extremely high premium on fostering interaction between presenters and attendees, encouraging people to approach speakers throughout the show. Live! 360 DEV expects to draw 700-plus attendees. It's the absolute antithesis of the Comdex approach and, frankly, it's pretty cool.

In an era of accessible streaming media and limited travel budgets, it really doesn't make sense to drop 100,000 people into a room and call it a conference. Smaller events promise better focus, greater interaction and, ultimately, better value to the developers and attendees traveling to the show.

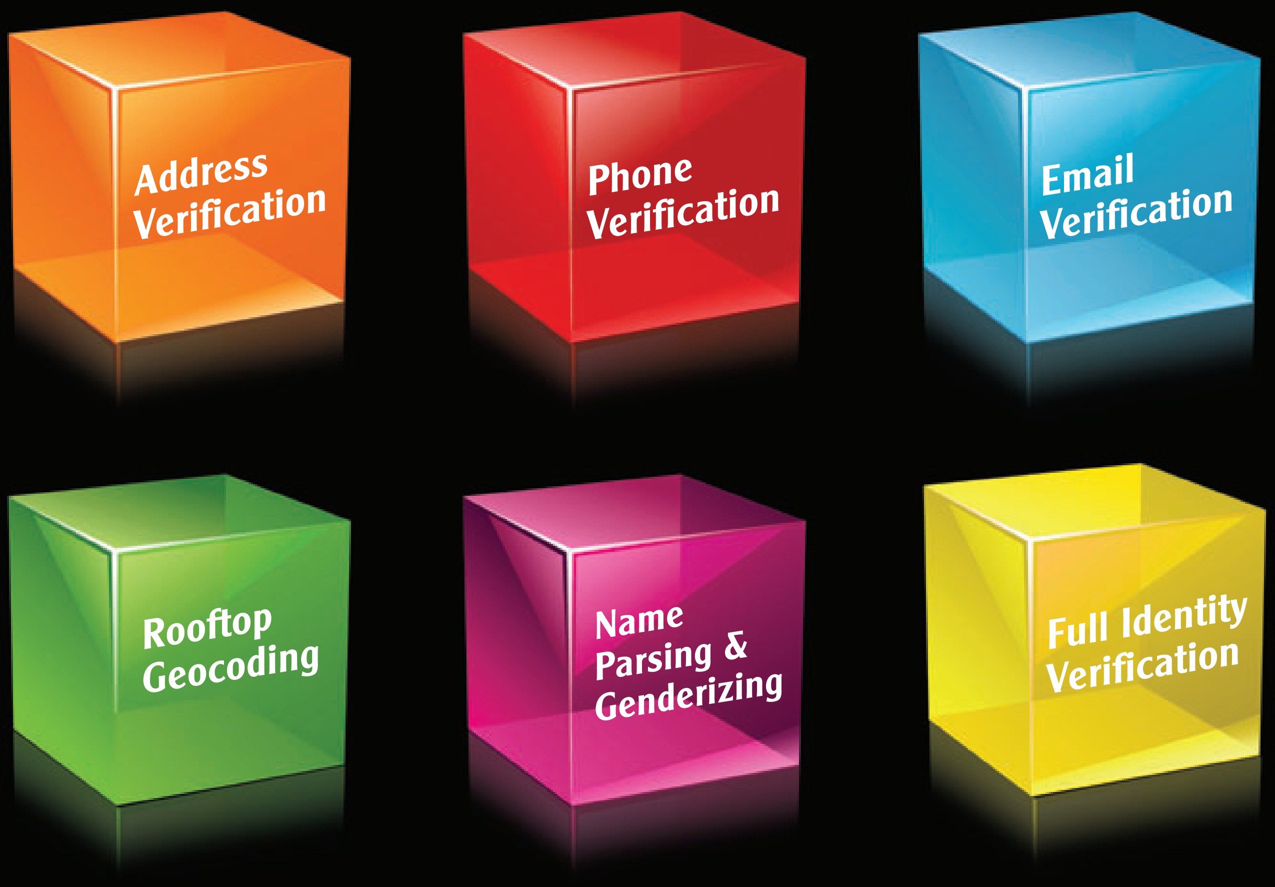
By the time you read this, Build 2014 registration will have long since sold out. If you didn't hit the Web site early on Jan. 14, you're almost certainly out of luck. Enjoy the streaming Web video and congratulate yourself for saving several hundred dollars in frustrating air travel.

For those who did sneak in, I expect you can look forward to an outstanding event. Take advantage of the small footprint. Seek out presenters, ask smart questions and argue with your fellow attendees. There's a lot to get out of these shows, and often the smaller events afford the biggest opportunities to learn and grow. Take advantage of it.

[illegible]

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

## Building Blocks for Global Data Quality Success



### A strong foundation for enterprise data starts with Melissa Data.

Our powerful, scalable data cleansing and integration tools help you profile, cleanse, consolidate, and enrich your contact data. Gain a better understanding of your customer, vendor and supplier data; improve deliverability; increase cost savings; and enhance your operational efficiencies with Melissa Data.

**Advanced Functionalities:**

- Verify, correct, and enhance addresses for 240+ countries
- Add lat/long coordinates to addresses all over the world
- Match and consolidate data to create the golden record
- Append missing contact data like phone numbers and email addresses
- Integrate into many technologies with multiplatform capabilities

**www.MelissaData.com**
**or call 1-800-MELISSA (635-4772)**

MELISSA DATA®
Your Partner in Data Quality

# Content Negotiation and Web API for the ASP.NET MVC Developer

One of the things I like most in ASP.NET MVC is the ability to expose a façade of methods that can be easily invoked from HTTP clients, including jQuery-based pages, mobile apps and plain C# back ends. For a long time, building this service layer took place in the realm of Windows Communication Foundation (WCF) services. Attempts were made to specialize WCF for HTTP, such as the introduction of the webHttpBinding mechanism and frameworks such as the now-retired REST Starter Kit. None of these approaches, though, could really eliminate developer roadblocks such as notorious WCF over-configuration, overuse of attributes and a structure not specifically designed for testability. Then came Web API—a new framework designed to be thin, testable, independent from the hosting environment (for example, IIS) and HTTP-focused.

However, Web API has a programming interface that looks almost too similar to ASP.NET MVC, in my opinion. This isn't a negative remark, though, as ASP.NET MVC has a clean and well-defined programming interface. Web API actually started with a programming model that looked similar to WCF and then grew to resemble ASP.NET MVC.

In this article, I'll provide a view of Web API from the perspective of the average ASP.NET MVC developer, and focus on a functional area of Web API that represents a plus over plain ASP.NET MVC: content negotiation.

## Web API at a Glance

Web API is a framework you can use to create a library of classes that can handle HTTP requests. The resulting library, along with some initial configuration settings, can be hosted in a runtime environment and consumed by callers via HTTP. Public methods on controller classes become HTTP endpoints. Configurable routing rules help define the form of URLs used to access specific methods. With the exception of routing, however, most of what defines the default form of URL handling in Web API is convention rather than configuration.

If you're an ASP.NET MVC developer, at this point you might stop reading and wonder why on earth you'd want to use a new framework that seems to just duplicate the concept of controllers you "already have" in ASP.NET MVC.

The quick answer to that is, yes, you probably don't need Web API in ASP.NET MVC, because you can achieve nearly the same functionality via plain controllers. For example, you can easily return data formatted as JSON or XML strings. You can easily return binary data or plain text. You can shape up the URL templates you like best.

The same controller class can serve JSON data or an HTML view, and you can easily separate controllers that return HTML from controllers that just return data. In fact, a common practice is to have an ApiController class in the project where you stuff all endpoints expected to return plain data. Here's an example:

```
public class ApiController : Controller
{
public ActionResult Customers()
{
  var data = _repository.GetAllCustomers();
  return Json(data, JsonRequestBehavior.AllowGet);
  }
  ...
}
```

Web API uses the best of the ASP.NET MVC architecture and improves it in two main areas. First, it introduces a new logical layer known as content negotiation with a standard set of rules to request data in a given format, whether JSON, XML or some other format. Second, Web API has no dependencies whatsoever on ASP.NET and IIS—more specifically, it has no dependency on the system.web.dll library. Certainly it *can* be hosted in an ASP.NET application under IIS. However, while this probably remains the most common scenario, a Web API library can be hosted in any other application that provides an ad hoc hosting environment, such as a Windows service, a Windows Presentation Foundation (WPF) application or a console application.

At the same time, if you're an expert ASP.NET MVC developer, the Web API concepts of controllers, model binding, routing and action filters will be familiar to you.

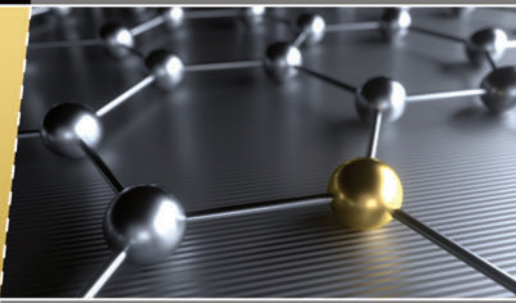
## Why Web Forms Developers Love Web API

If you're an ASP.NET MVC developer, you might be initially confused regarding the benefits of Web API because its programming model looks nearly identical to ASP.NET MVC. However, if you're a Web Forms developer, you shouldn't be confused. With Web API, exposing HTTP endpoints from within a Web Forms application is a child's game. All it takes is adding one or more classes similar to this:

```
public class ValuesController : ApiController
{
  public IEnumerable<string> Get()
  {
    return new string[] { "value1", "value2" };
  }

  public string Get(int id)
  {
    return "value";
  }
}
```

# Empower Your Customers

New! v5.0



## Create & Edit PDFs in .Net - ActiveX - WinRT

- Edit, process and print PDF 1.7 documents
- Create, fill-out and annotate PDF forms
- Fast and lightweight 32- and 64-bit components for .Net and ActiveX/COM
- New WinRT Component enables publishing C#, C++CX or Javascript apps to Windows Store
- New Postscript/EPS to PDF conversion module

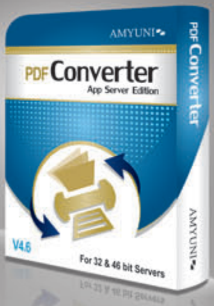
## Complete Suite of Accurate PDF Components

- All your PDF processing, conversion and editing in a single package
- Combines Amyuni PDF Converter and PDF Creator for easy licensing, integration and deployment
- Includes our Microsoft certified PDF Converter printer driver
- Export PDF documents into other formats such as JPeg, PNG, XAML or HTML5

## Advanced HTML to PDF & XAML

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

## High Performance PDF Printer For Desktops and Servers



- Our high-performance printer driver optimized for Web, Application and Print Servers. Print to PDF in a fraction of the time needed with other tools. WHQL tested for Windows 32 and 64-bit including Windows Server 2012 and Windows 8
- Standard PDF features included with a number of unique features. Interface with any .Net or ActiveX programming language
- Easy licensing and deployment to fit system administrator's requirements

CERTIFIED FOR Windows Server 2008 · Windows 7 · Windows 8 · Windows Server 2012 Certified

AMYUNI

All development tools available at

**www.amyuni.com**

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

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

Note that this is the same code you'd use to add a Web API controller to an ASP.NET MVC application. You also have to specify routes. Here's some code you want to run at application startup:

```
RouteTable.Routes.MapHttpRoute(
  name: "DefaultApi",
  routeTemplate: "api/{controller}/{id}",
  defaults: new { id = System.Web.Http.RouteParameter.Optional });
```

Unless otherwise annotated with the NonAction attribute, any public methods on the class that match the default naming and routing conventions are public HTTP-callable endpoints. They can be called from any client without the need for generated proxy classes, web.config references or special code.

Routing conventions in Web API dictate the URL starts with /api followed by the controller name. Note that there's no action name clearly expressed. The action is determined by the type of request, whether GET, PUT, POST or DELETE. A method name that begins with Get, Put, Post or Delete is conventionally mapped to the corresponding action. For example, a method GetTasks on a TaskController will be invoked for any GET request to a URL such as /api/task.

Regardless of the apparent similarity of behavior and class names with ASP.NET MVC, Web API lives in a completely separate set of assemblies and uses a completely different set of types—System.Net.Http is the primary assembly.

## Inside Web API Content Negotiation

"Content negotiation" is often used to describe the process of inspecting the structure of an incoming HTTP request to figure out the formats in which the client wishes to receive responses. Technically, though, content negotiation is the process in which client and server determine the best possible representation format to use in their interactions. Inspecting the request typically means looking into a couple of HTTP headers such as Accept and Content-Type. Content-Type, in particular, is used on the server for processing POST and PUT requests and on the client for choosing the formatter for HTTP responses. Content-Type is not used for GET requests.

The internal machinery of content negotiation, however, is much more sophisticated. The aforementioned scenario is the most typical—because of default conventions and implementations—but it isn't the only one possible.

The component that governs the negotiation process in Web API is the class called DefaultContentNegotiator. It implements a public interface (IContentNegotiator), so you can replace it entirely if needed. Internally, the default negotiator applies several distinct criteria in order to figure out the ideal format for the response.

The negotiator works with a list of registered media type formatters—the components that actually turn objects into a specific format. The negotiator goes through the list of formatters and stops at the first match. A formatter has a couple of ways to let the negotiator know it can serialize the response for the current request.

The first check occurs on the content of the MediaTypeMappings collection, which is empty by default in all predefined media type formatters. A media type mapping indicates a condition that, if verified, entitles the formatter to serialize the response for the ongoing request. There are a few predefined media type mappings. One looks at a particular parameter in the query string. For example, you can enable XML serialization by simply requiring that an xml=true expression is added to the query string used to invoke Web API. For this to happen, you need to have the following code in the constructor of your custom XML media type formatter:

```
MediaTypeMappings.Add(new QueryStringMapping("xml", "true", "text/xml"));
```

In a similar way, you can have callers express their preferences by adding an extension to the URL or by adding a custom HTTP header:

```
MediaTypeMappings.Add(new UriPathExtensionMapping("xml", "text/xml"));
MediaTypeMappings.Add(new RequestHeaderMapping("xml", "true",
  StringComparison.InvariantCultureIgnoreCase, false,"text/xml"));
```

For URL path extension, it means the following URL will map to the XML formatter:

```
http://server/api/news.xml
```

Note that for URL path extensions to work you need to have an ad hoc route such as:

```
config.Routes.MapHttpRoute(
  name: "Url extension",
  routeTemplate: "api/{controller}/{action}.{ext}/{id}",
  defaults: new { id = RouteParameter.Optional }
);
```

For custom HTTP headers, the constructor of the RequestHeaderMapping class accepts the name of the header, its expected value and a couple of extra parameters. One optional parameter indicates the desired string comparison mode, and the other is a Boolean that indicates if the comparison is on the entire string. If the negotiator can't find a match on the formatter using the media type mapping information, it looks at standard HTTP headers such as Accept and Content-Type. If no match is found, it again goes through the list of registered formatters and checks whether the return type of the request can be serialized by one of the formatters.

To add a custom formatter, insert something like the following code in the startup of the application (for example, in the Application_Start method):

```
config.Formatters.Add(xmlIndex, new NewsXmlFormatter());
```

## Customizing the Negotiation Process

Most of the time, media type mappings let you easily fulfill any special requirements for serialization. However, you can always replace the default content negotiator by writing a derived class and overriding the MatchRequestMediaType method:

```
protected override MediaTypeFormatterMatch MatchRequestMediaType(
  HttpRequestMessage request, MediaTypeFormatter formatter)
{
  ...
}
```

You can create a completely custom content negotiator with a new class that implements the IContentNegotiator interface. Once you have a handmade negotiator, you register it with the Web API runtime:

```
GlobalConfiguration.Configuration.Services.Replace(
  typeof(IContentNegotiator),
  new YourOwnNegotiator());
```

The preceding code usually goes in global.asax or in one of those handy config handlers that Visual Studio creates for you in the ASP.NET MVC Web API project template.

## Controlling Content Formatting from the Client

The most common scenario for content negotiation in Web API is when the Accept header is used. This approach makes content formatting completely transparent to your Web API code. The caller sets the Accept header appropriately (for example, to text/xml) and the Web API infrastructure handles it accordingly. The following code shows how to set the Accept header in a jQuery call to a Web API endpoint to get back some XML:

```
$.ajax({
  url: "/api/news/all",
  type: "GET",
  headers: { Accept: "text/xml; charset=utf-8" }
});
```

In C# code, you set the Accept header like this:

```
var client = new HttpClient();
client.Headers.Add("Accept", "text/xml; charset=utf-8");
```

Any HTTP API in any programming environment lets you set HTTP headers. And if you foresee that you can have callers where this might be an issue, a best practice is to also add a media type mapping so the URL contains all the required information about content formatting.

Bear in mind that the response strictly depends on the structure of the HTTP request. Try requesting a Web API URL from the address bar of Internet Explorer 10 and Chrome. Don't be surprised to see you get JSON in one case and XML in the other. The default Accept headers might be different in various browsers. In general, if the API will be publicly used by third parties, you should have a URL-based mechanism to select the output format.

## Scenarios for Using Web API

Architecturally speaking, Web API is a big step forward. It's becoming even more important with the recent Open Web Interface for .NET (OWIN) NuGet package (Microsoft.AspNet.Web-Api.Owin) and Project Katana, which facilitate hosting the API in external apps through a standard set of interfaces. If you're building solutions other than ASP.NET MVC applications, using Web API is a no-brainer. But what's the point of using Web API within a Web solution based on ASP.NET MVC?

With plain ASP.NET MVC, you can easily build an HTTP façade without learning new things. You can negotiate content fairly easily with just a bit of code in some controller base class or in any method that needs it (or by creating a negotiated ActionResult). It's as easy as having an extra parameter in the action method signature, checking it and then serializing the response to XML or JSON accordingly. This solution is practical as long as you limit yourself to using XML or JSON. But if you have more formats to take into account, you'll probably want to use Web API.

As previously mentioned, Web API can be hosted outside of IIS—for example, in a Windows service. Clearly, if the API lives within an ASP.NET MVC application, you're bound to IIS. The type of hosting therefore depends on the goals of the API layer you're creating. If it's meant to be consumed only by the surrounding ASP.NET MVC site, then you probably don't need Web API. If your created API layer is really a "service" for exposing the API of some business context, then Web API used within ASP.NET MVC makes good sense. ■

**DINO ESPOSITO** *is the author of "Architecting Mobile Solutions for the Enterprise" (Microsoft Press, 2012) and the upcoming "Programming ASP.NET MVC 5" (Microsoft Press). A technical evangelist for the .NET and Android platforms at JetBrains and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents.wordpress.com and on Twitter at twitter.com/despos.*

**THANKS** to the following technical expert for reviewing this article: *Howard Dierking (Microsoft)*

WORKING WITH FILES?

DOC PPT XLS PDF MSG

✓ CONVERT
✓ PRINT
✓ CREATE
✓ COMBINE
✓ MODIFY

100% Standalone - No Office Automation

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

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

ASPOSE
Your File Format Experts

SCAN FOR
20% SAVINGS

# ASPOSE.TOTAL

Every Aspose component combined in *ONE* powerful suite!

## Powerful File Format APIs

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

*... and many others!*

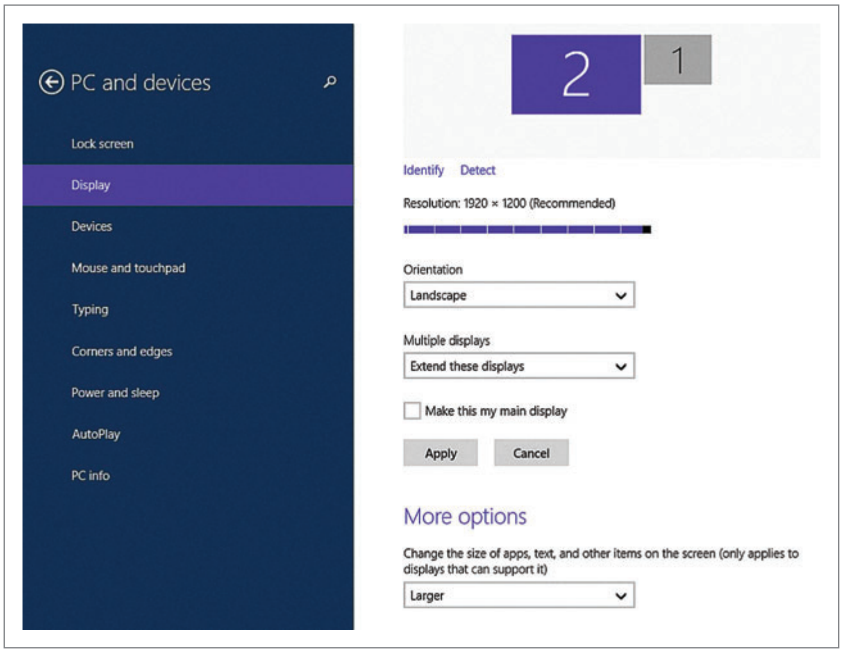
Aspose.Total for .NET
Aspose.Total for Java
Aspose.Total for Cloud
Aspose.Total for Android

Get your FREE evaluation copy at www.aspose.com

.NET Java Cloud Android

# Write High-DPI Apps for Windows 8.1

Windows 8 introduced a new programming model for Windows apps, based on the Windows Runtime (WinRT), which lets users dynamically change the size of screen elements with a PC setting. You can see what this option looks like in the PC settings in **Figure 1**. On my desktop, the options are Default and Larger. On my Surface Pro, the options are Smaller and Default. It really depends on the device, and in particular, the vertical resolution of the attached displays. More important, Windows Store apps receive an event whenever this option is changed and can thus dynamically update their rendering code to reflect the current scaling factor.



Resolution: 1920 × 1200 (Recommended)

Orientation: Landscape

Multiple displays: Extend these displays

☐ Make this my main display

More options

Change the size of apps, text, and other items on the screen (only applies to displays that can support it): Larger

Figure 1 **Windows 8.1 PC Setting Affecting Windows Store Apps**

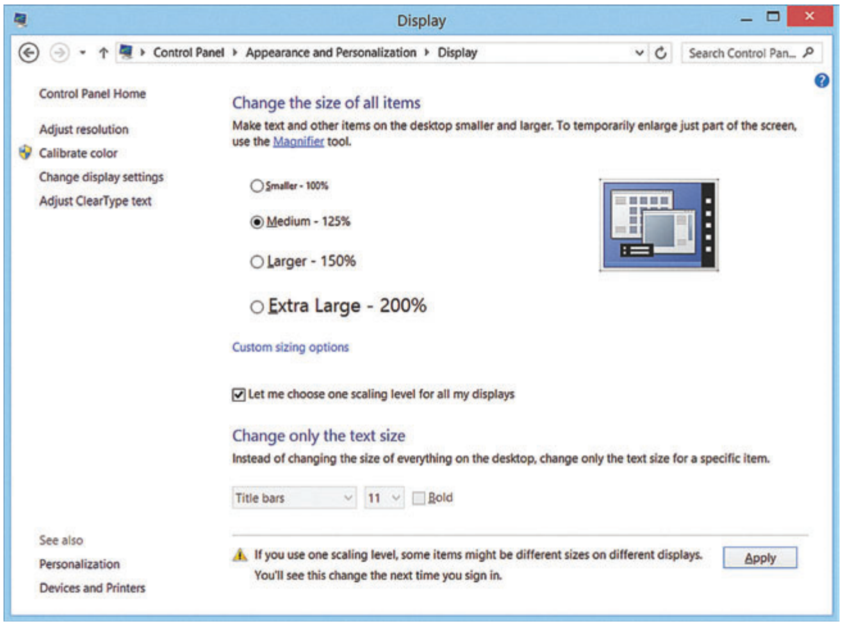
The desktop on Windows 8, however, remained static. Desktop applications continued to be serviced by a system DPI setting, any changes to which would only take effect after the user signs out and back in again, effectively forcing all applications to shut down and restart. You can see this option, which is still available in Windows 8.1, in **Figure 2**, with its more granular Smaller, Medium, Larger and Extra Large options. It reminds me of a trip to the local coffee shop. Here, too, the options that may be available depend on the attached displays. My Surface Pro, for example, only includes Smaller, Medium and Larger.

Needless to say, this split personality—like many things in Windows 8—can be quite confusing for the developer, let alone the user. While Windows 8.1 doesn't really address the confusion in any meaningful way, it does finally allow desktop applications to similarly handle DPI scaling dynamically, thus the user is no longer forced to shut everything down and bring up a new logon session. But Windows 8.1 goes a lot further and really brings new life to multi-monitor configurations.

While the window in **Figure 2** looks quite similar to what was available in Windows 8, it now sports a little checkbox that was added in Windows 8.1. Although it's checked in **Figure 2**, the default is unchecked. The checkbox title, "Let me choose one scaling level for all my displays," hints at the other capability that's new in Windows 8.1: the ability for different monitors to have different scaling factors. The checkbox title is a little confusing, as clearing this checkbox still offers value for users who only have a single monitor. In that case, it still offers the user the option of changing the scaling factor dynamically. Checking this option really represents a legacy or compatibility mode for DPI behavior. So whether you have multiple monitors—or more important, regardless of how many monitors your users might normally use—you're going to want to come to grips with these new options. They'll affect your applications whether you like it or not. Clearing this checkbox reveals the window in **Figure 3**. Again, this is now the default on Windows 8.1.

If you think about it, there's really no difference between **Figure 2** and **Figure 3**. The former uses four radio buttons and the latter uses a slider with four possible positions. The only real difference is that changes to the slider take effect immediately, or at least as soon as you hit the Apply button. This is much the same experience, at least for the user, as the scaling option for Windows Store apps. Changes to the radio button selection, however, only take effect the next time the user signs in.

The four possible values, for either the slider or radio buttons, correspond to four DPI scaling factors and are illustrated in **Figure 4**. As a developer, I caution you not to read too much into the specific DPI values. They're meant to reflect the resolution or


Display

Control Panel ▸ Appearance and Personalization ▸ Display

Search Control Pan...

Control Panel Home

Adjust resolution

Calibrate color

Change display settings

Adjust ClearType text

Change the size of all items

Make text and other items on the desktop smaller and larger. To temporarily enlarge just part of the screen, use the Magnifier tool.

○ Smaller - 100%

◉ Medium - 125%

○ Larger - 150%

○ Extra Large - 200%

Custom sizing options

☑ Let me choose one scaling level for all my displays

Change only the text size

Instead of changing the size of everything on the desktop, change only the text size for a specific item.

Title bars | 11 | ☐ Bold

See also

Personalization

Devices and Printers

If you use one scaling level, some items might be different sizes on different displays. You'll see this change the next time you sign in.

Apply

Figure 2 **Pre-Windows 8.1 PC Setting Affecting Desktop Applications**

pixel density of the screen, but in reality many factors influence the DPI value—such as form factor and distance to the screen—so the effective DPI value you end up using has little to do with an actual inch. These four options also represent the full spectrum of possibilities, but what a particular PC might offer depends on the vertical resolution of its displays.

This is illustrated in **Figure 5**. These limits are intended to keep UI elements from getting cropped off the bottom of the display. If your display has fewer than 900 lines of vertical resolution, then you won't have any options and the 100 percent scaling factor will be all there is. As the vertical resolution of your display increases, you're presented with more options until you reach 1,440 lines of vertical resolution, at which point you experience all four possible options. These options do not, however, affect the scaling on all monitors equally. This is where the concept of per-monitor DPI scaling comes from. It isn't entirely obvious at first glance because the OS takes into account both the vertical resolution as well as the native DPI for the physical display.

As a developer of desktop applications, it's important to realize there are now two scaling factors that may be at play. There's the system DPI scaling factor and then there's a per-monitor DPI scaling factor. The system DPI scaling factor corresponds to one of the values in **Figure 4**—with the exception of devices such as Windows Phone—and remains constant for the duration of the logon session. The system DPI value is based on the initial radio button shown in **Figure 2** or the slider position shown in **Figure 3**.

To retrieve the system DPI value, you start by getting hold of the desktop device context. Yes, this boils down to the old GDI API, but it has nothing to do with GDI rendering and is only a historical footnote. First, to get a handle representing the desktop device context, you call the

It's important to realize there are now two scaling factors that may be at play.

GetDC function with a nullptr instead of a window handle. This special value indicates you want the device context for the desktop as a whole rather than a particular window:

```
auto dc = GetDC(nullptr);
```

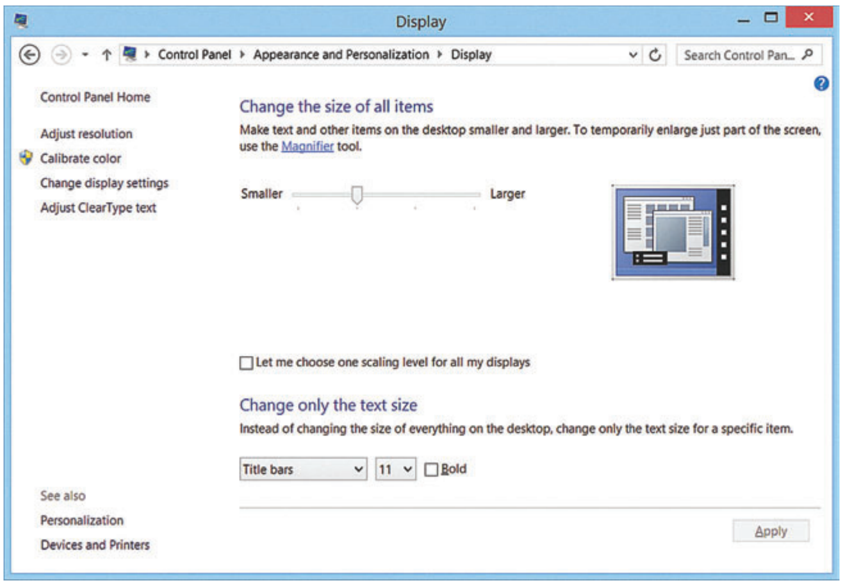
Naturally, you must remember to free this handle when you're done:

```
ReleaseDC(nullptr, dc);
```

The first parameter is the handle to the window to which the device context refers. Again, a nullptr value represents the desktop. Now, given the device context, you can use the GetDeviceCaps function to retrieve the system DPI scaling factor for the x and y axes as follows:

```
auto x = GetDeviceCaps(dc, LOGPIXELSX);
auto y = GetDeviceCaps(dc, LOGPIXELSY);
```

Having a different value for the x and y axes dates back to the dark ages when printers routinely offered different scaling factors horizontally and vertically. I've never come across a display that offers up non-square pixels, but I hear they do exist in some industries for which special graphics cards have been developed. LOGPIXELSX and LOGPIXELSY represent the number of pixels per logical inch along the


Display

Control Panel ▸ Appearance and Personalization ▸ Display

Search Control Pan...

Control Panel Home

Adjust resolution

Calibrate color

Change display settings

Adjust ClearType text

Change the size of all items

Make text and other items on the desktop smaller and larger. To temporarily enlarge just part of the screen, use the Magnifier tool.

Smaller — Larger

☐ Let me choose one scaling level for all my displays

Change only the text size

Instead of changing the size of everything on the desktop, change only the text size for a specific item.

Title bars | 11 | ☐ Bold

See also

Personalization

Devices and Printers

Apply

Figure 3 **Windows 8.1 PC Setting Affecting Dynamic and Per-Monitor Scaling for the Desktop**

Figure 4 **System Scaling Factors**

| DPI | Percentage |
|---|---|
| 96 | 100 |
| 120 | 125 |
| 144 | 150 |
| 192 | 200 |

Figure 5 **Scaling Options Relative to Vertical Resolution**

| Resolution | Scaling Options |
|---|---|
| ... – 900 | 100% |
| 900 – 1079 | 100% – 125% |
| 1080 – 1439 | 100% – 125% – 150% |
| 1440 – ... | 100% – 125% – 150% – 200% |

width and height of the desktop. Again, this is a logical inch and isn't meant to reflect reality. Also, given these are the system DPI values, it means they're the same for all monitors that span the desktop regardless of how relatively big or how small they might be. Therein lies the problem.

If you plug your Dell Venue 8 Pro tablet with its 8-inch screen into an array of 30-inch Dell UltraSharp monitors, you're going to have to make a difficult choice. I routinely plug two or three vastly different monitors into my desktop's graphics card. A system-wide DPI scaling factor just doesn't cut it. What's needed is for each monitor to have a DPI scaling factor ideally suited to its relative size or resolution. This is exactly what Windows 8.1 offers with its per-monitor DPI scaling support.

Windows 8.1 offers up three different levels of DPI awareness. This is obvious when you look at the Process Explorer window, shown in **Figure 6**. You can see some applications are completely DPI-unaware, as in the case of the command prompt. Most applications that were written for Windows 7 and Windows 8 are system DPI-aware, or at least claim to be. Examples include Microsoft Outlook and the calculator. Per-monitor DPI awareness is the third and optimal level of awareness. Examples in **Figure 6** include Internet Explorer and Microsoft Word. Interestingly, Word isn't actually per-monitor DPI-aware at the moment, but I've disabled display scaling for Word to avoid blurriness. This has the effect of overriding the process DPI awareness so the desktop window manager won't scale the window. This was done with a compatibility option.

The main point is you better make sure your apps are per-monitor DPI-aware and scale accordingly. How exactly can you do this? Keep reading.

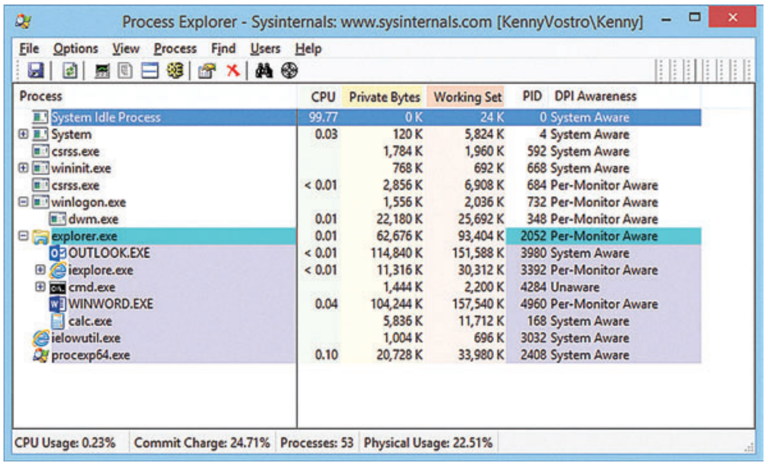
Different applications make various assertions about their level of DPI awareness, and then the desktop window manager—the service responsible for composing the various application windows together—determines how to scale different application windows based on their individual DPI claims such that all windows are presented at some consistent scale. If an application claims to be per-monitor DPI-aware, the desktop window manager won't scale the window at all and assumes the application knows what it's doing. If an application is system DPI-aware, the desktop window manager will scale the window based on the assumption that it was rendered to the system DPI scaling factor returned by the GetDeviceCaps function I mentioned earlier. And if an application is DPI-unaware, the desktop window manager assumes the window was rendered to the traditional 96 DPI and scales it accordingly.

Before I consider how to actually write a well-behaving, per-monitor, DPI-aware application, I'll discuss what it takes to make such a claim. Windows Vista introduced the SetProcessDPIAware function to mark the calling process as DPI-aware. This function took no arguments and simply turned on DPI awareness, so to speak. This was before per-monitor awareness, so this was a simple binary state. Either you're DPI-aware or you're not. Windows 8.1 introduced a new function called SetProcessDpiAwareness that provides more control over this level of awareness:

```
VERIFY_(S_OK, SetProcessDpiAwareness(PROCESS_PER_MONITOR_DPI_AWARE));
```

I routinely plug two or three vastly different monitors into my desktop's graphics card. A system-wide DPI scaling factor just doesn't cut it.

This sets the process's DPI awareness to the given level. Constants are available to cover the three possible states: DPI-unaware, system DPI-aware and per-monitor DPI-aware. There's also a corresponding GetProcessDpiAwareness function to query this value for a given process. But using code to set the awareness level has a number of drawbacks. You need to be careful to call these functions early on in your application's lifetime. Sometimes this isn't practical and causes a number of issues when mixing executables and DLLs. A better solution is called for.


Process Explorer - Sysinternals: www.sysinternals.com [KennyVostro\Kenny]
File Options View Process Find Users Help

| Process | CPU | Private Bytes | Working Set | PID | DPI Awareness |
|---|---|---|---|---|---|
| System Idle Process | 99.77 | 0 K | 24 K | 0 | System Aware |
| System | 0.03 | 120 K | 5,824 K | 4 | System Aware |
| csrss.exe | | 1,784 K | 1,960 K | 592 | System Aware |
| wininit.exe | | 768 K | 692 K | 668 | System Aware |
| csrss.exe | < 0.01 | 2,856 K | 6,908 K | 684 | Per-Monitor Aware |
| winlogon.exe | | 1,556 K | 2,036 K | 732 | Per-Monitor Aware |
| dwm.exe | 0.01 | 22,180 K | 25,692 K | 348 | Per-Monitor Aware |
| explorer.exe | 0.01 | 62,676 K | 93,404 K | 2052 | Per-Monitor Aware |
| OUTLOOK.EXE | < 0.01 | 114,840 K | 151,588 K | 3980 | System Aware |
| iexplore.exe | < 0.01 | 11,316 K | 30,312 K | 3392 | Per-Monitor Aware |
| cmd.exe | | 1,444 K | 2,200 K | 4284 | Unaware |
| WINWORD.EXE | 0.04 | 104,244 K | 157,540 K | 4960 | Per-Monitor Aware |
| calc.exe | | 5,836 K | 11,712 K | 168 | System Aware |
| ielowutil.exe | | 1,004 K | 696 K | 3032 | System Aware |
| procexp64.exe | 0.10 | 20,728 K | 33,980 K | 2408 | System Aware |

CPU Usage: 0.23% Commit Charge: 24.71% Processes: 53 Physical Usage: 22.51%

Figure 6 **Process Explorer Showing Purported DPI Awareness**

# Windows. Web. Mobile.
## Your next great app starts here.

DevExpress .NET controls, frameworks and libraries were built with you in mind. Built for those who demand the highest quality and expect the best performance... for those who require reliable tools engineered to meet today's needs and address tomorrow's requirements.

Experience the DevExpress Difference today and download your free 30-day trial and let's build great apps, together.

WIN WinForms | ASP ASP.NET | WPF WPF | SL Silverlight | Windows 8 XAML | HTML JS | Reporting | DevExtreme Mobile

Learn more and download your free trial
**devexpress.com/try**

DevExpress™

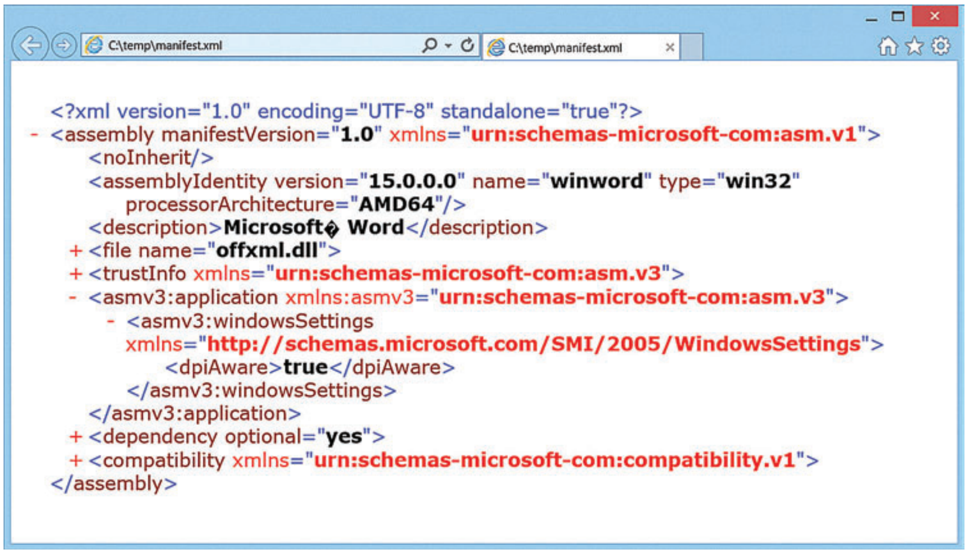

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
    <noInherit/>
    <assemblyIdentity version="15.0.0.0" name="winword" type="win32"
      processorArchitecture="AMD64"/>
    <description>Microsoft� Word</description>
  + <file name="offxml.dll">
  + <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  - <asmv3:application xmlns:asmv3="urn:schemas-microsoft-com:asm.v3">
    - <asmv3:windowsSettings
      xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
        <dpiAware>true</dpiAware>
      </asmv3:windowsSettings>
    </asmv3:application>
  + <dependency optional="yes">
  + <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
</assembly>
```

Figure 7 **The Microsoft Word Manifest File**

Applications can embed or distribute a manifest file—along with their binaries—that contains hints for the shell or Windows loader to use in determining how to prepare the application process before any code begins to run. The Windows shell uses this for a variety of purposes, including security features, assembly dependency information and, yes, DPI-awareness claims.

These manifest files are just text files. Either they're embedded within the executable as a traditional Win32 resource or they're simply shipped alongside the executable. You can, for example, take a peek at the manifest file for Word by using the manifest tool included with the Windows SDK using the following command:

```
mt -inputresource:"C:\ ... \WINWORD.EXE" -out:manifest.xml
```

I've elided the full path to conserve space. If all goes well, you'll be rewarded with an XML document in the current directory. You can see what this looks like in **Figure 7**. In the middle of some otherwise unrelated information about the Word executable is the dpiAware XML element and its value of true.

If an application doesn't have a manifest file and doesn't use the programmatic approach of setting its awareness level, then it's assumed to be DPI-unaware. If an application has a manifest file but doesn't contain the dpiAware element, then it's again assumed to be DPI-unaware. On Windows 7, the presence of the dpiAware element is enough for the system to assume it's DPI-aware. It doesn't matter what value this element contains or whether it even has a value. Window 8 is a little more specific. It expects a value starting with T, on the assumption that it says "True." Case doesn't matter.

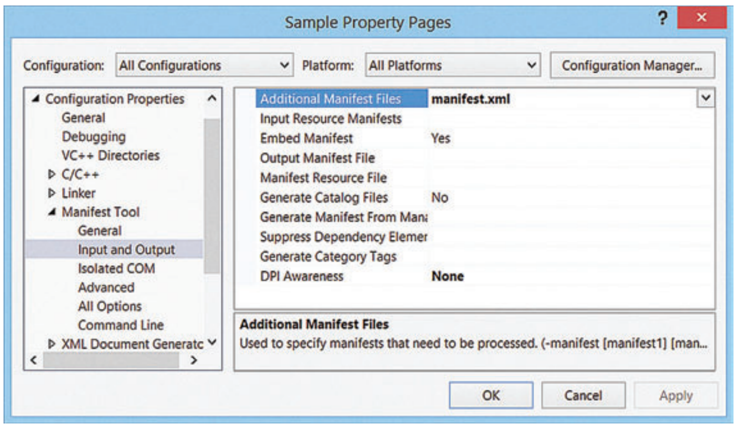
So that deals with DPI-unaware and system DPI-aware applications. What about per-monitor DPI-aware applications? Well, when this feature was first developed, a new value was chosen for the dpiAware element: Per Monitor. The desire to let application developers produce a single binary capable of targeting both Windows 7 and Windows 8 with support for system DPI awareness, as well as per-monitor DPI awareness on Windows 8, led to a new value to be chosen: True/PM. This enables Windows 7 and Windows 8 to accept the binary as system DPI-aware, and Windows 8.1 accepts it as per-monitor DPI-aware. You can, of course, continue to use Per Monitor if you only need to support Windows 8.1.

Unfortunately, Visual C++ 2013 doesn't yet support this new value, but there's a way to make it happen. The Manifest Tool that's part of the Visual C++ project build has an option to merge in additional manifest files. All you need to do is create a text file with the correct dpiAware element and value and merge it into your build. The following code shows the XML you need:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
  manifestVersion="1.0">
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <dpiAware xmlns=
        "http://schemas.microsoft.com/SMI/2005/WindowsSettings">
        True/PM</dpiAware>
    </windowsSettings>
  </application>
</assembly>
```

Once you've created the manifest file, you can simply update your project's settings, as illustrated in **Figure 8**. The manifest is set as an additional manifest file and the DPI awareness is set to None.

When it comes to rendering your application's window, you need to make sure you have the right DPI scaling factor. If you're looking after a legacy application with deep roots in GDI graphics and USER controls, then you're likely going to experience a lot of challenges in trying to make it per-monitor DPI-aware. These are the same challenges faced by the Windows and Office teams at Microsoft. But if you've been following my column, you know that Direct2D is the way to go. Direct2D takes care of all your DPI scaling needs and presents you with a logical coordinate system independent of the physical pixels and the DPI scaling factor. Of course, for Direct2D to be able to do this effectively, you need to tell it what DPI values to use for a particular render target.


Sample Property Pages

Configuration: All Configurations | Platform: All Platforms

| | |
|---|---|
| Additional Manifest Files | manifest.xml |
| Input Resource Manifests | |
| Embed Manifest | Yes |
| Output Manifest File | |
| Manifest Resource File | |
| Generate Catalog Files | No |
| Generate Manifest From Mani | |
| Suppress Dependency Elemer | |
| Generate Category Tags | |
| DPI Awareness | None |

**Additional Manifest Files**
Used to specify manifests that need to be processed. (-manifest [manifest1] [man...

Figure 8 **Manifest Tool Options**

NEW **HOSTING**

## MAXIMUM FLEXIBILITY FOR YOUR WEB PROJECTS

**HOSTING PACKAGES FOR PROFESSIONALS**

STARTING AT

**$0.99** per month*

### ALL INCLUSIVE
- Included Domains: .com, .net, .org, .biz, .info
- Unlimited Power: webspace, traffic, mail accounts, SQL databases
- Linux or Windows operating system

### POWERFUL APPS
- Over 140 popular apps (Drupal™, WordPress, Joomla!™, Typo3, Magento® and many more...)
- App Expert Support to answer your questions

### POWERFUL TOOLS
- Premium software, including Adobe® Dreamweaver® CS5.5, and NetObjects Fusion® 2013
- 1&1 Mobile Website Builder
- **NEW: PHP 5.5**, Perl, Python, Ruby

### SUCCESSFUL MARKETING
- 1&1 Search Engine Optimization
- Listing in business directories
- 1&1 E-Mail Marketing Manager
- Facebook® Credits

### STATE-OF-THE-ART TECHNOLOGY
- Maximum Availability (Geo-redundancy)
- 300 Gbit/s network connection
- 2 GB RAM guaranteed
- **NEW:** Maximum performance with 1&1 CDN powered by CloudFlare™
- **NEW:** SiteLock security scan included!

MEMBER OF united internet

Call **1 (877) 461-2631**

**1 TRIAL** 30 DAY MONEY BACK GUARANTEE

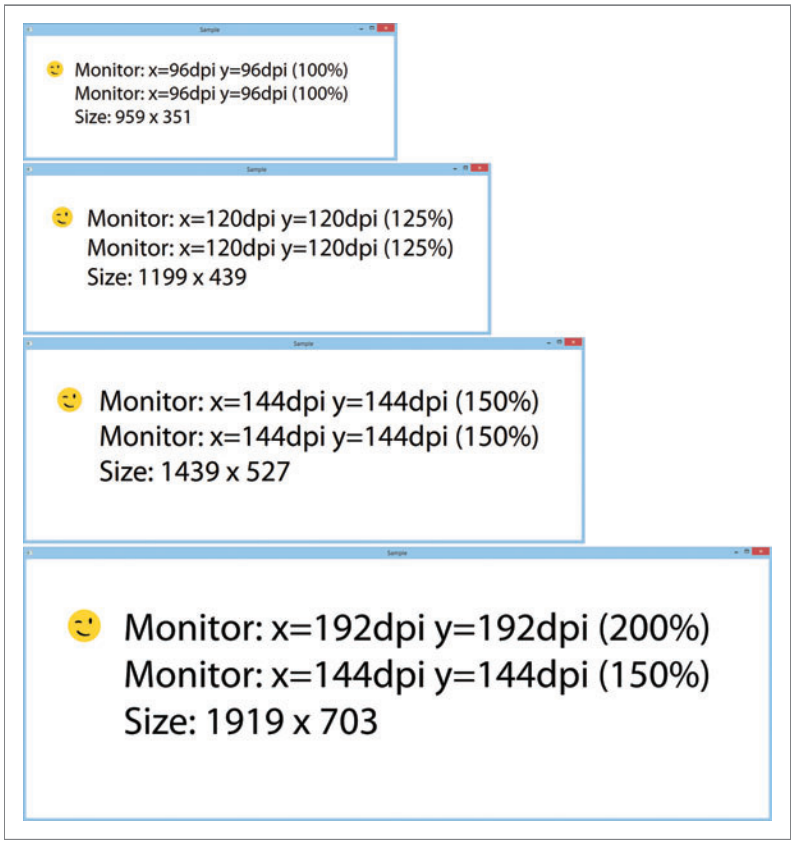
**1 MONTH** FLEXIBLE PAYMENT OPTIONS

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

1&1®

**1and1.com**

* Offer valid for a limited time only. The $0.99/month price reflects a 12-month pre-payment option for the 1&1 Starter Hosting package. Regular price of $2.99/month after 12 months. Some features listed are only available with package upgrade. (Visit www.1and1.com for full details.) Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2014 1&1 Internet. All rights reserved.


Monitor: x=96dpi y=96dpi (100%)
Monitor: x=96dpi y=96dpi (100%)
Size: 959 x 351

Monitor: x=120dpi y=120dpi (125%)
Monitor: x=120dpi y=120dpi (125%)
Size: 1199 x 439

Monitor: x=144dpi y=144dpi (150%)
Monitor: x=144dpi y=144dpi (150%)
Size: 1439 x 527

Monitor: x=192dpi y=192dpi (200%)
Monitor: x=144dpi y=144dpi (150%)
Size: 1919 x 703

Figure 9 **Per-Monitor DPI-Aware Application**

Traditionally, Direct2D applications simply retrieved the DPI values from the Direct2D factory, which in turn simply called GetDeviceCaps, as I illustrated earlier. But this is no longer sufficient. As I've shown, the DPI values may now change on the fly, and the value provided by GetDeviceCaps is only useful if you're trying to fit in with the rendering of a system DPI-aware application.

If you've invested in Direct2D, you're just a few steps away from being per-monitor DPI-aware.

Instead, right after you've created your Direct2D render target, you need to query the DPI value for the monitor nearest your window:

```
auto monitor = MonitorFromWindow(window, MONITOR_DEFAULTTONEAREST);
```

Given this monitor handle, you can call the GetDpiForMonitor function to retrieve the DPI values for this specific monitor:

```
auto x = unsigned {};
auto y = unsigned {};
VERIFY_(S_OK, GetDpiForMonitor(monitor, MDT_EFFECTIVE_DPI, &x, &y));
```

The effective DPI value for a given monitor won't necessarily correspond exactly to options presented in **Figure 4**. It again depends on a number of factors, including resolution, the physical DPI of the display and the assumed distance to the display surface. Finally, you can call the Direct2D render target's SetDpi method and Direct2D will take care of properly scaling your content as needed.

I did mention this can all happen dynamically. Your application may be running and suddenly the user changes the scale with the window shown in **Figure 3** or drags your window to another monitor with a different DPI scaling factor. In those cases, the Windows shell will send per-monitor DPI-aware application windows a new window message called WM_DPICHANGED.

Inside your message handler, you can once again call the MonitorFromWindow and GetDpiForMonitor functions to get the effective DPI values for the current monitor and then simply update your Direct2D render target with its SetDpi method again. Alternatively, the message's WPARAM packs both the x and y DPI values so this becomes a simple matter of extracting the low and high order words:

```
auto x = LOWORD(wparam);
auto y = HIWORD(wparam);
```

The WM_DPICHANGED message's LPARAM is, however, indispensable. It provides a new suggested position and size of your window based on the new scale factor that's now in effect. This ensures that while Direct2D takes care of scaling your content, you can also scale your window's actual size on the desktop and position it appropriately. The message's LPARAM is just a pointer to a RECT structure:

```
auto rect = *reinterpret_cast<RECT *>(lparam);
```

You can then simply call the SetWindowPos function to update your window's position and size:

```
VERIFY(SetWindowPos(window,
                    0, // No relative window
                    rect.left,
                    rect.top,
                    rect.right - rect.left,
                    rect.bottom - rect.top,
                    SWP_NOACTIVATE | SWP_NOZORDER));
```

And that's all it takes. If you've invested in Direct2D, you're just a few steps away from being per-monitor DPI-aware. Few applications have made the leap, so your application is bound to stand out! If you've got an existing Windows Presentation Foundation (WPF) application, there's also a sample available at bit.ly/IPDN3p that shows you how to integrate these same principles outlined in this article into your WPF code base.

**Figure 9** provides an example of a per-monitor DPI-aware application, automatically scaling both its content and its window size based on WM_DPICHANGED messages. The emoticon indicates which monitor the window is currently positioned on. For a more interactive example, please check out my Pluralsight course at bit.ly/1fgTifi, where you can also get the sample code for this application. ■

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

**THANKS** to the following technical expert for reviewing this article: *James Clarke (Microsoft)*

# PRECISELY PROGRAMMED FOR SPEED

## DynamicPDF—Comprehensive PDF Solutions for .NET Developers

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

DynamicPDF®

WWW.DYNAMICPDF.COM

**TRY OUR PDF SOLUTIONS FREE TODAY!**

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

ceTesoftware

LIVE! 360 DEV March 10 - 14, 2014 | Las Vegas, NV Planet Hollywood Resort & Casino

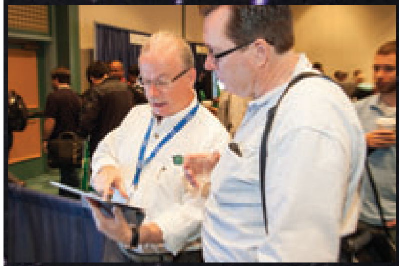
LAS VEGAS LIVE! 360 DEV

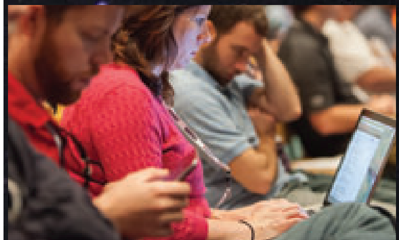
PLANET HOLLYWOOD RESORT AND CASINO

COMPREHENSIVE TRAINING FOR THE DEVELOPER WORLD

| LIVE! | LIVE! | LIVE! | LIVE! | LIVE! |
|---|---|---|---|---|
| Visual Studio | SQL Server | SharePoint | Web Dev | Modern Apps |

## COMPREHENSIVE TRAINING





The Developer World is always changing; new technologies emerge, current ones evolve and demands on your time grow. Live! 360 DEV offers comprehensive training through 5 co-located events on the most relevant and leading edge technologies in your world today. You'll learn from pre-eminent experts in the industry, network with like-minded peers, and return home with the knowledge and solutions you need to tackle your biggest development challenges.

live360events.com/lasvegas

## Live! 360 DEV Explores:

- Visual Studio Live! – May the Code be with you. The most trusted source in the universe for .NET Training for 21 years and counting.
- Web Dev Live! – NEW EVENT! Web Dev Live! will dive deep to explore all that is HTML5, JavaScript and ASP.NET.
- Modern Apps Live! – Launch your mobile, cross-device & cloud development training here.
- SharePoint Live! – Set your course for collaboration with these sessions designed strictly for devs.
- SQL Server Live! – Your Mission? Conquering coding against SQL Server.
- This means you have five events with over a hundred sessions to choose from – mix and match sessions to create your own, custom event line-up - it's like no other dev conference available today.

TURN THE PAGE FOR MORE EVENT DETAILS

## FOR THE DEVELOPER WORLD

## Explore the World of Live! 360 DEV

LIVE! Visual Studio

### Visual Studio Live!

Much like outer space, the .NET development platform is ever-changing and constantly expanding. Visual Studio Live! exists to help guide you through this universe, featuring code-filled days, networking nights and independent education. Whether you are a .NET developer, software architect or a designer, Visual Studio Live!'s multi-track events include focused, cutting-edge education on the .NET platform that you'll be ready to implement as soon as you get back to the office.

LIVE! Modern Apps

Presented in Partnership with Magenic

### Modern Apps Live!

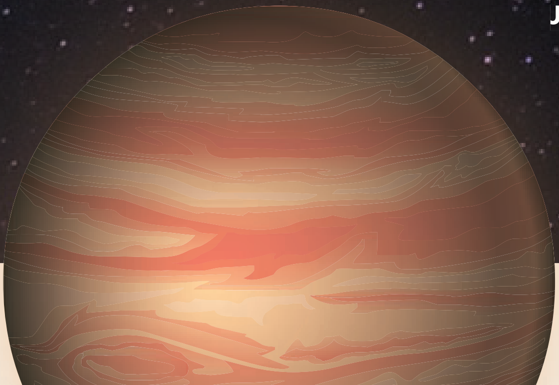
Presented in partnership with Magenic, Modern Apps Live! brings development managers, software architects, and development leads together to break down the latest and greatest techniques in low-cost, high-value application development. What sets Modern Apps Live! apart is the singular topic focus; sessions build on each other as the conference progresses, leaving you with a holistic understanding of modern applications.

## And INTRODUCING...

LIVE! Web Dev

### Web Dev Live!

The producers of Live! 360 DEV bring you a new, dynamic, content-rich event — Web Dev Live! We're seeing an evolution where server-side web technologies continue to improve, and at the same time JavaScript and HTML 5 are becoming viable smart client development technologies that operate on their own. This is a great time to jump into the JavaScript programming world, or to update your knowledge of web development based on the latest tools and capabilities.



live360events.com/lasvegas

LIVE! SharePoint

## SharePoint Live!

This March, SharePoint Live! will be for developers only, where expert speakers will guide an exploration of patterns and practices, highlighting the capabilities of various development tools. Select sessions will focus on methodologies for development that will supplement your overall Live! 360 DEV experience.

LIVE! SQL Server

## SQL Server Live!

Developers have an incredible amount of impact on the overall performance of a database application, and SQL Server Live! shines a spotlight to help developers deliver more. From T-SQL enhancements to "never do this" worst-practices reviews, there's plenty of content to supplement your Live! 360 DEV agenda!

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

## CONNECT WITH LIVE! 360

twitter.com/live360events

facebook.com/live360events

Join the "Live! 360" Group

REGISTER BY FEBRUARY 12

AND SAVE $300

Use promo code DEVFEB4

PLATINUM SPONSOR esri

SUPPORTED BY Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA

# Data, Meet My New Friend, F#

I've had some exposure to functional programming over the past few years. Some of this was implicit: Coding with lambdas in LINQ is functional programming. Some has been explicit: Using Entity Framework and the LINQ to SQL CompiledQuery type forced me to use .NET func logic. That was always a little tricky because I did it so infrequently. I've also been exposed to functional programming through the infectious enthusiasm of Rachel Reese, a Microsoft MVP, who not only participates in my local user group (VTdotNET) but also runs the VTFun group here in Vermont, which focuses on many aspects and languages of functional programming. The first time I went to a VTFun meeting, it was filled with mathematicians and rocket scientists. I'm not kidding. One of the regulars is a condensed matter theorist at University of Vermont. Swoon! I was a little overwhelmed by the high-level discussions, but it was fun to actually feel like the dummy in the room. Most of what was said went right over my head—except for a single statement that caught my attention: "Functional languages don't need no stinkin' foreach loops." Whoa! I wanted to know what that meant. Why don't they need foreach loops?

What I've usually heard is that functional programming is great for mathematical operations. Not being a math geek, I interpreted that to mean "for demos that use the Fibonacci sequence to test out asynchronous behavior" and didn't pay much more attention. That's the problem with hearing just a short elevator pitch about functional programming.

But I finally started hearing a more accurate characterization—that functional programming is great for data science. That certainly appeals to a data geek. F#, the functional language of the Microsoft .NET Framework, brings all kinds of data science capabilities to .NET developers. It has entire libraries dedicated to charting, time manipulation and set operations. It has APIs with logic dedicated to 2D, 3D and 4D arrays. It comprehends units of measure and is able to constrain and validate based on specified units.

F# also enables interesting coding scenarios in Visual Studio. Rather than building up your logic in code files and then debugging, you can write and execute code line by line in an interactive window and then move the successful code into a class file. You can get a great feel for F# as a language with Tomas Petricek's article, "Understanding the World with F#" (bit.ly/1cx3cGx). By adding F# into the .NET toolset, Visual Studio becomes a powerful tool for building applications that perform data science logic.

Code download available at msdn.microsoft.com/magazine/msdnmag0214.

In this article, I'll focus on one aspect of F# and functional programming that I've come to understand since hearing that comment about not needing foreach loops. Functional languages are really good at working with sets of data. In a procedural language, when you work with sets, you have to explicitly iterate through them to perform logic. A functional language, in contrast, comprehends sets on a different level, so you just need to ask it to perform a function on a set, rather than loop through the set and perform a function on each item. And that function can be defined with lots of logic, including math, if necessary.

LINQ provides shortcuts for this, even a ForEach method into which you can pass a function. But in the background, your language (perhaps C# or Visual Basic) simply translates this into a loop.

A functional language such as F# has the ability to perform the set function at a lower level, and it's much quicker thanks to its easy parallel processing. Add to this other key benefits, such as a rich math-processing ability and an incredibly detailed typing system

Figure 1 **CylinderMeasurement Class**

```
public class CylinderMeasurement
{
  public CylinderMeasurement(double widthA, double widthB, double height)
  {
    WidthA = widthA;
    WidthB = widthB;
    Height = height;
  }
  public int Id { get; private set; }
  public double Height { get; private set; }
  public double WidthB { get; private set; }
  public double WidthA { get; private set; }
  public int LoadPounds { get; private set; }
  public double Psi { get; set; }
  public CylinderType CylinderType { get; set; }
  public void UpdateLoadPoundsAndTypeEnum(int load, CylinderType cylType) {
    LoadPounds = load; CylinderType = cylType;
  }
   private double? Ratio {
    get {
      if (Height > 0 && WidthA + WidthB > 0) {
        return Math.Round(Height / ((WidthA + WidthB) / 2), 2);
      }
      return null;
    }
  }
  public double ToleranceFactor {
    get {
      if (Ratio > 1.94 || Ratio < 1) {
        return 1;
      }
      return .979;
    }
  }
}
```

that even understands units of measure, and you've got a powerful tool for performing calculations on large sets of data.

There's much more to F# and other functional languages that's still far beyond my reach. But what I want to do in this column is focus on a way to quickly benefit from one particular aspect of functional languages without making a huge investment: moving logic from the database into my application. This is the way I like to learn: find a few things I can understand and use them to take some baby steps into a new platform, language, framework or other type of tool.

I've heard Reese try to make it clear to developers that using F# doesn't mean switching development languages. In the same way you might use a LINQ query or a stored procedure to solve a particular problem, you can create a library of F# methods to solve the kinds of problems in your app that functional languages are really good at.

What I'll focus on here is extracting business logic that was built into my database, logic for handling large sets of data—something at which the database is excellent—and replacing it with functional methods.

And because F# is designed to work with sets and is really clever with mathematical functions, the code can be more efficient than it might be in SQL or in procedural languages such as C# or Visual Basic. It's so easy to have F# execute the logic on the items in the set in parallel. Not only can this reduce the amount of code you'd likely need in a procedural language to emulate this behavior, the parallelization means the code will run much faster. You could design your C# code to run in parallel, but I'd rather not go through that effort, either.

## A Real-World Problem

Many years ago, I wrote a Visual Basic 5 app that had to collect, maintain, and report a lot of scientific data and perform a lot of calculations. Some of those calculations were so complex I sent them to an Excel API.

You can create a library of F# methods to solve the kinds of problems in your app that functional languages are really good at.

One of the calculations involved determining the pounds per square inch (PSI) based on the amount of weight that caused a chunk of material to break. The chunk could be any one of a number of cylindrical shapes and sizes. The app would use the measurements of the cylinder and, depending on its shape and size, a specific formula to calculate its area. It would then apply a relevant tolerance factor and, finally, the amount of weight it took to break the cylinder. All of this together provided the PSI for the particular material being tested.

Figure 2 **Calculator Class to Calculate PSI**

```
public static class CylinderCalculator
  {
    private static CylinderMeasurement _currentCyl;

    public static void UpdateCylinders(IEnumerable<CylinderMeasurement> cyls) {
      foreach (var cyl in cyls)
      {
        _currentCyl = cyl;
        cyl.Psi = GetPsi();
      }
    }

    private static double GetPsi() {
      var area = GetAreaForCylinder();
      return PsiCalculator(area);
    }


    private static double GetAreaForCylinder() {
      switch (_currentCyl.CylinderType)
      {
        case CylinderType.FourFourEightCylinder:
          return 3.14159*((_currentCyl.WidthA + _currentCyl.WidthB)/2)/2*
            ((_currentCyl.WidthA + _currentCyl.WidthB)/2/2);
        case CylinderType.SixSixTwelveCylinder:
          return 3.14159*((_currentCyl.WidthA + _currentCyl.WidthB)/2)/2*
            ((_currentCyl.WidthA + _currentCyl.WidthB)/2/2);
        case CylinderType.ThreeThreeSixCylinder:
          return _currentCyl.WidthA*_currentCyl.WidthB;
        case CylinderType.TwoTwoTwoCylinder:
          return ((_currentCyl.WidthA + _currentCyl.WidthB)/2)*
            ((_currentCyl.WidthA + _currentCyl.WidthB)/2);
        default:
          throw new ArgumentOutOfRangeException();
      }
    }


    private static int PsiCalculator(double area) {
      if (_currentCyl.LoadPounds > 0 && area > 0)
      {
        return (int) (Math.Round(_currentCyl.LoadPounds/area/1000*
          _currentCyl.ToleranceFactor, 2)*1000);
      }
      return 0;
    }
  }
```

In 1997, leveraging the Excel API to evaluate the formula from within Visual Basic 5 and Visual Basic 6 felt like a pretty clever solution.

## Moving the Eval

Years later, I revamped the application in .NET. At that time, I decided to take advantage of the power of SQL Server to execute the PSI calculation on large sets of cylinders after they were updated by a user, rather than having the user's computer spend time on all of those calculations. That worked out pretty well.

More years passed and my ideas about business logic in the database changed. I wanted to pull that calculation back to the client side and, of course, the client machines were faster by then, anyway. It wasn't too difficult to rewrite the logic in C#. After a user updated a series of cylinders with the weight it took to break them (the load, represented in pounds), the app would iterate through the updated cylinders and calculate the PSI. Then I could update cylinders in the database with their new loads and PSI values.

For the sake of comparing familiar C# to the final outcome in F# (which you'll see shortly), I've provided the listing of the cylinder type, CylinderMeasurements, in **Figure 1** and my C# Calculator class in **Figure 2**, so you can see how I derive the PSIs for a set of cylinders. It's the CylinderCalculator.UpdateCylinders method that's called to start the PSI calculation for a set of cylinders. It iterates through each cylinder in the set and performs the appropriate calculations. Note that one of the methods, GetAreaForCalculation, is dependent on the cylinder type because I calculate the area of the cylinder using the appropriate formula.

## Data Focus and Faster Processing with F#

Finally, I discovered that F#, thanks to its natural inclination for manipulating data, provides a much better solution than evaluating one formula at a time.

At the introductory session on F# given by Reese, I explained this problem, which had been nagging at me for so many years, and asked if a functional language could be used to solve it in a more satisfying way. She confirmed that I could apply my complete calculation logic on a full set and let F# derive the PSIs for many cylinders in parallel. I could get the client-side functionality and a performance boost at the same time.

The key for me was to realize I could use F# to solve a particular problem in much the same way I use a stored procedure—it's simply another tool in my tool belt. It doesn't require giving up my investment in C#. Perhaps there are some who are just the reverse—writing the bulk of their apps in F# and using C# to attack particular problems. In any case, using the C# CylinderCalculator as a guide, Reese created a small F# project that did the task and I was able to replace a call to my calculator with a call to hers in my tests, as shown in **Figure 3**.

If, like me, you're new to F#, you might look only at the amount of code and see no point in choosing this route over C#. Upon closer examination, however, you may appreciate the terseness of the language, the ability to define the formulas more elegantly and, in the end, the simple way I can apply the calculatePsi function Reese defined to the array of cylinders I passed to the method.

I could get the client-side functionality and a performance boost at the same time.

The terseness is thanks to the fact that F# is better designed to perform math functions than C# is, so defining those functions is more efficient. But besides the geek appeal of the language, I was interested in performance. When I increased the number of cylinders per set in my tests, initially I did not see a performance improvement over C#. Reese explained that the test environment is more expensive when using F#. So I then tested the performance in a console app using the Stopwatch to report the time passed. The app built up a list of 50,000 cylinders, started the Stopwatch, passed the cylinders to either the C# or F# calculator to update the PSI value for each cylinder, then stopped the Stopwatch when the calculations were complete.

In most of the cases, the C# process took about three times longer than the F# process, although about 20 percent of the

Figure 3 **The F# PSI Calculator**

```
module calcPsi =
  let fourFourEightFormula WA WB = 3.14159*((WA+WB)/2.)/2.*((WA+WB)/2./2.)
  let sixSixTwelveFormula WA WB = 3.14159*((WA+WB)/2.)/2.*((WA+WB)/2./2.)
  let threeThreeSixFormula (WA:float) (WB:float) = WA*WB
  let twoTwoTwoFormula WA WB = ((WA+WB)/2.)*((WA+WB)/2.)
  // Ratio function
  let ratioFormula height widthA widthB =
    if (height > 0. && (widthA + widthB > 0.)) then
      Some(Math.Round(height / ((widthA + widthB)/2.), 2))
    else
      None

  // Tolerance function
  let tolerance (ratioValue:float option) = match ratioValue with
    | _ when (ratioValue.IsSome && ratioValue.Value > 1.94) -> 1.
    | _ when (ratioValue.IsSome && ratioValue.Value < 1.) -> 1.
    | _ -> 0.979


  // Update the PSI, and return the original cylinder information.
  let calculatePsi (cyl:CylinderMeasurement) =
    let formula = match cyl.CylinderType with
      | CylinderType.FourFourEightCylinder -> fourFourEightFormula
      | CylinderType.SixSixTwelveCylinder -> sixSixTwelveFormula
      | CylinderType.ThreeThreeSixCylinder -> threeThreeSixFormula
      | CylinderType.TwoTwoTwoCylinder -> twoTwoTwoFormula
      | _ -> failwith "Unknown cylinder"

    let basearea = formula cyl.WidthA cyl.WidthB
    let ratio = ratioFormula cyl.Height cylinder.WidthA cyl.WidthB
    let tFactor = tolerance ratio

    let PSI = Math.Round((float)cyl.LoadPounds/basearea/1000. * tFactor, 2)*1000.

    cyl.Psi <- PSI
    cyl

  // Map evaluate to handle all given cylinders.
  let getPsi (cylinders:CylinderMeasurement[])
            = Array.Parallel.map calculatePsi cylinders
```

time C# would beat F# by a small margin. I can't account for the oddity, but it's possible there's more I need to understand to perform truer profiling.

## Keeping an Eye Out for Logic That Begs for a Functional Language

So while I have to work at my F# skills, my new understanding is going to apply nicely to apps I already have in production as well as future apps. In my production apps, I can look at business logic I've relegated to the database and consider if an app would benefit from an F# replacement. With new apps, I now have a keener eye for spotting functionality I can code more efficiently with F#, performing data manipulation, leveraging strongly typed units of measurement and gaining performance. And there's always the fun of learning a new language and finding just the right scenarios for which it was built! ■

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

**Thanks** to the following technical expert for reviewing this article:
*Rachel Reese (Firefly Logic)*

# Extreme Performance Linear Scalability

NCache™
100% .NET

**For .NET & Java Apps**

JvCache™
100% Java

Remove data storage performance bottlenecks and scale your applications to extreme transaction processing (XTP). Cache data in memory and reduce expensive database trips. NCache scales linearly by letting you add inexpensive cache servers at runtime. JvCache is 100% Java implementation of NCache.

## Enterprise Distributed Cache

- Extremely fast & linearly scalable with 100% uptime
- Mirrored, Replicated, Partitioned, and Client Cache
- NHibernate & Entity Framework Level-2 Cache

## ASP.NET Optimization in Web Farms

- ASP.NET Session State storage
- ASP.NET View State cache
- ASP.NET Output Cache provider
- ASP.NET JavaScript merge/minify

## Runtime Data Sharing

- Powerful event notifications for pub/sub data sharing

FREE EXPRESS EDITION

**www.alachisoft.com** **Download a FREE trial!** **1-925-236-3830**

BRUNO TERKALY AND
RICARDO VILLALOBOS

# The Windows Azure Service Bus and the Internet of Things

Machine-to-machine (M2M) computing is fast becoming a technology that all developers and architects need to embrace. Numerous studies suggest a coming world of tens of billions of devices (a half-dozen for every human on earth) by 2020 (bit.ly/M2qBII). One reason this is happening is because it has never been easier for garage tinkerers and hobbyists to prototype a consumer or commercial device for later manufacture and sale. Getting started with both the hardware and software is incredibly inexpensive. For less than $100, you can order an Arduino or a Raspberry PI.

But you do get what you pay for and these devices—especially the Arduino—represent the low end of the device-capability spectrum. With 32KB of Flash storage and 4KB of RAM, they lack the power to run anything but the simplest aspects of a Web stack. The Arduino is an open source microcontroller fitted with a small chip, memory and storage. The software consists of a standard programming-language compiler and a boot loader that executes on the microcontroller. You can add expansion boards that plug into the device, hooking up motor controls, GPS, Ethernet, an LCD display, sensors, actuators and more. Pay a little more money and you can get the more powerful Raspberry PI, which comes with a version of the GNU/Linux OS and 256KB of RAM.

These devices rarely have value unless they're connected to something else, perhaps a cloud back end that receives data and sends commands. But connecting to, communicating with and managing these devices from an application running in the cloud presents some special challenges. The sheer number of devices, as well as their limited battery life and bandwidth, forces a cloud developer to carefully consider all options.

In this article, we'll take a look at how developers are trying to overcome the key challenges of addressability, bandwidth and security. We will debunk the myth that IPv6 and virtual private networks (VPNs) are simple, efficient, and secure, and will propose that the Windows Azure Service Bus is the perfect product to elegantly overcome these challenges. To drive home some of the concepts, we'll present four patterns you can use on your client device when communicating with the cloud back end. Finally, we'll present a brief introduction to Service Bus support for the Advanced Message Queuing Protocol (AMQP) 1.0—an interoperable and efficient protocol for device-to-cloud communication.

For many years, secure connectivity meant using TCP/IP with IPv4, combined with VPNs. This worked reasonably well, but is now showing signs of age. For starters, it's difficult to get a unique IP address out on the public Internet to use with a device—we've pretty much run out of IP addresses. Diehard fans of this approach promise that IPv6 will come to the rescue. The conventional wisdom is that if you give the device a unique IP address, all your difficult problems are solved. Unfortunately, this solves only a small part of the overall problem. Giving each device its own unique IP address is definitely not the silver bullet many had hoped.

Just to be clear, IPv6 and VPNs are fraught with problems in a crowded, connected-device world. Bandwidth, in particular, is a challenge. Chatty connectivity between device and network can lead to excessive traffic. Moreover, using typical HTTP request/response approaches for all messaging drains battery life on many devices. Perhaps most important is that security can't be guaranteed. VPNs are definitely insecure in some scenarios. Before presenting a solution, let's dive a little deeper and explore these problems.

Devices that create excessive network traffic communicating with the cloud back end are problematic. Bandwidth costs money, potentially a

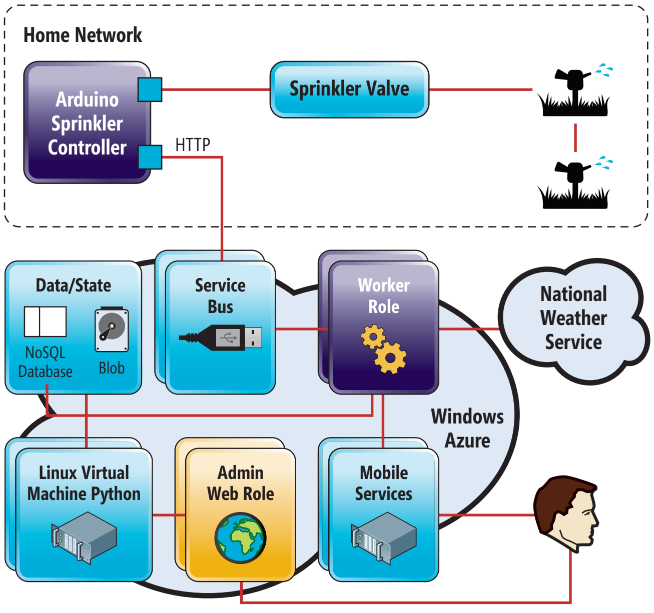


Figure 1 **Arduino Sprinkler System Reference Architecture**

lot of money if there are many chatty devices. Furthermore, more data means more CPU use, which means more energy, a precious resource on a mobile device. Most battery-powered devices equipped with a Wi-Fi transmitter and a SIM card need to enter a low-power "sleep" mode during periods when they aren't transmitting or receiving data. The IEEE 802.11 standard defines this power-save polling feature. The data gets buffered when the device is sleeping. Once it awakens, the buffered data is delivered. Chatty networks fill buffers and prematurely awaken sleeping devices.

HTTP request/reply approaches can be ridiculously wasteful, given the size of the payload relative to the overall HTTP request-response infrastructure. Suppose a device simply needs to report a number to the cloud, such as temperature, pressure or GPS coordinate. That binary data is probably only a few bytes in size, but the HTTP POST is generally at least 500 to 1,000 bytes, with the request header alone ranging from 200 to 2,000 bytes. Sure, some developers use tricks, like stuffing everything into the HTTP header to avoid the overhead of the body part of the HTTP request. But that isn't sufficient, and the size of the HTTP request only gets bigger when you have to transmit security credentials.

Here's some simple math to convince you. Imagine your device has to send temperature data every 5 seconds and the payload for the temperature data is a generous 20 bytes. In a 24-hour period, the temperature data by itself would transmit from the device to the cloud about 350,000 bytes. If you add in the HTTP request/response envelope, you raise each transmission by 800 bytes, a factor of 41, sending more than 14MB to the cloud instead of just the 350KB of temperature data. This can get prohibitively expensive if you're supporting thousands of devices.

Perhaps the biggest misconception is that VPNs are inherently safe. The reality is that VPN networks can be risky, especially when devices connected to a VPN are outside the manufacturer's or operator's immediate physical control. Once a single device is breached, all devices connected to the same VPN are vulnerable. Once an untrusted user gets access to a connected device, he or she can use the device to explore and attack your internal resources. Despite these shortcomings, VPNs are often the only option offered by many carriers.

## The Windows Azure Service Bus Approach

Windows Azure Service Bus offers some great solutions to these challenges. Leveraging the Service Bus is more secure, because the device is only an endpoint on the Internet where it can place messages into a queue. The device can't reach other protected network resources inside cloud services. In addition, using the Service Bus for device connectivity costs less in terms of power, because the device can sleep more often, waking up periodically to pull any waiting messages from the queue.

The Service Bus provides even more value because it can:

- Decouple device communication and interaction from your cloud service
- Enable load leveling and load balancing among several instances of your back-end service
- Identify duplicate messages
- Gather messages into logical groups (called Sessions)

Figure 2 **Four Patterns for Device-Cloud Service Communication**

| Pattern | Summary | Example |
|---|---|---|
| Telemetry | A client device sends data (one way) to a cloud service. | A device publishes messages about the temperature to Topics. The cloud service subscribes to some or all of these temperature messages. |
| Inquiry | A client device sends a query to the cloud service and receives a response. | A device inquires about upcoming weather conditions by posting a weather inquiry to a topic. The cloud service subscribes to inquiries and posts a message response to a topic of its own to which the device subscribes. |
| Command | A cloud service issues a command to a client device and the client device returns a success or failure response. | The cloud service publishes a temperature message/command to a topic to which a device subscribes. The device then turns water on or off and sends a reply back to the cloud service by posting a response to a topic. |
| Notification | A cloud service issues a one-way out-of-band notification to a client device that's important for the device's operation. | The cloud service sends a time-reset message to a device by publishing the message to a topic to which that device subscribes. |

- Implement transactional behavior and atomicity
- Support ordered delivery of messages and provide a time-to-live for each message
- Extend to publish-subscribe scenarios easily using Topics and Subscriptions

To get a more concrete idea of how a device connects to and communicates with the cloud back end, take a look at **Figure 1**, which depicts how a special-purpose device might fit into a bigger architecture. We'll use the canonical example of OpenSprinkler, an open source, Internet-based sprinkler/irrigation valve controller that's capable of checking the weather before deciding to turn on the water. It's built using Arduino parts. Note in **Figure 1** that the Arduino controls the sprinkler system using a home network as the Internet connection and communicates with a cloud back end.

## Solving Connectivity Problems

Windows Azure Service Bus does a great job of solving the addressability and network connectivity challenges. The Arduino device would probably sit behind some NAT layer, making it difficult to reach from a cloud service. Fortunately, the Service Bus dramatically simplifies connectivity by acting as a relay service and serving as a proxy for a cloud back end. Moreover, it can provide Queues, Topics and Subscriptions, which enables it to act as an event hub for messages sent between the cloud and the device. The decoupled nature of a queue acting as a relay lets the device asynchronously send and receive messages to and from the cloud, even if only occasionally connected. For security, the device can be authenticated with SharedAccessSignature, SharedSecret, SAML or SimpleWebToken.

Notice in **Figure 1** that one or more worker roles may be reading from the Service Bus message queue. Worker roles can make decisions and issue commands back to the device. Other worker roles might be getting weather information from other systems, such as the National Weather Service. Worker roles may also be saving all the unprocessed incoming events into a NoSQL database, such as MongoDB.

**Figure 1** also shows mobile users interacting with Web roles to schedule watering. Mobile users can receive push notifications from Windows Azure Mobile Services (WAMS), which supports all the major notification networks, such as Windows Notification Services (WNS), Microsoft Push Notification Service (MPNS), Apple Push Notification Service (APNS) and Google Cloud Messaging for Android (GCM). WAMS makes it easy to support Windows, iOS and Android.

You can even envisage a machine-learning part of the architecture. Windows Azure can support Linux VMs and it's quite simple to configure PyMongo (a Python driver for MongoDB) to read the event stream produced by various devices and use machine-learning techniques in PyML to find patterns or make predictions about the event stream data. Based on certain predictions or patterns, the cloud service can choose to send commands to the device, such as turning on or shutting off the water.

A messaging system that's the primary endpoint for sending and receiving data is extensible, because devices can continue to send a single message stream while new Subscriptions can be added to a Service Bus Topic for each new system that will consume that message stream. These systems can be for real-time analytics and machine-learning as well as other scenarios described earlier.

## Communicating with the Cloud

There are four patterns that can be used on the client to communicate with the cloud service. These are outlined in **Figure 2**.

All four patterns leverage Service Bus Topics and Subscriptions. Depending on the direction of the communication (device to cloud service or cloud service to device), the device can either subscribe to topics or publish to topics. Topics are simply a mechanism to send messages, while subscriptions are used to consume messages. You can create filter rules for subscriptions to allow more fine-grained control over which messages are retrieved. Worker roles in the cloud service can be used to publish messages to topics or to consume messages from subscriptions.

Due to space constraints, we can't illustrate all of the patterns here in this article, so we'll delve into just one. **Figure 3** shows a reference implementation of the Command pattern. It demonstrates that devices from Buildings 1 and 2 can subscribe to messages (Commands) and post responses back to topics. Note that worker roles in the cloud service can publish messages to a Temperature and Shade Command Topic and that specific devices can separately subscribe to Temperature or Shade Control messages. Service Bus Topics and Subscriptions can be used in a wide variety of combinations to partition the message flow appropriately.

## AMQP and Interconnectivity

The Windows Azure Service Bus team recently announced support for the Advanced Message Queuing Protocol (AMQP) 1.0, an open standard with a binary application layer for message-oriented middleware. Its main value is that it's highly interoperable and that it uses a binary format on the wire to minimize payload size.

AMQP supports reliable message transfer, queuing, routing, pub/sub and more. Because it's a wire-level protocol targeting data streaming across the network, any compliant tool can interact with the data regardless of the implementation language. This enables cross-platform, hybrid applications using an open, standard protocol. The library lets you mix and match languages, frameworks, and OSes, supporting .NET, Java, Python and PHP. You'll find more information on the Windows Azure Samples page at `aka.ms/G3izk8`. Designed to be light and interoperable, AMQP is a good fit for many of today's devices needing connectivity to a cloud back end.

However, AMQP is too much software for today's Arduino, which lacks the necessary memory, storage and processing power. Running AMQP requires support for Transport Layer Security (TLS), a cryptographic protocol that provides communication security over TCP. TLS uses X.509 certificates (asymmetric cryptography) to validate the identity of communicating parties across the wire. In addition, the Apache Qpid Proton client-based messaging library is often used to integrate with the AMQP to simplify communications across routers, bridges and proxies. All of this raises the question: How do you support low-end devices connecting to cloud back ends while enjoying the benefits of the Service Bus messaging infrastructure?

One option is to pay more money and get a Raspberry PI. If you don't want to do that, you'll

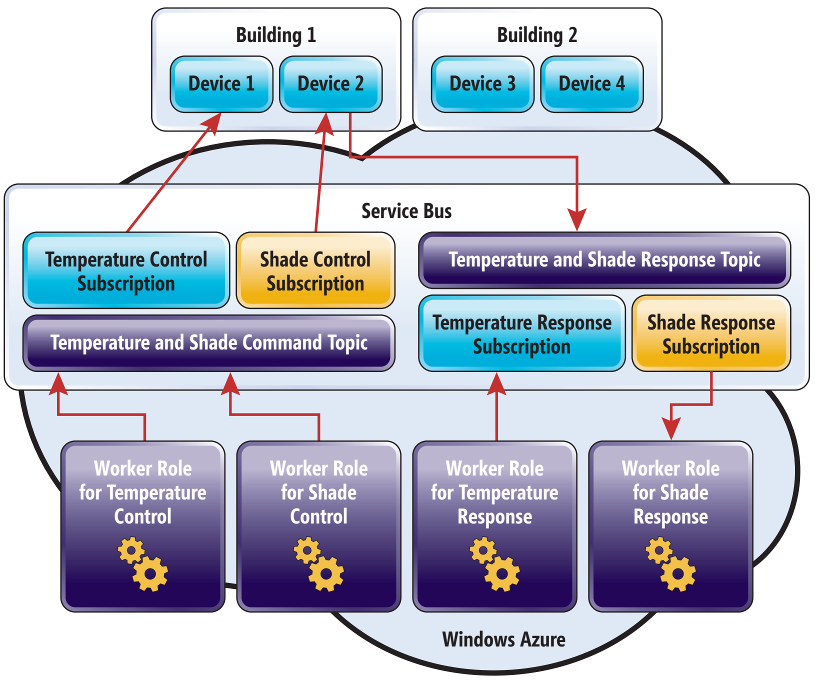

Building 1
Device 1
Device 2
Building 2
Device 3
Device 4
Service Bus
Temperature Control Subscription
Shade Control Subscription
Temperature and Shade Response Topic
Temperature and Shade Command Topic
Temperature Response Subscription
Shade Response Subscription
Worker Role for Temperature Control
Worker Role for Shade Control
Worker Role for Temperature Response
Worker Role for Shade Response
Windows Azure

Figure 3 **Architecture for the Command Pattern**

need to be more creative. You can start by leveraging Clemens Vasters' code at bit.ly/1acvLdS, which lets an Arduino receive a command to blink an LED light on the microcontroller. The code implements a device gateway, providing a TCP endpoint to which the Arduino connects. To maintain the connection through NATs and the Windows Azure load balancer, the cloud service needs to ping the Arduino every 235 seconds (just less than 4 minutes). See Vasters' C# project, LedBlinkerServer.

In our next column, we'll take a deeper dive to explain how the code works and how you can get the Arduino to send and receive messages to and from the Service Bus.

## Wrapping Up

In this month's column we presented four patterns that can be used to build a reliable message exchange between a device and cloud services. We introduced AMQP, the open source message-queuing protocol that helps to increase interoperability and minimize bandwidth and is completely supported by the Windows Azure Service Bus. Finally, we began discussing how to support low-end devices connecting to cloud back ends while using the Service Bus messaging infrastructure, which we'll continue in our next article.

We'd like to thank Clemens Vasters and Abhishek Lal for helping us understand the brave new world of connected devices. Clearly, the world of special-purpose devices connected to cloud services is growing rapidly. Traditional approaches to communicating with a cloud service need to be reevaluated. Security, bandwidth, network reliability, and interoperability are just some of the challenges that architects and developers face with special-purpose devices in the M2M world. Using the Windows Azure Service Bus makes those challenges far less daunting. ■

**Bruno Terkaly** *is a developer evangelist for Microsoft. His depth of knowledge comes from years of experience in the field, writing code using a multitude of platforms, languages, frameworks, SDKs, libraries and APIs. He spends time writing code, blogging and giving live presentations on building cloud-based applications, specifically using the Windows Azure platform. You can read his blog at blogs.msdn.com/b/brunoterkaly.*

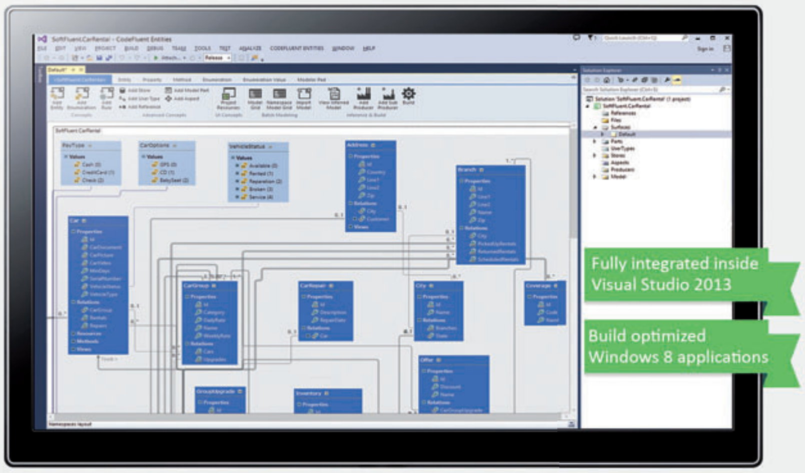
**Ricardo Villalobos** *is a seasoned software architect with more than 15 years of experience designing and creating applications for companies in multiple industries. Holding different technical certifications, as well as a master's degree in business administration from the University of Dallas, he works as a cloud architect in the DPE Globally Engaged Partners team for Microsoft, helping companies worldwide to implement solutions in Windows Azure. You can read his blog at blog.ricardovillalobos.com.*

*Terkaly and Villalobos jointly present at large industry conferences. They encourage readers of Windows Azure Insider to contact them for availability. Terkaly can be reached at bterkaly@microsoft.com and Villalobos can be reached at Ricardo.Villalobos@microsoft.com.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Abhishek Lal and Clemens Vasters*

Save your time!

Stop writing repetitive code and focus on what matters


Fully integrated inside Visual Studio 2013
Build optimized Windows 8 applications

Generate rock-solid foundations for your .NET applications

CodeFluent Entities

A centralized model

Don't ever repeat yourself. A single model, from Visual Studio drives all your developments from database to UI.

Continuous generation

Generate continuously throughout developments without losing code or data.

Flexibility

Support multiple databases, languages, user interfaces and architectures by simply turning features on and off.

Migration & interoperability

Import from existing databases and create .NET applications on top of them or migrate them to new ones.

GET A LICENCE WORTH $399 FOR FREE

Go to www.softfluent.com/forms/msdn-2014

Tools for developers, by developers.

More information at www.softfluent.com

Contact us at info@softfluent.com

SoftFluent

# Explore the Microsoft .NET Framework 4.5.1

Gaye Oncul Kok

The Microsoft .NET Framework 4.5.1 release, along with Visual Studio 2013, introduces innovative features to increase developer productivity and application performance. Additionally, it provides new features for improving the UX of consuming .NET NuGet packages, which is important because NuGet is a primary delivery vehicle for .NET Framework libraries.

The previous product, the .NET Framework 4.5, was a big release with many new features. It has been installed on more than 200 million machines. The .NET Framework 4.5.1 was released about 14 months later in October 2013, and despite the short time frame, it comes packed with many features requested by customers. In this article, I'll review the new features in the .NET Framework 4.5.1, and for more details, you can refer to .NET Framework 4.5.1 RTM (bit.ly/1bBIEPN) and .NET Framework 4.5.1 Preview (bit.ly/10Vr2ft) posts on the .NET Framework Blog.

The .NET Framework 4.5.1 is only a part of what the .NET team (of which I'm a member) has been working on over the past year. We also shipped several libraries on NuGet to fill platform gaps and to enable new scenarios. I'll provide an overview of our .NET NuGet libraries and also highlight one of our deep investments, the new .NET just-in-time (JIT) compiler, which shipped as a Community Technology Preview (CTP) release around the same time as the .NET Framework 4.5.1.

This article discusses:

- New debugging features
- Application performance enhancements
- Easier use of NuGet libraries
- The new RyuJIT compiler

Technologies discussed:

Microsoft .NET Framework 4.5.1, Visual Studio 2013

## More Productive Development

I'll start with new debugging features delivered with the .NET Framework 4.5.1 to improve developer productivity.

**Async Debugging Improvements** After setting up a solid and easy-to-use base for the asynchronous programming model in the previous Framework releases, we wanted to smooth out some remaining aspects for the overall developer experience with the .NET Framework 4.5.1. Two questions are essential for debugging asynchronous code: "How did I get into this async method?" and "What is the state of all the tasks in my application?" Visual Studio 2013 introduces enhancements to the Call Stack and Tasks windows to help you find answers to these questions in a much more intuitive way. These improvements are supported for desktop, Web and Windows Store apps on Windows 8.1 and are available for C++ and JavaScript as well.

It's common to have nested async method calls within an app or library, which rely on the await keyword to manage the flow of execution. Previously, Visual Studio didn't show the chain of async calls when stopped at a breakpoint within a Task. Visual Studio 2013 provides a logical and sequential view of methods in a nested chain of calls for both asynchronous and synchronous methods. This makes it easier to understand how the program reached a location inside an asynchronous call.

**Figure 1** shows an asynchronous code sample. **Figure 2** and **Figure 3** demonstrate the difference between the call stack views of Visual Studio 2012 and Visual Studio 2013 for that code. More details of this

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## LAS VEGAS 2014

**March 10 - 14 | Planet Hollywood Hotel & Casino**

vslive.com/lasvegas

MAY THE CODE BE WITH YOU

Visual Studio LIVE!
YOUR GUIDE TO THE .NET DEVELOPMENT UNIVERSE
LIVE!

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

**Topics include:**

- Visual Studio/ .NET
- Windows 8.1/ WinRT
- WPF/Silverlight
- Cloud Computing
- Windows Phone
- Cross-Platform Mobile Development
- ASP.NET
- HTML5
- Mobile Web
- JavaScript

WANT TO LEARN MORE?

SCAN THIS QR CODE TO JOIN OUR MAILING LIST

Turn over for BONUS CONTENT →

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

LAS VEGAS 2014
March 10 – 14 | Planet Hollywood Hotel & Casino

LIVE! 360 DEV

Visual Studio Live! Las Vegas is part of Live! 360 DEV, which means you'll have access to four (4) other co-located events at no additional cost:

**Five (5) events** means over a hundred sessions to choose from – mix and match sessions to create your own, custom event line-up – it's like no other dev conference available today!

live360events.com/lasvegas

SQL Server LIVE!
SQL SERVER FOR MODERN DEVELOPERS

SharePoint LIVE!
TRAINING FOR COLLABORATION

Modern Apps LIVE!
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

AND INTRODUCING Web Dev Live!

Web Dev LIVE!
HTML5, JAVASCRIPT & ASP.NET TRAINING

**REGISTER BY FEBRUARY 12 AND SAVE $300!**

vslive.com/lasvegas

**CONNECT WITH VISUAL STUDIO LIVE!**

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

Use promo code VSLFEBTI

EVENT SPONSOR Microsoft | PLATINUM SPONSOR esri | SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE | PRODUCED BY 1105 MEDIA INC

Figure 1 **Asynchronous Code Sample**

```
private async void ShowSampleImg_Click(object sender, RoutedEventArgs e)
{
  string imgUri = "http://example.com/sample.jpg";
  BitmapImage bitmap = new BitmapImage();
  bitmap.BeginInit();
  bitmap.StreamSource = await GetSampleImgMemStream(imgUri);
  bitmap.EndInit();
  sampleImg.Source = bitmap;
}
private async Task<MemoryStream> GetSampleImgMemStream(string srcUri)
{
  Stream stream = await GetSampleImage(srcUri);
  var memStream = new MemoryStream();
  await stream.CopyToAsync(memStream);
  memStream.Position = 0;
  return memStream;
}
private async Task<Stream> GetSampleImage(string srcUri)
{
  HttpClient client = new HttpClient();
  Stream stream = await client.GetStreamAsync(srcUri);
  return stream;
}
```

feature can be found in the "Debugging Asynchronous Code in Visual Studio 2013—Call Stack enhancements" blog post at bit.ly/19NTNez.

The Tasks window in Visual Studio 2013 is designed to help you understand the state of async tasks in your apps by displaying all the currently running and scheduled tasks. It's a replacement for the Parallel Tasks window that was available in previous Visual Studio versions. **Figure 4** shows a snapshot of a Visual Studio 2013 Tasks window for the sample code given in **Figure 1**.

**x64 Edit and Continue** This was a popular debugger feature request, with more than 2,600 votes on the Visual Studio UserVoice site where users can request new features (bit.ly/14YIM8X). Developers have loved using the Edit and Continue feature since it was introduced with Visual Studio 2005 and the .NET Framework 2.0 release, for x86 projects. Edit and Continue makes it easier to write the correct code by letting you change the source code during a debugging session, while app state is available. You can even move the instruction pointer so you can replay code after making a change. It provides a more productive development experience because you don't have to stop and restart the session to validate your changes.

x64 support for Edit and Continue is now enabled with Visual Studio 2013 and the .NET Framework 4.5.1 release. You can use this feature for debugging desktop applications (Windows Presentation Foundation, Windows Forms and so on), Windows Store apps, ASP.NET Web applications and Windows Azure Cloud Services projects targeting x64, AnyCPU or x86 architectures.

**Managed Return Value Inspection** Debugger support for managed return values is another popular request with more than 1,000 votes on the UserVoice site. The Visual C++ debugger has an existing feature that allows you to observe the return values of methods, and we wanted the same capability for .NET as well. This feature is useful for many code patterns. However, you can really see its value with nested methods, as demonstrated in **Figure 5**. With this feature, you no longer have to worry about storing the results of your methods in locals solely to make debugging easier. When you step over a method call, both direct return values and the return values of the embedded methods will be displayed in the Autos window along with the parameter values passed to the functions. You can also use the Immediate window to access the last return value through the use of the new $ReturnValue pseudo-variable.

**Windows Store Development Enhancements** We responded to feedback and provided .NET support for new Windows Runtime (WinRT) features to improve the .NET Windows Store app development experience.

One of the pain points was converting a .NET Stream to a WinRT IRandomAccessStream. In the .NET Framework 4.5.1, we added a new extension method, AsRandomAccessStream, for System.IO.Stream to solve this problem. You can now write the following code, which allows you to easily provide an IRandomAccessStream:

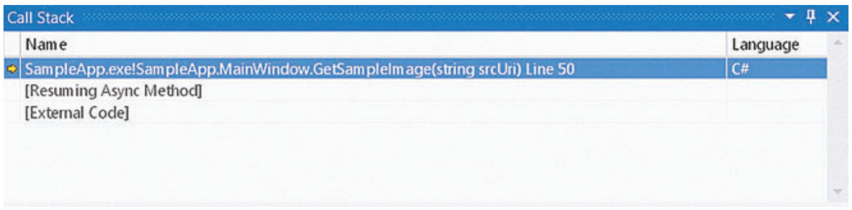
```
// EXAMPLE: Get image from URL via networking I/O
var client = new HttpClient();
Stream stream = await client.GetStreamAsync(imageUrl);
var memStream = new MemoryStream();
await stream.CopyToAsync(memStream);
memStream.Position = 0;
var bitmap = new BitmapImage();
bitmap.SetSource(memStream.AsRandomAccessStream());
image.Source = bitmap;
```

This example code reads an image from the Web and displays it in a XAML Image control (represented by the "image" variable).

Another improvement is error propagation in the Windows Runtime. The Windows Runtime, in Windows 8.1, enables exceptions to pass between WinRT components. With this support, an exception can be thrown from a C++ WinRT component and be caught in C# (or vice versa). Additional information for the exception is now available via the Message and StackTrace properties on System.Exception.

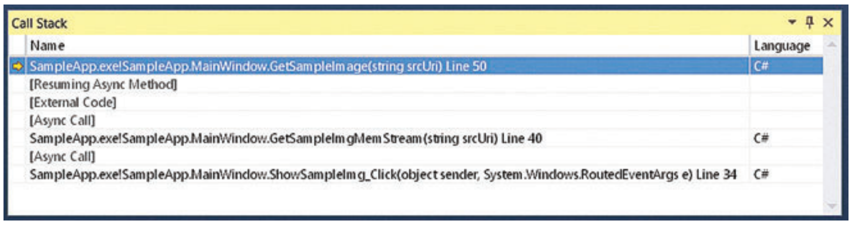
The Windows Runtime also added support for nullable value types in structures. You can build managed WinRT components that expose structs with this new feature, such as in this sample code:

```
public struct PatientRecord
{
  public string Name;
  public int Age;
  public string HomeAddress;
  // InsuranceID is nullable
  public int? InsuranceId;
}
```



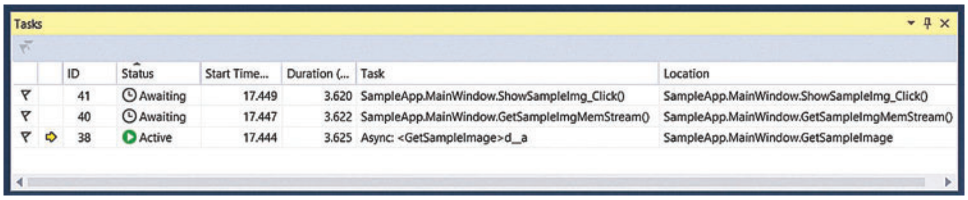
| Name | Language |
|---|---|
| SampleApp.exe!SampleApp.MainWindow.GetSampleImage(string srcUri) Line 50 | C# |
| [Resuming Async Method] | |
| [External Code] | |

Figure 2 **Visual Studio 2012 Call Stack Window**



| Name | Language |
|---|---|
| SampleApp.exe!SampleApp.MainWindow.GetSampleImage(string srcUri) Line 50 | C# |
| [Resuming Async Method] | |
| [External Code] | |
| [Async Call] | |
| SampleApp.exe!SampleApp.MainWindow.GetSampleImgMemStream(string srcUri) Line 40 | C# |
| [Async Call] | |
| SampleApp.exe!SampleApp.MainWindow.ShowSampleImg_Click(object sender, System.Windows.RoutedEventArgs e) Line 34 | C# |

Figure 3 **Visual Studio 2013 Call Stack Window**



| | | ID | Status | Start Time... | Duration (... | Task | Location |
|---|---|---|---|---|---|---|---|
| | | 41 | Awaiting | 17.449 | 3.620 | SampleApp.MainWindow.ShowSampleImg_Click() | SampleApp.MainWindow.ShowSampleImg_Click() |
| | | 40 | Awaiting | 17.447 | 3.622 | SampleApp.MainWindow.GetSampleImgMemStream() | SampleApp.MainWindow.GetSampleImgMemStream() |
| | | 38 | Active | 17.444 | 3.625 | Async: <GetSampleImage>d__a | SampleApp.MainWindow.GetSampleImage |

Figure 4 **Visual Studio 2013 Tasks Window**

## Better Application Performance

Application performance is a constant focus area for the .NET Framework team. In this release, we responded to feedback on the garbage collector and significantly improved ASP.NET app startup.

**ASP.NET App Suspension** This feature is one of the top highlights of the .NET Framework 4.5.1 due to the significant performance gain it provides, particularly for shared hosting scenarios where site density and startup latency are critical. ASP.NET App Suspension will enable shared hosters—either commercial Web hosting companies or enterprise IT systems—to host many more ASP.NET Web sites on a server with faster app startup time.

ASP.NET App Suspension depends on IIS Idle Worker Process Page-Out, which is a new IIS feature in Windows Server 2012 R2. IIS Idle Worker Process Page-Out introduces a new "suspended" state in addition to the existing "inactive" and "active" states for Web sites. This new "suspended" state releases critical resources used by the site for other sites to use, specifically CPU and memory, while still enabling the site to be resumed quickly.

**Figure 6** shows the state transitions of ASP.NET sites using App Suspension. A Web site starts in the inactive state. It's loaded into memory and transitions to active with the first page request. After a period of idle time, the site will be suspended, per application pool configuration (bit.ly/1aajEeL). Upon subsequent requests to the site, it can quickly return to the active state. This cycle can happen many times. Up until now, sites would get terminated and become inactive after a certain amount of idle time.

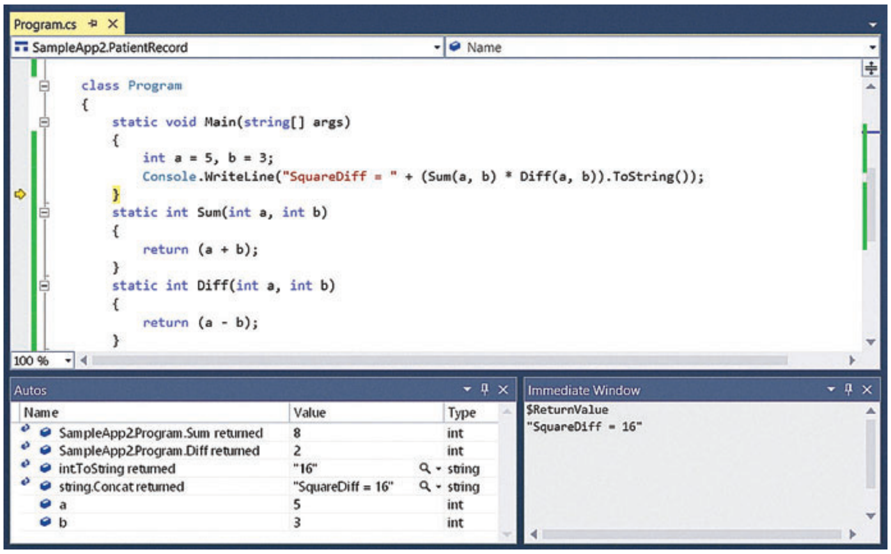
No code change is required to use this new feature. ASP.NET App Suspend is enabled automatically by configuring an IIS application pool for "Suspend" on Windows Server 2012 R2.

Earlier I touted a "significant performance gain" achieved with this feature, and I'd like to back this up with some numbers coming from our performance labs. We conducted extensive performance experiments to measure the startup time gain for "resume from suspend" compared to "start after terminate." We did these experiments on a machine under significant request load, accessing a large number of application pools, with the intent of recreating a "shared hosting" environment. The results showed a 90 percent reduction in the startup time for sites that were accessed after suspension. We also measured the improvement to site density. We were able to host about seven times more ASP.NET sites on Windows Server 2012 R2 when ASP.NET App Suspension was enabled. **Figure 7** shows the results of these experiments. More insights into these experiments can be found in the "ASP.NET App Suspend – responsive shared .NET Web hosting" blog post at bit.ly/17fl6dM.

**Multi-Core JIT Compilation Enhancements** Multi-core JIT compilation is now enabled by default for ASP.NET apps. Performance measurements show up to 40 percent reductions in cold startup time with multi-core JIT enabled. It provides startup benefits by performing JIT compilation on multiple cores, in parallel to code execution. Under the covers, multi-core JIT was extended to support dynamically loaded assemblies, which are common in ASP.NET apps. The additional support also benefits client apps, where multi-core JIT remains an opt-in feature. More details about the multi-core JIT feature can be found in the related .NET Framework Blog post, "An easy solution for improving app launch performance," at bit.ly/RDZ4eE.

**On-Demand Large Object Heap (LOH) Compaction** LOH compaction is an important requirement for some scenarios, and it's now available in this release. First, a little background information, as LOH might not be familiar to you. The garbage collector stores objects larger than 85,000 bytes in the LOH. The LOH can get fragmented, and in some cases this might lead to relatively large heap sizes or even OutOfMemoryException exceptions. These situations, although rare, occur because there aren't enough contiguous memory blocks available in the LOH to satisfy an allocation request, even though there might be enough space in total.

With LOH compaction, you can reclaim and merge smaller unused memory blocks, making them available for larger allocations, which makes better overall use of machine memory. Although this idea sounds appealing, the feature isn't intended for common use. Compacting LOH is an expensive process and can cause long pauses in an application, so it should only be deployed into production after analysis and testing.



```
class Program
{
    static void Main(string[] args)
    {
        int a = 5, b = 3;
        Console.WriteLine("SquareDiff = " + (Sum(a, b) * Diff(a, b)).ToString());
    }
    static int Sum(int a, int b)
    {
        return (a + b);
    }
    static int Diff(int a, int b)
    {
        return (a - b);
    }
```

| Name | Value | Type |
|---|---|---|
| SampleApp2.Program.Sum returned | 8 | int |
| SampleApp2.Program.Diff returned | 2 | int |
| intToString returned | "16" | string |
| string.Concat returned | "SquareDiff = 16" | string |
| a | 5 | int |
| b | 3 | int |

```
$ReturnValue
"SquareDiff = 16"
```

Figure 5 **Visual Studio 2013 Autos and Intermediate Windows**

/update/2014/02

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

## ComponentOne Studio Enterprise 2013 v3 | from $1,315.60

ComponentOne a division of GrapeCity

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

- Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers
- Visual Studio 2013 and Bootstrap support
- New advanced theming tools for WinForms and ASP.NET
- 40+ UI widgets built with HTML5, jQuery, CSS3, and SVG
- New Windows Store Sparkline, DropDown, & Excel controls

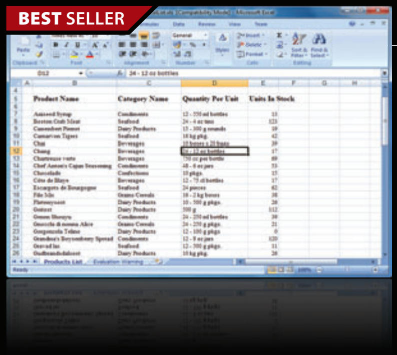
BEST SELLER
Using the Project Explorer
Project Explorer Sections
Navigating in the Explorer
Tips & Tricks

## Help & Manual Professional | from $583.10

ec software

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

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



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

ASPOSE Your File Format Experts

**Every Aspose .NET component in one package.**

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

## GdPicture.NET Ultimate | from $4,955.09

GdPicture imaging technologies

**All-in-one AnyCPU document-imaging and PDF toolkit for .NET and ActiveX.**

- Document viewing, processing, printing, scanning, OMR, OCR, Barcode Recognition, DICOM
- Annotate image and PDF within your Windows & Web applications
- Read, write and convert vector & raster images in more than 90 formats, including PDF
- Color detection engine for image and PDF compression
- 100% royalty-free and world leading Imaging SDK

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

| US Headquarters | European Headquarters | Asia / Pacific Headquarters |
|---|---|---|
| ComponentSource | ComponentSource | ComponentSource |
| 650 Claremore Prof Way | 30 Greyfriars Road | 3F Kojimachi Square Bldg |
| Suite 100 | Reading | 3-3 Kojimachi Chiyoda-ku |
| Woodstock | Berkshire | Tokyo |
| GA 30188-5188 | RG1 1PE | Japan |
| USA | United Kingdom | 102-0083 |

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

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

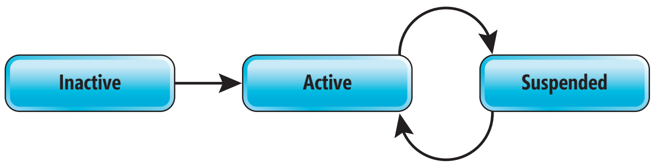


Figure 6 **The State Transitions of ASP.NET Web Sites**

## Easier Use of .NET Framework NuGet Libraries

We intend to deliver .NET Framework versions more frequently to make new features and fixes available sooner. In fact, that's already started with the .NET Framework 4.5.1. Additionally, we use NuGet as a release vehicle to deliver our library features and fixes faster in response to customer feedback.

NuGet is a relatively new package format for the .NET Framework. It provides a standard format for packaging libraries that target one or more .NET profiles and can be consistently consumed by developer tools such as Visual Studio. NuGet.org is the primary NuGet repository and the only one the .NET team uses. Visual Studio comes with an integrated NuGet client for referencing and using NuGet packages in your projects.

We've been shipping .NET libraries on NuGet for the past few years. We've found NuGet is a great way to deliver libraries to a large number of developers and to multiple .NET platforms at the same time. We've improved the NuGet UX in Visual Studio 2013 based on broad feedback, particularly for enterprise scenarios.

**Better Discoverability and Official Support** The Microsoft and .NET NuGet feed was created to improve the discoverability of Microsoft packages. NuGet.org hosts thousands of packages, which could make it challenging to discover the new .NET packages among all the others. This new curated feed provides you with a scoped view of the official Microsoft and .NET packages on NuGet.org. We intend to only add packages to this feed that meet the same quality and support requirements as the .NET Framework. Therefore, you can use these packages in all the same places you use .NET APIs. We've also created a Web view of this feed on the "Microsoft .NET Framework NuGet Packages" page (bit.ly/19D5QLE), hosted on the .NET Framework Blog.

The NuGet team helped us enable this experience by updating their client in Visual Studio to include filtering by curated feeds. **Figure 8** shows the NuGet client in Visual Studio 2013.

**Serviceability** Some enterprise customers told us they were waiting to adopt our NuGet packages until central servicing was offered for these libraries through Microsoft Update. We've added this update capability in the .NET Framework 4.5.1, enabling apps to take advantage of the new feature. Microsoft Update will be an additional release vehicle for .NET NuGet libraries in the unlikely case that we need to quickly and broadly update a library for a critical security issue. Even with this new option in place, we'll continue to use NuGet as a primary vehicle for library updates and fixes.

**Automatic Resolution of Version Conflicts** Apps can reference more than one version of a NuGet package. For desktop and Web apps, you needed to manually resolve version conflicts to ensure that a consistent set of libraries is loaded at run time, which may be challenging and inconvenient. To address that, Visual Studio 2013 automatically configures apps to use the highest referenced version of each library, which solves the issue through a straightforward policy. It also matches the policy already used for Windows Phone and Windows Store apps.

Visual Studio 2013 will automatically generate binding redirects in app.config at build time if version conflicts are found within the app. These binding redirects map each of the versions found for a given library to the highest version found. At run time, your app will use a single version—the highest one referenced—of each library. The main motivation behind this feature was to provide a better experience for consuming NuGet libraries; however, it works for any library. The "How to: Enable and Disable Automatic Binding Redirection" topic in the MSDN Library (bit.ly/1e0i3zW) provides more details about this feature.

## And Much More ...

Up to this point, I've summarized what was delivered in the .NET Framework 4.5.1 release. In the same time frame, we delivered some important new components and features through other release vehicles as well.

**HTTP Client Libraries NuGet Package** The HTTP client library provides a consistent and modern networking .NET API. It lets you write intuitive and asynchronous code (using the await keyword) to access services exposed through HTTP with method names that directly correspond to the HTTP primitives, such as GET, PUT, POST and DELETE. It also provides direct access to HTTP headers and the response body as any of the String, Stream or Byte[] types.

At first, HttpClient was only available for the .NET Framework 4.5 desktop and Windows Store apps. Portable library and Windows Phone app developers had to use HttpWebRequest and HttpWebResponse, with their non-Task-based Asynchronous Pattern (TAP) model. Based on popular demand for portable library and Windows Phone support, we shipped the portable version of the HttpClient library on NuGet to fill the platform gap. As a result, all .NET developers have access to HttpClient, with its TAP-async API.

After the first few versions of the HttpClient NuGet package were released, we added automatic

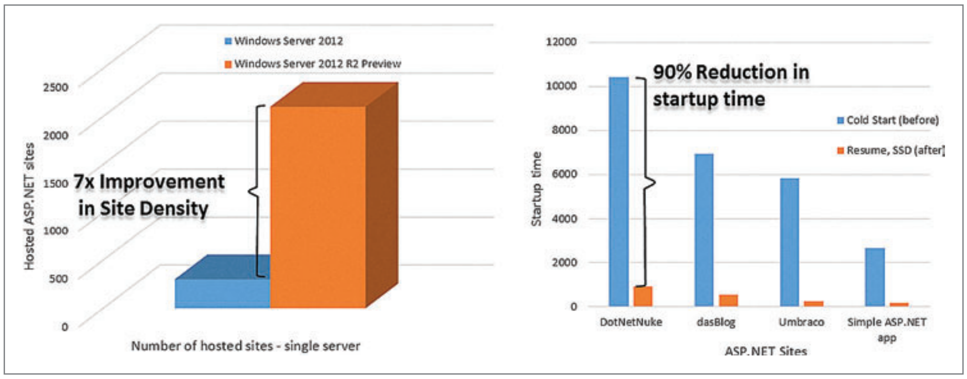


Figure 7 **ASP.NET App Suspension Performance Numbers Seen in the .NET Lab**

/n software

# We didn't invent the Internet...

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

TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- E-Business
  AS2, EDI/X12, NAESB, OFTP …
- Credit Card Processing
  Authorize.Net, TSYS, FDMS …
- Shipping & Tracking
  FedEx, UPS, USPS …
- Accounting & Banking
  QuickBooks, OFX …
- Internet Business
  Amazon, eBay, PayPal …
- Internet Protocols
  FTP, SMTP, IMAP, POP, WebDav …
- Secure Connectivity
  SSH, SFTP, SSL , Certificates …
- Secure Email
  S/MIME, OpenPGP …
- Network Management
  SNMP, MIB, LDAP, Monitoring …
- Compression & Encryption
  Zip, Gzip, Jar, AES …

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

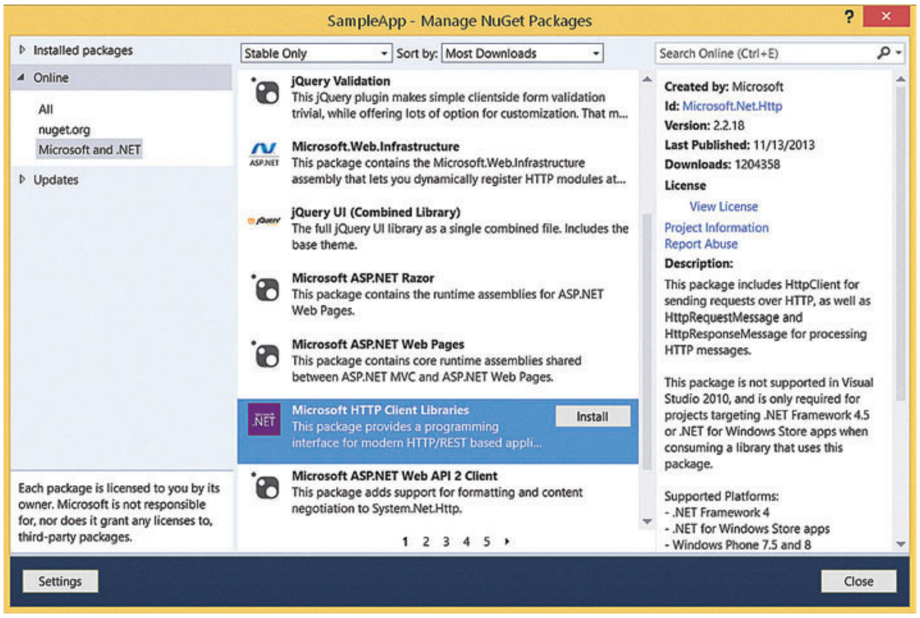
## The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to any application, on any platform, anywhere, and you do the rest. Since 1994, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

connectivity
powered by

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

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


SampleApp - Manage NuGet Packages

Installed packages
Online
All
nuget.org
Microsoft and .NET
Updates

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

Stable Only | Sort by: Most Downloads

**jQuery Validation**
This jQuery plugin makes simple clientside form validation trivial, while offering lots of option for customization. That m...

**Microsoft.Web.Infrastructure**
This package contains the Microsoft.Web.Infrastructure assembly that lets you dynamically register HTTP modules at...

**jQuery UI (Combined Library)**
The full jQuery UI library as a single combined file. Includes the base theme.

**Microsoft ASP.NET Razor**
This package contains the runtime assemblies for ASP.NET Web Pages.

**Microsoft ASP.NET Web Pages**
This package contains core runtime assemblies shared between ASP.NET MVC and ASP.NET Web Pages.

**Microsoft HTTP Client Libraries**
This package provides a programming interface for modern HTTP/REST based appli...
Install

**Microsoft ASP.NET Web API 2 Client**
This package adds support for formatting and content negotiation to System.Net.Http.

1 2 3 4 5 ›

Search Online (Ctrl+E)

**Created by:** Microsoft
**Id:** Microsoft.Net.Http
**Version:** 2.2.18
**Last Published:** 11/13/2013
**Downloads:** 1204358
**License**
View License
Project Information
Report Abuse
**Description:**
This package includes HttpClient for sending requests over HTTP, as well as HttpRequestMessage and HttpResponseMessage for processing HTTP messages.

This package is not supported in Visual Studio 2010, and is only required for projects targeting .NET Framework 4.5 or .NET for Windows Store apps when consuming a library that uses this package.

Supported Platforms:
- .NET Framework 4
- .NET for Windows Store apps
- Windows Phone 7.5 and 8

Settings | Close

Figure 8 **The NuGet Client in Visual Studio 2013**

decompression functionality (bit.ly/13xWATe) in response to feedback. Automatic decompression of HTTP responses helps minimize data requirements, which is useful not only on mobile devices, but also helps with the perception of performance on the desktop.

Microsoft HTTP Client Libraries on NuGet (bit.ly/1a2DPNY) has had great adoption with more than 1.3 million downloads. You can use this package in apps targeting Windows Phone 7.5 and higher, Silverlight 4 and higher, .NET Framework 4 and higher, Windows Store, and Portable Class Libraries (PCL).

**Microsoft Immutable Collections NuGet Package** This is another popular .NET package, which provides easy-to-use, high-performance immutable collections, such as ImmutableList<T> and ImmutableDictionary<TKey, TValue>. Immutable collections, once constructed, don't allow modification. This enables passing immutable types across threads or async contexts without concern about concurrent operations. Even the original creator of the collection can't add or remove items.

The .NET Framework has read-only collection types, such as ReadOnlyCollection<T> and IReadOnlyList<T>. These types guarantee the consumer can't change the data. However, there's no similar guarantee for the provider. This might cause data corruption if the provider and consumer are operating concurrently on different threads. With immutable collection types, you're guaranteed a given instance never changes.

The Microsoft Immutable Collections NuGet package (bit.ly/18xhE5W) is available as a portable library and can be used in desktop and Windows Store apps targeting the .NET Framework 4.5 and higher, PCL, and Windows Phone 8 apps. For more insights and details, I encourage you to start with the "Immutable collections ready for prime time" post (bit.ly/18Y3xp8) on the .NET Framework Blog and the MSDN documentation at bit.ly/189XR9U.

**The New .NET JIT Compiler, RyuJIT** The JIT compiler is one of our key investment areas to improve app performance. The .NET team recently announced the CTP release of the next-generation x64 JIT compiler, code-named "RyuJIT." RyuJIT is twice as fast in compiling code relative to the existing x64 JIT compiler, meaning apps using RyuJIT start up to 30 percent faster depending on the percentage of startup time that's spent in JIT compilation. (Note that time spent in the JIT compiler is only one component of startup time among others, thus the app doesn't start twice as fast because the JIT is twice as fast.) At the same time, RyuJIT doesn't compromise on code quality, and the modern JIT compiler opens up more avenues for future code quality optimizations.

Beyond the performance gains, RyuJIT highlights the .NET team's commitment to customer engagement. Less than a month after the CTP was released, we released an updated version incorporating customer feedback. We'll continue the deep customer engagement and quick cadence of improvements.

We started RyuJIT with a focus on x64 as part of building a first-class cloud platform. As the team moves forward, we'll build support for other architectures. You can get more details about the RyuJIT project and how to download and use the CTP in the "RyuJIT: The next-generation JIT compiler for .NET" post at bit.ly/19RvBHf. I encourage you to try it out and send us feedback.

## Looking for Feedback

In this article, I provided an overview of the new features in the .NET Framework 4.5.1 release. The .NET team delivered many important customer-requested features along with some innovative surprises such as ASP.NET App Suspension and async-aware debugging.

We're shaping the future of .NET with projects that often span multiple .NET releases, in key areas such as the JIT, garbage collection and libraries. In this article, I also provided insights into one of these deep investments, the new .NET JIT compiler, RyuJIT, which was recently shipped as a CTP release.

Note that the .NET team is actively listening for feedback. You can follow .NET news and give the team feedback through the following channels:

- .NET Framework Blog (blogs.msdn.com/b/dotnet)
- Facebook (facebook.com/Dotnet)
- Twitter (twitter.com/DotNet)
- E-mail (dotnet@microsoft.com)
- Visual Studio UserVoice (bit.ly/K26kTu)
- MSDN Forums (bit.ly/19c0uU3) ■

**Gaye Oncul Kok** *is a program manager for the CLR and the .NET Framework at Microsoft, where she works on the .NET Ecosystem team.*

**Thanks** to the following Microsoft technical experts for reviewing this article: *Habib Heydarian, Richard Lander, Immo Landwerth, Andrew Pardoe, Subramanian Ramaswamy and Alok Shriram*

ALEXSYS TEAM®

**WORKFLOW APPLICATIONS | HELP DESK | BUG TRACKING**

# MADE EASY!

**Alexsys Team®** offers ***flexible task management*** software for Windows, Web, and Mobile Devices. Track whatever you need to get the job done - anywhere, anytime on practically any device!

**Thousands are using Alexsys Team® to create workflow solutions and web apps - without coding!** Fortune 500 companies, state, local, DOD, and other federal agencies use Team to manage their tasks. Easily tailor Team to meet your exact requirements - even if they change daily, weekly, or even every minute.

**Alexsys Team® Features Include:**

- Form and Database customization
- Custom workflows
- Role-based security
- Adaptive Rules Engine
- DOD CAC card support
- Time recording
- Automated Escalations, Notifications, and SOAP Messaging
- Supports MS-SQL, MySQL, and Orcale databases

Our renowned tech support team is here to make you and your team a success. Don't have enough resources? Our professional services staff has helped companies large and small use Alexsys Team® at a fraction of the cost of those big consulting firms.

Find out yourself:

**Free Trial and Single User FreePack™ available at Alexcorp.com**

FIND OUT MORE!

**1-888-880-ALEX (2539)**
**ALEXCORP.COM**

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

LIVE LONG AND CODE

SUPPORTED BY

Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA INC

LIVE! Visual Studio
YOUR GUIDE TO THE .NET DEVELOPMENT
UNIVERSE
LIVE!
CHICAGO

This May, developers, software architects, engineers, and designers will blast off in the windy city for four days of unbiased and cutting-edge education on the Microsoft Platform. Live long and code with .NET gurus, launch ideas with industry experts and rub elbows with Microsoft stars in pre-conference workshops, 60+ sessions and fun networking events – all designed to make you better at your job. Plus, explore hot topics like Web API, jQuery, MongoDB, SQL Server Data Tools and more!

**Tracks Include:**

- Visual Studio/.NET
- Windows Client (Windows 8.1/WinRT/WPF)
- JavaScript/HTML5 Client
- ASP.NET
- Cloud Computing
- Windows Phone
- Cross-Platform Mobile Development
- SharePoint/Office
- SQL Server

## CHICAGO 2014

**May 5 – 8 | Chicago Hilton**

**Register by March 5 and Save $300!**

Use promo code VSLFEB2

CONNECT WITH VISUAL STUDIO LIVE!

**twitter.com/vslive – @VSLive**

**facebook.com – Search "VSLive"**

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

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

vslive.com/chicago

# Build More Efficient Windows Store Apps Using JavaScript: Performance

Eric Schmidt

In exploring how to build more efficient Windows Store apps, I first looked at error handling. In this second article, I'll examine several techniques for improving a Windows Store app's performance, focusing on memory usage and HTML UI responsiveness. I'll introduce the new predictable object lifecycle model in the Windows Library for JavaScript on Windows 8.1 (WinJS 2.0). Then, I'll examine Web Workers and the new Scheduler API in WinJS 2.0, both of which complete background tasks without locking the UI. As in the previous article, I'll present both diagnostic tools for finding the problems and solutions for addressing the issues uncovered.

I'll assume you're fairly familiar with building Windows Store apps using JavaScript. If you're relatively new to the platform, I suggest beginning with the basic "Hello World" example (bit.ly/WbVHC) or, for more of a challenge, the "Hilo" sample for JavaScript (bit.ly/SgI0AA). If you haven't read the previous article, you can find it at msdn.microsoft.com/magazine/dn519922.

This article discusses:

- Setting up a sample app that surfaces content from the Web
- Disposing of objects and checking for memory leaks
- Implementing the dispose pattern in WinJS
- Using the Scheduler and Web Worker APIs

Technologies discussed:

Windows 8.1, Windows Library for JavaScript 2.0, Visual Studio 2013

Code download available at:

msdn.microsoft.com/magazine/msdnmag0214

## Setting up the Example

Throughout this article, I draw on specific examples you can test in your own code. You can follow along or download the complete code to peruse at your leisure.

I'm using different test cases than in the previous article, so you'll want to add some new buttons to the global NavBar if you're following along. (You can just start a brand-new Navigation app project if you prefer—that works, too.) The new NavBarCommands are shown in **Figure 1**.

For these test cases, I use the more realistic scenario of an app that surfaces content from the Web. This app fetches data from the United States Library of Congress Print & Photographs Online Catalog Web service (1.usa.gov/1d8nEio). I've written a module that wraps calls to the Web service in promise objects and defines classes for storing the received data. **Figure 2** shows the module, in a file titled searchLOC.js (/js/searchLOC.js).

Remember to link to the searchLOC.js file from default.html at the root of your project before you try to call into it.

## Disposing of Objects

In JavaScript, an object remains in memory as long as it can be reached through a lexical environment or chain of references. Once all references to the object have been removed, the Garbage Collector de-allocates memory from the object. As long as a reference to the object remains, the object stays in memory. A

Figure 1 **Additional NavBarCommands in Default.html**

```
<div data-win-control="WinJS.UI.NavBar">
  <div data-win-control="WinJS.UI.NavBarContainer">

    <!-- Other NavBarCommand elements. -->

    <div id="dispose"
      data-win-control="WinJS.UI.NavBarCommand"
      data-win-options="{
        location: '/pages/dispose/dispose.html',
        icon: 'delete',
        label: 'Dispose pattern in JS'
    }">
    </div>
    <div id="scheduler"
      data-win-control="WinJS.UI.NavBarCommand"
      data-win-options="{
        location: '/pages/scheduler/scheduler.html',
        icon: 'clock',
        label: 'Scheduler'
    }">
    </div>
    <div id="worker"
      data-win-control="WinJS.UI.NavBarCommand"
      data-win-options="{
        location: '/pages/worker/worker.html',
        icon: 'repair',
        label: 'Web worker'
    }">
    </div>
  </div>
</div>
```

memory leak occurs if a reference to an object (and therefore the object itself) remains beyond when it's needed.

One common cause of memory leaks in JavaScript applications are "zombie" objects, which typically occur when a JavaScript object references a DOM object and that DOM object is removed from the document (through a call to removeChild or innerHTML). The corresponding JavaScript object remains in memory, even though the corresponding HTML has vanished:

```
var newSpan = document.createElement("span");
document.getElementById("someDiv").appendChild(newSpan);

document.getElementById("someDiv").innerHTML = "";

WinJS.log && WinJS.log(newSpan === "undefined");
// The previous statement outputs false to the JavaScript console.
// The variable "newSpan" still remains even though the corresponding
// DOM object is gone.
```

For a normal Web page, the life of an object extends only for as long as the browser displays the page. Windows Store apps can't ignore these sorts of memory leaks. Apps commonly use a single HTML page as a content host, where that page persists throughout the app session (which could last for days, or even months). If an app changes state (the user navigates from one page to another, for example, or a ListView control is scrolled so that some items fall out of visibility) without cleaning up memory allocated to unneeded JavaScript objects, that memory can become unavailable to the app.

## Checking for Memory Leaks

Luckily, Visual Studio 2013 has new features that can help developers track down memory leaks—in particular the Performance and Diagnostics window. For this test case and the next, I'll demonstrate a couple of the tools that it surfaces.

For this test case, I'll add a custom control to my solution that purposely allows memory leaks. This control, named SearchLOCControl (/js/SearchLOCControl.js), creates a search text box and then displays results after a response to a query has been received.

Figure 2 **Accessing the Print & Photographs Online Catalog Web Service**

```
(function () {
  "use strict";

  var baseUrl = "http://loc.gov/pictures/"
  var httpClient = new Windows.Web.Http.HttpClient();

  function searchPictures(query) {
    var url = baseUrl + "search/?q=" + query + "&fo=json";
    var queryURL = encodeURI(url);

    return httpClient.getStringAsync(
      new Windows.Foundation.Uri(queryURL)).
      then(function (response) {

        return JSON.parse(response).results.map(function (result) {
          return new SearchResult(result);
        });
     });
  }

  function getCollections() {
    var url = baseUrl + "?fo=json";

    return httpClient.getStringAsync(new Windows.Foundation.Uri(url)).
      then(function (response) {

         return JSON.parse(response).featured.
           map(function (collection) {
             return new Collection(collection);
         });
      });
  }

  function getCollection(collection) {
    var url = baseUrl + "search/?co=" + collection.code + "&fo=json";
    var queryUrl = encodeURI(url);

    return httpClient.getStringAsync(new Windows.Foundation.Uri(queryurl)).
      then(function (response) {

        collection.pictures = JSON.parse(response).
          results.map(function (picture) {
            return new SearchResult(picture);
        });

        return collection;
      });
  }

  function Collection(info) {
    this.title = info.title;
    this.featuredThumb = info.thumb_featured;
    this.code = info.code;
    this.pictures = [];
  }

  function SearchResult(data) {
    this.pictureThumb = data.image.thumb;
    this.title = data.title;
    this.date = data.created_published_date;
  }

  WinJS.Namespace.define("LOCPictures", {
    Collection: Collection,
    searchPictures: searchPictures,
    getCollections: getCollections,
    getCollection: getCollection
  });
})();
```

Figure 3 **Custom SearchLOCControl**

```
(function () {
  "use strict";

  WinJS.Namespace.define("SearchLOCControl", {
    Control: WinJS.Class.define(function (element) {
      this.element = element;
      this.element.winControl = this;

      var htmlString = "<h3>Library of Congress Picture Search</h3>" +
        "<div id='searchQuery' data-win-control='WinJS.UI.SearchBox'" +
          "data-win-options='{ placeholderText: \"Browse pictures\" }'></div>" +
          "<br/><br/>" +
          "<div id='searchResults' class='searchList'></div>" +
          "<div id='searchResultsTemplate'" +
            "data-win-control='WinJS.Binding.Template'>" +
            "<div class='searchResultsItem'>" +
              "<img src='#' data-win-bind='src: pictureThumb' />" +
              "<div class='details'>" +
                "<p data-win-bind='textContent: title'></p>" +
                "<p data-win-bind='textContent: date'></p>" +
              "</div>" +
            "</div>"+
        "</div>";
   // NOTE: This is an unusual technique for accomplishing this
   // task. The code here is written for extreme brevity.
      MSApp.execUnsafeLocalFunction(function () {
        $(element).append(htmlString);
        WinJS.UI.processAll();
      });

      this.searchQuery = $("#searchQuery")[0];
      searchQuery.winControl.addEventListener("querysubmitted", this.submitQuery);
    }, {
      submitQuery: function (evt) {
        var queryString = evt.target.winControl.queryText;
        var searchResultsList = $("#searchResults")[0];
        $(searchResultsList).append("<progress class='win-ring'></progress>");

        if (queryString != "") {
          var searchResults = LOCPictures.searchPictures(queryString).
            then(function (response) {
              var searchList = new WinJS.Binding.List(response),
                searchListView;

              if (searchResultsList.winControl) {
                searchListView = searchResultsList.winControl;
                searchListView.itemDataSource = searchList.dataSource;
              }
              else {
                searchListView = new WinJS.UI.ListView(searchResultsList, {
                  itemDataSource: searchList.dataSource,
                  itemTemplate: $("#searchResultsTemplate")[0],
                  layout: { type: WinJS.UI.CellSpanningLayout}
                });
              }

              WinJS.UI.process(searchListView);
            });
          }
        }
      })
    })
})();
```

**Figure 3** shows the code for SearchLOCControl.js. Again, remember to link to this new JavaScript file from default.html.

Note that I use jQuery to build my custom control, which I add to my solution using the NuGet Package Manager. Once you've downloaded the NuGet package to your solution, you'll need to manually add a reference to the jQuery library in default.html.

The SearchLOCControl relies on some styling I've added to default.css (/css/default.css), which is shown in **Figure 4**.

Now I add a new page control named dispose.html (/pages/dispose/dispose.html) to the solution and add the following HTML markup inside the <section> tag of dispose to create the custom control:

```
<button id="dispose">Dispose</button><br/><br/>
<div id="searchControl" data-win-control="SearchLOCControl.Control"></div>
```

Finally, I add code to the PageControl.ready event handler in the dispose.js file (/pages/dispose/dispose.js) that naively destroys the control and creates a memory leak by setting the innerHTML of the control's host <div> to an empty string, as shown in **Figure 5**.

Figure 4 **Styling Added to Default.css**

```
.searchList {
  height: 700px !important;
  width: auto !important;
}

.searchResultsItem {
  display: -ms-inline-grid;
  -ms-grid-columns: 200px;
  -ms-grid-rows: 150px 150px
}

  .searchResultsItem img {
    -ms-grid-row: 1;
    max-height: 150px;
    max-width: 150px;
  }

  .searchResultsItem .details {
    -ms-grid-row: 2;
  }
```

Now I can test the control's memory usage. The Performance and Diagnostics window provides several tools for measuring the performance of a Windows Store app, including CPU sampling, app energy consumption, UI responsiveness and JavaScript function timing. (You can read more about these tools on the Visual Studio team's blog at bit.ly/1bESdOH.) If it's not already visible, you'll need to open the Performance and Diagnostics pane, either through the Debug menu (Visual Studio Express 2013 for Windows) or through the Analyze menu (Visual Studio Professional 2013 and Visual Studio Ultimate 2013).

For this test, I use the JavaScript memory monitoring tool. Here are the steps for performing the test:

1. In the Performance and Diagnostics window, select JavaScript Memory and then click Start. The project then runs in debugging mode. If prompted by a User Account Control dialog box, click Yes.

Figure 5 **Code in Dispose.js to "Destroy" the Custom Control**

```
(function () {
  "use strict";

  WinJS.UI.Pages.define("/pages/dispose/dispose.html", {
    ready: function (element, options) {
      WinJS.UI.processAll();

      $("#dispose").click(function () {
        var searchControl = $("#searchControl")[0];
        searchControl.innerHTML = "";
      });
    }

  // Other page control code.

  });
})();
```

# .NET TOOLS FOR DEV PROS

Whether you're building the most modern touch-enabled apps or maintaining and updating legacy applications, our flagship product, Studio Enterprise, helps to deliver rich, responsive, desktop and web apps on time and under budget.

DOWNLOAD YOUR FREE TRIAL

▸ componentone.com/se

ComponentOne®
a division of GrapeCity®

2. With the app project running, navigate to the dispose page, then switch to the desktop. In Visual Studio, in the current diagnostic session (a tab titled "Report*.diagsession"), click Take Heap Snapshot.
3. Switch back to the running app. In the search box, enter a query (for example, "Lincoln") and then press Enter. A ListView control appears that displays the image search results.
4. Switch back to the desktop. In Visual Studio, in the current diagnostic session (a tab titled "Report*.diagsession"), click Take Heap Snapshot.
5. Switch back to the running app. Click the Dispose button. The custom control disappears from the page.
6. Switch back to the desktop. In Visual Studio, in the current diagnostic session (a tab titled "Report*.diagsession") click Take Heap Snapshot and then click Stop. There are now three snapshots listed in the diagnostic session, as shown in **Figure 6**.

With the diagnostics data in hand, I can analyze the memory usage of the custom control. From a quick glance at the diagnostics session, I suspect that "disposing" the control didn't free up all of the memory associated with it.

In the report, I can examine the JavaScript objects on the heap for each snapshot. I want to know what remained in memory after the custom control was removed from the DOM. I'll click the link associated with the number of objects on the heap in the third snapshot (Snapshot #3 in **Figure 6**).

First I'll look at the Dominators view, which shows a sorted list of the objects by retained size. The objects consuming the most memory that are potentially easiest to free are listed at the top. In the Dominators view, I see a reference to the <div> with an id value "searchControl." When I expand it, I see that the search box, ListView and data associated with it are all in memory.

When I right-click the row for the searchControl <div> and select Show in root view, I see the event handlers for the button clicks are still in memory, too, as **Figure 7** shows.

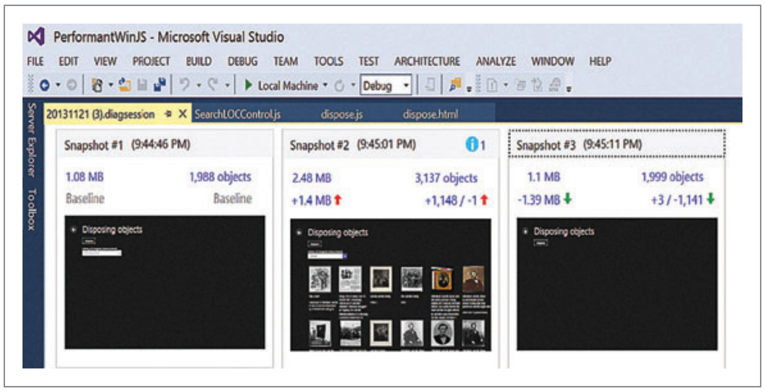
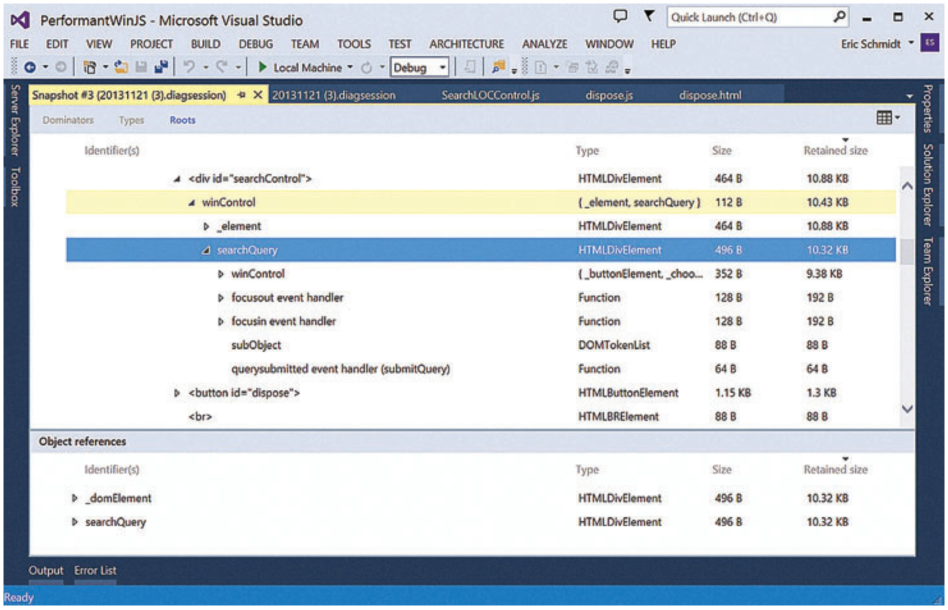


Figure 6 **Memory Usage Before Implementing the Dispose Pattern**



| Identifier(s) | Type | Size | Retained size |
|---|---|---|---|
| ◢ <div id="searchControl"> | HTMLDivElement | 464 B | 10.88 KB |
| ◢ winControl | { _element, searchQuery } | 112 B | 10.43 KB |
| ▷ _element | HTMLDivElement | 464 B | 10.88 KB |
| ◢ searchQuery | HTMLDivElement | 496 B | 10.32 KB |
| ▷ winControl | { _buttonElement, _choo... | 352 B | 9.38 KB |
| ▷ focusout event handler | Function | 128 B | 192 B |
| ▷ focusin event handler | Function | 128 B | 192 B |
| subObject | DOMTokenList | 88 B | 88 B |
| querysubmitted event handler (submitQuery) | Function | 64 B | 64 B |
| ▷ <button id="dispose"> | HTMLButtonElement | 1.15 KB | 1.3 KB |
| <br> | HTMLBRElement | 88 B | 88 B |

Object references

| Identifier(s) | Type | Size | Retained size |
|---|---|---|---|
| ▷ _domElement | HTMLDivElement | 496 B | 10.32 KB |
| ▷ searchQuery | HTMLDivElement | 496 B | 10.32 KB |

Figure 7 **Unattached Event Handler Code Taking up Memory**

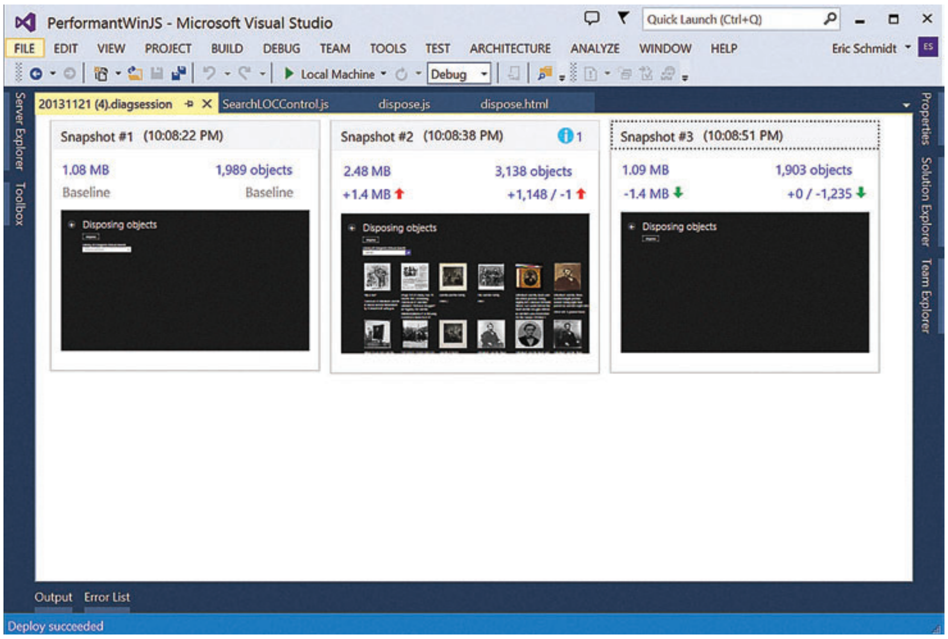
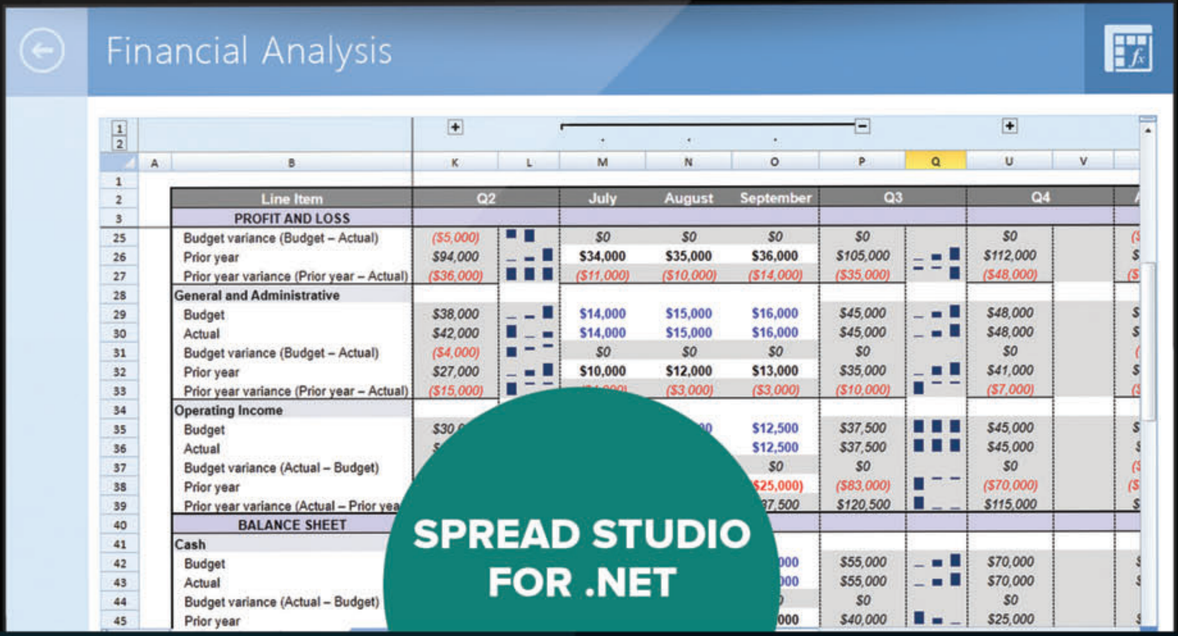


Figure 8 **Memory Usage After Implementing Dispose**

# THE EXPERTS IN SPREADSHEETS

When you need the power of a spreadsheet combined with the functionally of an advanced data grid, you need Spread, the world's #1 selling spreadsheet component for Microsoft Visual Studio development.



| | Line Item | Q2 | | July | August | September | Q3 | | Q4 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | PROFIT AND LOSS | | | | | | | | |
| 25 | Budget variance (Budget – Actual) | ($5,000) | | $0 | $0 | $0 | $0 | | $0 |
| 26 | Prior year | $94,000 | | $34,000 | $35,000 | $36,000 | $105,000 | | $112,000 |
| 27 | Prior year variance (Prior year – Actual) | ($36,000) | | ($11,000) | ($10,000) | ($14,000) | ($35,000) | | ($48,000) |
| 28 | General and Administrative | | | | | | | | |
| 29 | Budget | $38,000 | | $14,000 | $15,000 | $16,000 | $45,000 | | $48,000 |
| 30 | Actual | $42,000 | | $14,000 | $15,000 | $16,000 | $45,000 | | $48,000 |
| 31 | Budget variance (Budget – Actual) | ($4,000) | | $0 | $0 | $0 | $0 | | $0 |
| 32 | Prior year | $27,000 | | $10,000 | $12,000 | $13,000 | $35,000 | | $41,000 |
| 33 | Prior year variance (Prior year – Actual) | ($15,000) | | [illegible] | ($3,000) | ($3,000) | ($10,000) | | ($7,000) |
| 34 | Operating Income | | | | | | | | |
| 35 | Budget | $30 | | | | $12,500 | $37,500 | | $45,000 |
| 36 | Actual | | | | | $12,500 | $37,500 | | $45,000 |
| 37 | Budget variance (Actual – Budget) | | | | | $0 | $0 | | $0 |
| 38 | Prior year | | | | | $25,000) | ($83,000) | | ($70,000) |
| 39 | Prior year variance (Actual – Prior yea | | | | | 37,500 | $120,500 | | $115,000 |
| 40 | BALANCE SHEET | | | | | | | | |
| 41 | Cash | | | | | | | | |
| 42 | Budget | | | | | 000 | $55,000 | | $70,000 |
| 43 | Actual | | | | | 000 | $55,000 | | $70,000 |
| 44 | Budget variance (Actual – Budget) | | | | | | $0 | | $0 |
| 45 | Prior year | | | | | 000 | $40,000 | | $25,000 |

Spread provides a flexible and familiar spreadsheet/grid architecture, advanced charting, and a powerful formula library that is ideal for creating financial modeling and risk analysis, budgeting, insurance, scientific, and many other applications.

AVAILABLE FOR:
**Windows Forms · ASP.NET · WPF · Silverlight · WinRT**

- Read Microsoft Excel files and/or generate Excel output from your business application
- Provide Excel like characteristics, such as complete formula calculation support, enhanced filtering, sorting, conditional formatting, cell types, sheets, and more
- Create Complex Form/Data-entry Layouts that include many fields and calculations, such as insurance forms or tax forms
- Add Data Visualization, Analysis, and Dashboards

ComponentOne®
a division of GrapeCity®

DOWNLOAD YOUR FREE TRIAL
▸ componentone.com

Thankfully, I can fix this easily with only a few changes to my code.

## Implementing the Dispose Pattern in WinJS

In WinJS 2.0, all of the WinJS controls implement a "dispose" pattern to address the issue of memory leaks. Whenever a WinJS control falls out of scope (for example, when the user navigates to another page), WinJS cleans up all references to it. The control is marked for disposal, meaning that the internal Garbage Collector knows to release all the memory allocated to the object.

The dispose pattern in WinJS has three important characteristics that a control must provide in order to be properly disposed:

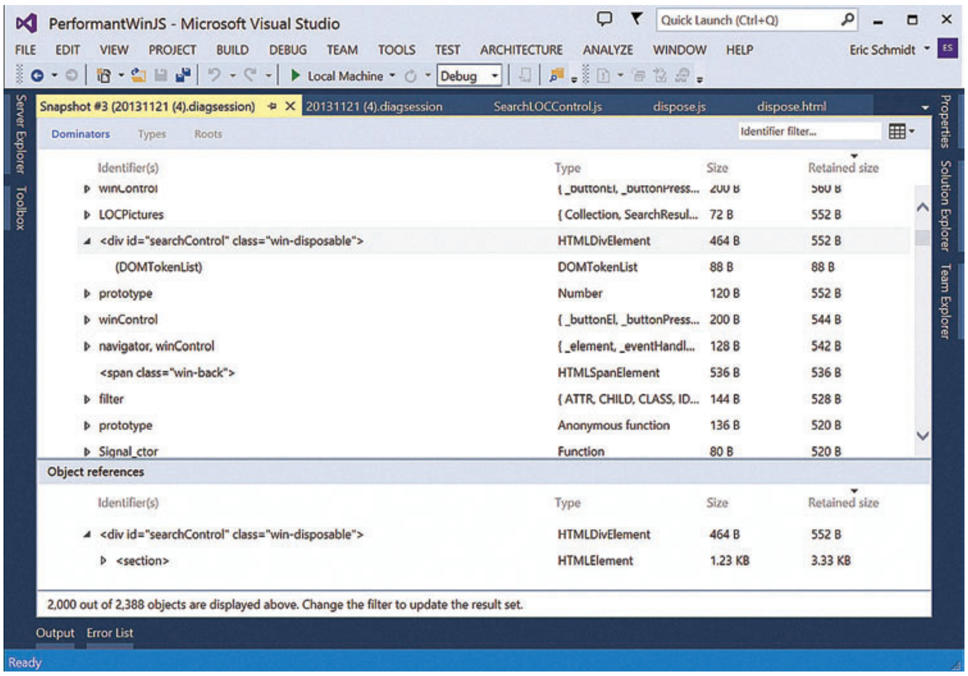
- The top-level container DOM element must have the CSS class "win-disposable."
- The control's class must include a field called _disposed that is initially set to false. You can add this member to a control (along with the win-disposable CSS class) by calling WinJS.Utilities.markDisposable.
- The JavaScript class that defines the control must expose a "dispose" method. In the dispose method:
    - All memory allocated to objects associated with the control needs to be released.
    - All event handlers need to be detached from the child DOM objects.
    - All children of the control must have their dispose methods called. The best way to do this is by calling WinJS.Utilities.disposeSubTree on the host element.
    - All outstanding promises that might be referenced within the control need to be canceled (by calling the Promise.cancel method and then nulling the variable out).

So, in the constructor function for SearchLOCControl.Control, I add the following lines of code:

```
this._disposed = false;
WinJS.Utilities.addClass(element,
"win-disposable");
```

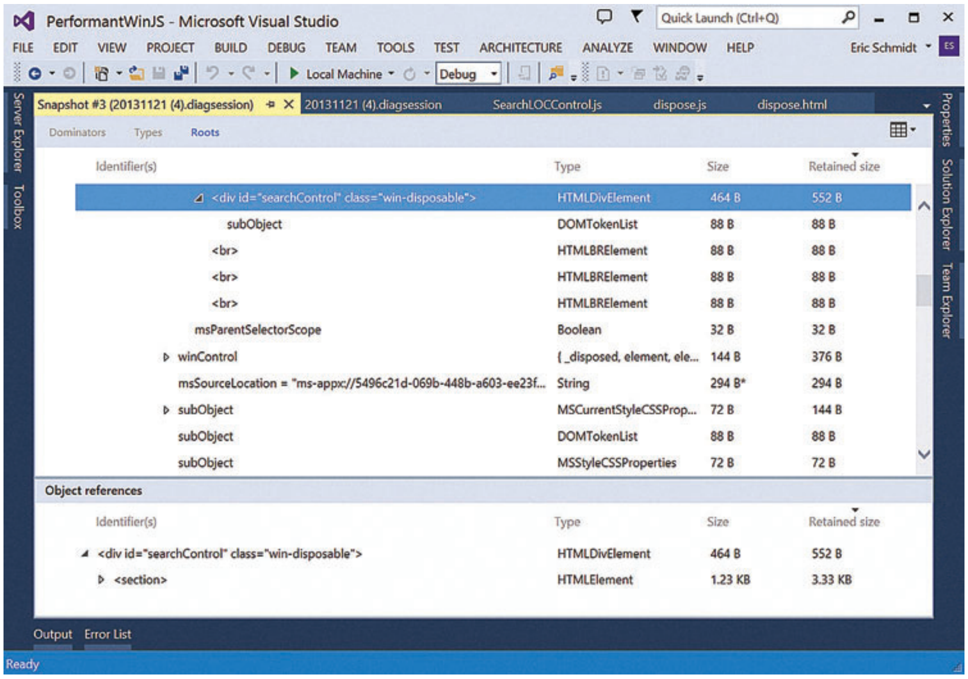
Next, inside the SearchLOC-Control class definition (the call to WinJS.Class.define), I add a new instance member named dispose. Here's the code for the dispose method:

```
dispose: function () {
  this._disposed = true;

  this.searchQuery.winControl.removeEventListener("querysubmitted",
    this.submitQuery);

  WinJS.Utilities.disposeSubTree(this.element);

  this.searchQuery = null;
  this._element.winControl = null;
  this._element = null;
}
```



```
PerformantWinJS - Microsoft Visual Studio                     Quick Launch (Ctrl+Q)
FILE  EDIT  VIEW  PROJECT  BUILD  DEBUG  TEAM  TOOLS  TEST  ARCHITECTURE  ANALYZE  WINDOW  HELP     Eric Schmidt  ES
Local Machine  Debug
Snapshot #3 (20131121 (4).diagsession)   20131121 (4).diagsession   SearchLOCControl.js   dispose.js   dispose.html
Dominators   Types   Roots                                                  Identifier filter...

    Identifier(s)                                           Type                          Size      Retained size
  ▷ winControl                                              { _buttonEl, _buttonPress...  200 B     560 B
  ▷ LOCPictures                                             { Collection, SearchResul...  72 B      552 B
  ◢ <div id="searchControl" class="win-disposable">         HTMLDivElement                464 B     552 B
      (DOMTokenList)                                        DOMTokenList                  88 B      88 B
  ▷ prototype                                               Number                        120 B     552 B
  ▷ winControl                                              { _buttonEl, _buttonPress...  200 B     544 B
  ▷ navigator, winControl                                   { _element, _eventHandl...    128 B     542 B
    <span class="win-back">                                 HTMLSpanElement               536 B     536 B
  ▷ filter                                                  { ATTR, CHILD, CLASS, ID...   144 B     528 B
  ▷ prototype                                               Anonymous function            136 B     520 B
  ▷ Signal_ctor                                             Function                      80 B      520 B
Object references
    Identifier(s)                                           Type                          Size      Retained size
  ◢ <div id="searchControl" class="win-disposable">         HTMLDivElement                464 B     552 B
    ▷ <section>                                             HTMLElement                   1.23 KB   3.33 KB

2,000 out of 2,388 objects are displayed above. Change the filter to update the result set.
Output   Error List
Ready
```

Figure 9 **Dominators View After Implementing Dispose**



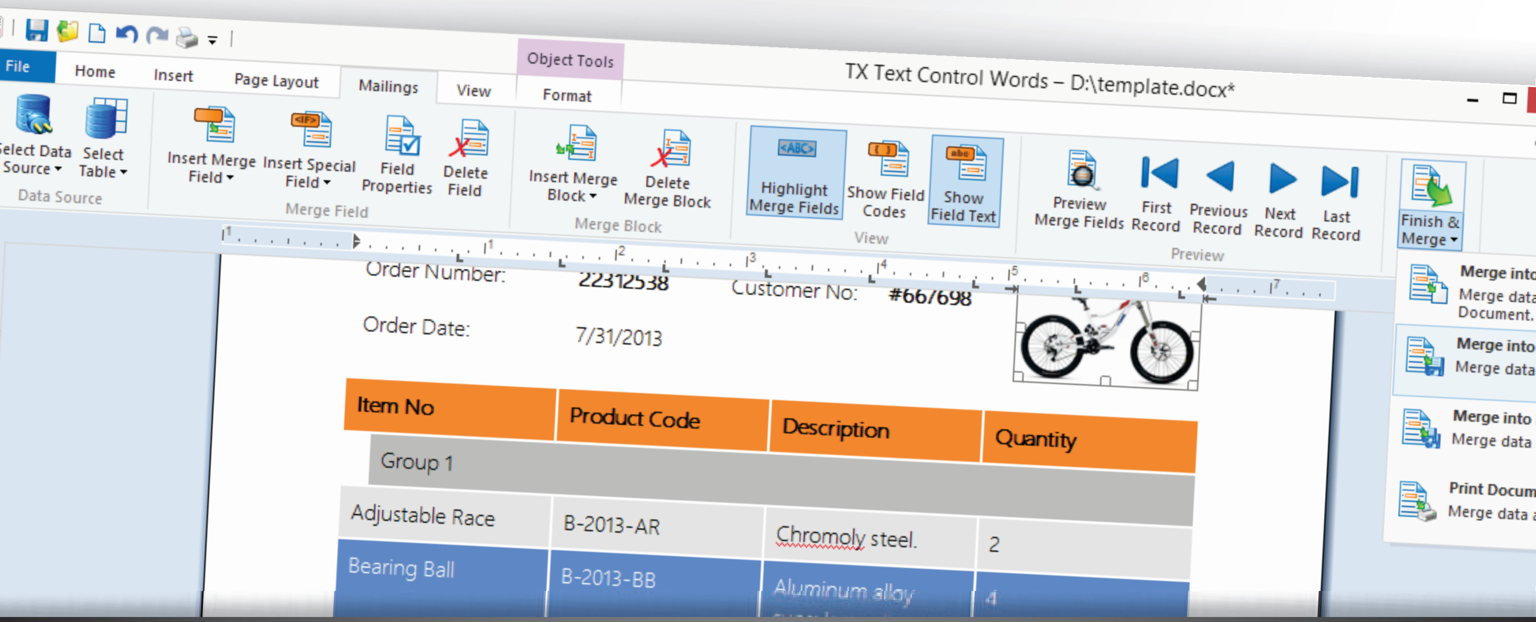
```
PerformantWinJS - Microsoft Visual Studio                     Quick Launch (Ctrl+Q)
FILE  EDIT  VIEW  PROJECT  BUILD  DEBUG  TEAM  TOOLS  TEST  ARCHITECTURE  ANALYZE  WINDOW  HELP     Eric Schmidt  ES
Local Machine  Debug
Snapshot #3 (20131121 (4).diagsession)   20131121 (4).diagsession   SearchLOCControl.js   dispose.js   dispose.html
Dominators   Types   Roots

    Identifier(s)                                               Type                          Size      Retained size
          ◢ <div id="searchControl" class="win-disposable">     HTMLDivElement                464 B     552 B
              subObject                                         DOMTokenList                  88 B      88 B
            <br>                                                HTMLBRElement                 88 B      88 B
            <br>                                                HTMLBRElement                 88 B      88 B
            <br>                                                HTMLBRElement                 88 B      88 B
          msParentSelectorScope                                 Boolean                       32 B      32 B
        ▷ winControl                                            { _disposed, element, ele...  144 B     376 B
          msSourceLocation = "ms-appx://5496c21d-069b-448b-a603-ee23f...  String             294 B*    294 B
        ▷ subObject                                             MSCurrentStyleCSSProp...      72 B      144 B
          subObject                                             DOMTokenList                  88 B      88 B
          subObject                                             MSStyleCSSProperties          72 B      72 B
Object references
    Identifier(s)                                               Type                          Size      Retained size
  ◢ <div id="searchControl" class="win-disposable">             HTMLDivElement                464 B     552 B
    ▷ <section>                                                 HTMLElement                   1.23 KB   3.33 KB

Output   Error List
Ready
```

Figure 10 **Roots View After Implementing Dispose**

TX TEXT CONTROL®
word processing components

# Creating a report is as easy as writing a letter

- Reuse MS Word documents as your reporting templates
- Create encrypted and print-ready Adobe PDF and PDF/A
- Royalty-free WYSIWYG template designer
- Powerful and dynamic 2D/3D charting support
- Easy database connections and master-detail nested blocks
- 1D/2D barcode support including QRCode, IntelligentMail, EAN



Order Number: 22312538 Customer No: #66/698

Order Date: 7/31/2013

| Item No | Product Code | Description | Quantity |
|---|---|---|---|
| Group 1 | | | |
| Adjustable Race | B-2013-AR | Chromoly steel. | 2 |
| Bearing Ball | B-2013-BB | Aluminum alloy | 4 |

www.textcontrol.com/reporting

txtextcontrol

US: +1 855-533-TEXT
EU: +49 421 427 067-10

Reporting | Rich Text Editing | Spell Checking | Barcodes | PDF Reflow

Visual Studio Microsoft Partner

Generally speaking, you don't need to clean up any variable in your code that's entirely contained within the code for the element. When the code for the control goes away, so do all of the internal variables. However, if the code for the control references something outside of itself—such as a DOM element, for example—then that reference will need to be nulled out.

Finally, I add an explicit call to the dispose method in dispose.js (/pages/dispose/dispose.js). Here's the updated click event handler for the button in dispose.html:

```
$("#dispose").click(function () {
  var searchControl = $("#searchControl")[0];
  searchControl.winControl.dispose();
  searchControl.innerHTML = "";
});
```

Now when I run the same JavaScript memory test, the diagnostics session looks much better (see **Figure 8**).

Examining the memory heap, I can see the "searchControl" <div> no longer has child elements associated with it (see **Figure 9**). None of the sub controls remain in memory and the associated event handlers are gone, too (see **Figure 10**).

## Improving Responsiveness: Scheduler and Web Workers

Apps can become unresponsive when the UI is waiting to be updated based on an external process. For example, if an app makes multiple requests to a Web service in order to populate a UI control, the control—the entire UI, for that matter—can get stuck while waiting on the requests. This can cause the app to stutter or seem unresponsive.

To demonstrate this, I create another test case where I populate a Hub control with the "featured collections" provided by the Library of Congress Web service. I add a new Page Control named scheduler.html for the test case to my project (/pages/scheduler/scheduler.js). In the HTML for the page, I declare a Hub control that contains six HubSection controls (one for each featured collection). The HTML for the Hub control inside the <section> tags in scheduler.html is shown in **Figure 11**.

Next, I get the featured collections data from the Web service. I'll add a new file named data.js to my solution (/js/data.js) that calls the Web service and returns a WinJS.Binding.List object. **Figure 12** shows the code for getting the featured collections data. Again, remember to link to data.js from default.html.

Now I need to insert the data into the Hub control. In the scheduler.js file (/pages/scheduler/scheduler.js), I'll add some code to the PageControl.ready function and define a new function, populateSection. The complete code is shown in **Figure 13**.

Note in **Figure 13** that I capture a reference to the promise returned by the call to Data.getFeaturedCollections and then explicitly cancel the promise when the page unloads. This avoids a possible race condition in a scenario where the user navigates to the page and then navigates away before the call to getFeaturedCollections has returned.

When I press F5 and navigate to scheduler.html, I notice the Hub control populates slowly after the page loads. It may be merely annoying on my machine, but on less powerful machines the lag could be significant.

Visual Studio 2013 includes tools for measuring the responsiveness of the UI in a Windows Store app. In the Performance and Diagnostics pane, I select the HTML UI Responsiveness test and then click Start. After the app starts running, I navigate to scheduler.html and watch the results appear in the Hub control. Once I've completed the task, I switch back to the

Figure 11 **Hub and HubSection Controls Declared in Scheduler.html**

```
<div id="featuredHub" data-win-control="WinJS.UI.Hub">
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 1'
    }"
    class="section">
  </div>
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 2'
    }"
    class="section">
  </div>
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 3'
    }"
    class="section">
  </div>
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 4'
    }"
    class="section">
  </div>
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 5'
    }"
    class="section">
  </div>
  <div data-win-control="WinJS.UI.HubSection"
    data-win-options="{
      header: 'Featured Collection 6'
    }"
    class="section">
  </div>
</div>
```

Figure 12 **Getting the Data from the Web Service**

```
(function () {
  "use strict";

  var data = LOCPictures.getCollections().
  then(function (message) {
    var data = message;
    var dataList = new WinJS.Binding.List(data);

    var collectionTasks = [];
    for (var i = 0; i < 6; i++) {
      collectionTasks.push(getFeaturedCollection(data[i]));
    }

    return WinJS.Promise.join(collectionTasks).then(function () {
      return dataList;
    });
  });

  function getFeaturedCollection(collection) {
    return LOCPictures.getCollection(collection);
  }

 WinJS.Namespace.define("Data", {
   featuredCollections: data
 });
})();
```

## NuoDB: The Distributed Database
## Scalable. SQL. Cloud-Ready.

- ✓ **Higher Performance:**
  64 bit support for Windows Server, Windows 7 and 8

- ✓ **Integrated Developer Experience:**
  Full support for Visual Studio 2012, LINQ and Entity Framework

- ✓ **Scale-Out Performance:**
  Scale elastically. New machines are introduced to a running database and become effective immediately

- ✓ **Cloud-Ready:**
  Azure compatibility for running / deploying easily in the cloud

**Free Download: www.nuodb.com/free-download**

**“NuoDB announced a bunch of Microsoft enhancements, many of which are great for developers.”**

Kevin Kline Blog
July 8, 2013

NUODB®

Figure 13 **Populating the Hub Control Dynamically**

```
(function () {
  "use strict";

  var dataRequest;

  WinJS.UI.Pages.define("/pages/scheduler/scheduler.html", {
    ready: function (element, options) {
      performance.mark("navigated to scheduler");

      dataRequest = Data.featuredCollections.
        then(function (collections) {
          performance.mark("got collection");

          var hub = element.querySelector("#featuredHub");
            if (!hub) { return; }

            var hubSections = hub.winControl.sections,
            hubSection, collection;

            for (var i = 0; i < hubSections.length; i++) {
              hubSection = hubSections.getItem(i);
              collection = collections.getItem(i);
              populateSection(hubSection, collection);
            }
        });
    },

    unload: function () {
      dataRequest.cancel();
    }

    // Other PageControl members ...
  });

  function populateSection(section, collection) {
    performance.mark("creating a hub section");
    section.data.header = collection.data.title;

    var contentElement = section.data.contentElement;
    contentElement.innerHTML = "";

    var pictures = collection.data.pictures;
    for (var i = 0; i < 6; i++) {

      $(contentElement).append("<img src='" + pictures[i].pictureThumb + "' />");
      (i % 2) && $(contentElement).append("<br/>")
    }
    }
})();
```

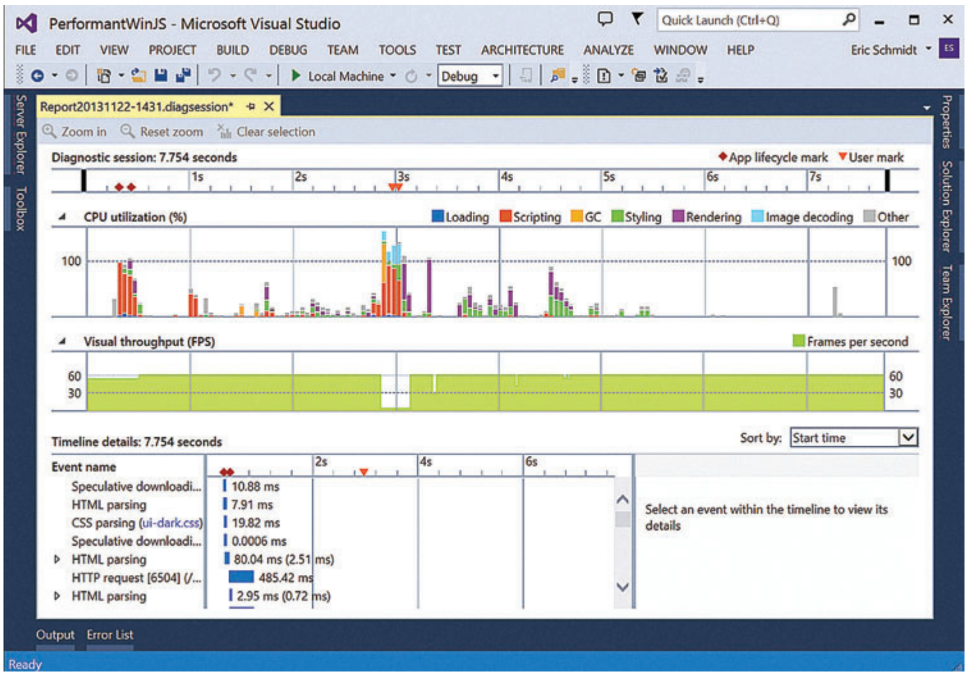
Desktop and then click Stop in the diagnostics session tab. **Figure 14** displays the results.

I see the frame rate dropped to 3 FPS for roughly half a second. I select the period of low frame rate to see more details (see **Figure 15**).

At this point in the timeline (**Figure 15**), the UI thread is absorbed in running scheduler.js. If you look closely at the timeline details, you see several user marks (orange "tick" marks). These indicate specific calls to performance.mark in the code. In scheduler.js, the first call to performance.mark occurs when scheduler.html loads. Populating each HubSection control with content invokes a subsequent call. From the results, more than half of the time spent evaluating scheduler.js occurred between when I navigated to the page (the first user mark) and when the sixth HubSection was populated with images (the last user mark).

(Keep in mind results will vary depending on your hardware. The HTML UI responsiveness tests shown in this article were run on a Microsoft Surface Pro with a third-generation Intel Core i5-3317U processor, running at 1.7Ghz, and Intel HD Graphics 400. )

To reduce lag, I might refactor my code so the HubSection controls are populated in a staggered manner. Users see content in the app soon after they navigate to it. The content for the first one to two hub sections should load immediately after navigation and the other HubSections can load afterward.



| Event name | |
|---|---|
| Speculative downloadi... | 10.88 ms |
| HTML parsing | 7.91 ms |
| CSS parsing (ui-dark.css) | 19.82 ms |
| Speculative downloadi... | 0.0006 ms |
| HTML parsing | 80.04 ms (2.51 ms) |
| HTTP request [6504] (/... | 485.42 ms |
| HTML parsing | 2.95 ms (0.72 ms) |

Figure 14 **HTML UI Responsiveness for Scheduler.html**

## Scheduler

JavaScript is a single-threaded environment, meaning everything is on the UI thread. In WinJS 2.0, Microsoft introduced the WinJS.Utilities.Scheduler to organize work performed on the UI thread (see bit.ly/1bFbpfb for more information).

The Scheduler creates a single queue of jobs to be run on the UI thread in the app. Jobs are completed based on priority, where higher priority jobs can preempt or postpone lower-priority jobs. Jobs are scheduled around actual user interaction on the UI thread, where the Scheduler slices up the time between calls and completes as many of the queued jobs as it can.

As mentioned, the scheduler executes jobs based on their priority, as set using the WinJS.Utilities.Scheduler.Priority

Distrubuted teams collaboratively fixing bugs inside Visual Studio.

## EFFICIENT COLLABORATION FOR DEVELOPMENT TEAMS

1 INCREASE SPEED & EFFICIENCY

2 IMPROVE GLOBAL TRAINING

3 HIRE BETTER TEAMS & ONBOARD FASTER

4 ENSURE BEST PRACTICES

*Designed to improve collaborative development and increase project awareness across distributed teams, VS Anywhere creates a lean and clean workflow that reduces the errors and inefficiencies that slow teams down.*

REDUCE COSTS ASSOCIATED WITH BUG FIXING, KNOWLEDGE TRANSFER AND DISTRIBUTED TEAMWORK BY 70% OR MORE

LEARN MORE AT VSANYWHERE.COM

VS Anywhere

Efficient Collaboration for Development Teams

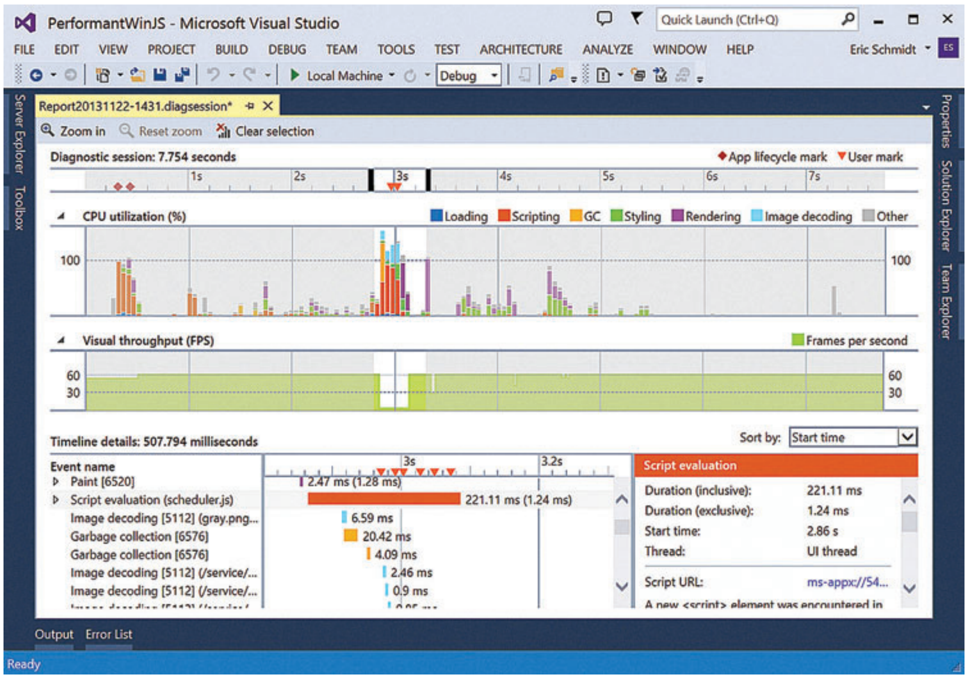

PerformantWinJS - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Eric Schmidt
Local Machine
Debug
Report20131122-1431.diagsession*
Zoom in
Reset zoom
Clear selection
Diagnostic session: 7.754 seconds
App lifecycle mark
User mark
CPU utilization (%)
Loading
Scripting
GC
Styling
Rendering
Image decoding
Other
Visual throughput (FPS)
Frames per second
Timeline details: 507.794 milliseconds
Sort by: Start time
Event name
Paint [6520]
Script evaluation (scheduler.js)
Image decoding [5112] (gray.png...
Garbage collection [6576]
Garbage collection [6576]
Image decoding [5112] (/service/...
Image decoding [5112] (/service/...
2.47 ms (1.28 ms)
221.11 ms (1.24 ms)
6.59 ms
20.42 ms
4.09 ms
2.46 ms
0.9 ms
Script evaluation
Duration (inclusive): 221.11 ms
Duration (exclusive): 1.24 ms
Start time: 2.86 s
Thread: UI thread
Script URL: ms-appx://54...
A new <script> element was encountered in
Server Explorer
Toolbox
Properties
Solution Explorer
Team Explorer
Output
Error List
Ready

Figure 15 **Timeline Details Where the UI Thread Evaluates Scheduler.js**

enumeration. The enumeration has seven values (in descending order): max, high, aboveNormal, normal, belowNormal, idle and min. Jobs of equal priority are run on a first-in, first-out basis.

Turning to the test case, I create a job on the Scheduler to populate each HubSection when scheduler.html loads. For each HubSection, I call Scheduler.schedule and pass in a function that populates the HubSection. The first two jobs are run at normal priority and all the others are run when the UI thread is idle. In the third parameter for the schedule method, thisArg, I pass in some context for the job.

The schedule method returns a Job object, which lets me monitor a job's progress or cancel it. For each job, I assign the same OwnerToken object to its owner property. This lets me cancel all scheduled jobs attributed to that owner token. See **Figure 16**.

Now when I run the HTML UI Responsiveness diagnostics test, I should see some different results. **Figure 17** shows the results of the second test.

During the second test, the app dropped fewer frames over a shorter period of time. The experience in the app was better, too: the Hub control populated more quickly and there was almost no lag.

## Web Workers

The standard Web platform includes the Web Worker API that lets an app run background tasks off the UI thread. In short, Web Workers (or just Workers) allow for multi-threading in JavaScript applications. You pass simple messages (a string or simple JavaScript object) to the Worker thread and the Worker returns messages back to the main thread using the postMessage method.

Workers run in a different script context from the rest of the app, so they can't access the UI. You can't create new HTML elements using createElement or leverage features of third-party libraries that rely on the document object (for example, the jQuery function—$). However,

Figure 16 **Updated Scheduler.js Using Scheduler API**

```
(function () {
  "use strict";

  var dataRequest, jobOwnerToken;
  var scheduler = WinJS.Utilities.Scheduler;

  WinJS.UI.Pages.define("/pages/scheduler/scheduler.html", {
    ready: function (element, options) {
      performance.mark("navigated to scheduler");

      dataRequest = Data.featuredCollections.
        then(function (collections) {
          performance.mark("got collection");

          var hub = element.querySelector("#featuredHub");
          if (!hub) { return; }

          var hubSections = hub.winControl.sections,
          hubSection, collection, priority;

          jobOwnerToken = scheduler.createOwnerToken();

          for (var i = 0; i < hubSections.length; i++) {
            hubSection = hubSections.getItem(i);
            collection = collections.getItem(i);

            priority ==  (i < 2) ? scheduler.Priority.normal :
              scheduler.Priority.idle;

            scheduler.schedule(function () {
                populateSection(this.section, this.collection)
              },
                priority,
                { section: hubSection, collection: collection },
                "adding hub section").
              owner = jobOwnerToken;
          }
        });
      },

      unload: function () {
       dataRequest && dataRequest.cancel();
       jobOwnerToken && jobOwnerToken.cancelAll();
    }

    // Other PageControl members ...
  });

  function populateSection(section, collection) {
    performance.mark("creating a hub section");
    section.data.header = collection.data.title;

    var contentElement = section.data.contentElement;
    contentElement.innerHTML = "";

    var pictures = collection.data.pictures;
    for (var i = 0; i < 6; i++) {

      $(contentElement).append("<img src='" + pictures[i].pictureThumb + "' />");
      (i % 2) && $(contentElement).append("<br/>")
    }
  }
})();
```

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

## Work With Relational Data, Not Complex APIs or Services

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

Give RSSBus a try today and see what mean:

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

rssbus
INTEGRATION YOUR WAY

## Manipulating the DOM Affects UI Responsiveness

**Adding new elements** to the DOM on an HTML page can hurt performance, particularly if you're adding more than a handful of new elements. The page needs to recalculate the positions of the other items on the page, then reapply styles, and, finally, repaint the page. For example, a CSS instruction that sets the top, left, width, height, or display style for an element will cause the page to be recalculated. (I recommend using either the built-in animation features in WinJS or the animation transforms available in CSS3 for manipulating the position of HTML elements instead.)

Yet injecting and displaying dynamic content is a common app design. Your best option for performance, when possible, is to use the data binding provided by the platform. Data binding in WinJS is optimized for a quick and responsive UX.

Otherwise, you'll need to decide between injecting raw HTML as a string into another element with innerHTML, or adding individual elements one at a time using createElement and appendChild. Using innerHTML will most often provide better performance, but you might not be able to manipulate the HTML once it's been inserted.

In my examples, I chose the $.append method in jQuery. With append, I can pass along raw HTML as a string and get immediate programmatic access to the new DOM nodes. (It also provides pretty good performance.)

Workers can access the Windows Runtime APIs, which means they can write to app data, issue toasts and tile updates, or even save files. They're well-suited for background tasks that require no input from the user, are computationally expensive or require multiple calls to a Web service. If you want more information about the Web Worker API, see the Worker reference documentation at bit.ly/1fllmip.

The benefit of using a Worker thread is that UI responsiveness won't be affected by the background work. The UI remains responsive and practically no frames are dropped. Also, Workers can import other JavaScript libraries that don't rely on the DOM, including the fundamental library for WinJS (base.js). So, you can, for instance, create promises in a Worker thread.

On the other hand, Workers aren't a cure-all for performance issues. The cycles for the Worker threads are still being allocated from the total CPU cycles available on the machine, even if they aren't coming from the UI thread. You need to be judicious about using Workers.

For the next test case, I'll use a Worker thread to retrieve a collection of images from the Library of Congress and populate a ListView control with those pictures. First, I'll add a new script to store the Worker thread named LOC-worker.js to my project, as follows:

```
(function () {
  "use strict";
  self.addEventListener("message", function (message) {
    importScripts("//Microsoft.WinJS.2.0/js/base.js", "searchLoC.js");
    LOCPictures.getCollection(message.data).
      then(
        function (response) {
          postMessage(response);
        });
  });
})();
```

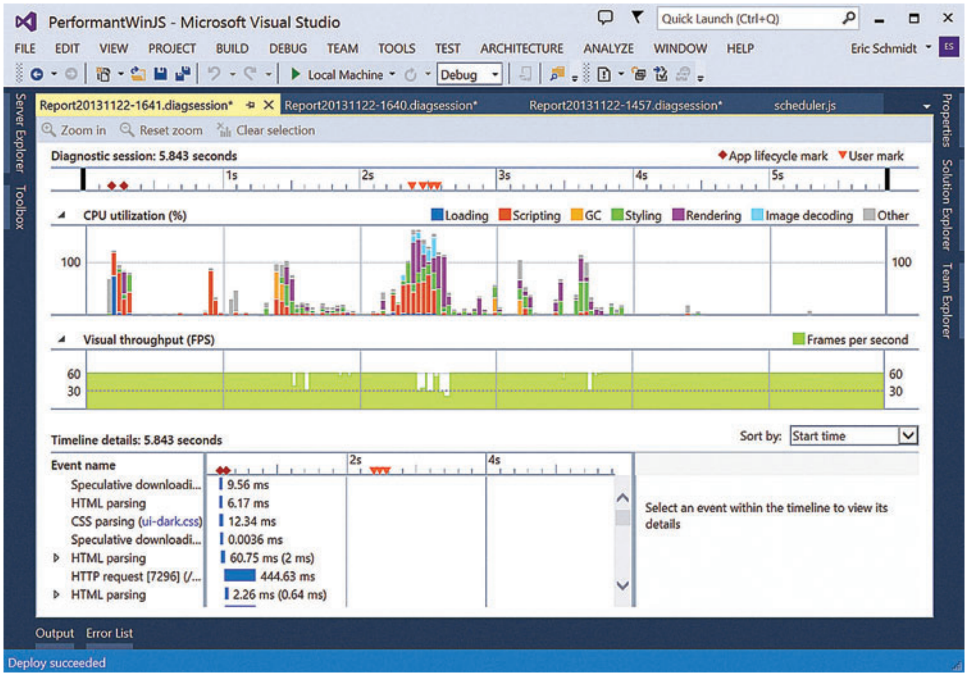
I use the importScripts function to bring base.js from the WinJS library and seachLOC.js scripts into the Worker's context, making them available for use.

Next, I add a new Page Control item named worker.html to my project (/pages/worker/worker.html). I add a little markup within the <section> tags in worker.html to contain the ListView control and define its layout. The control will be created dynamically when the Worker returns:

```
<div id="collection" class='searchList'>
  <progress class="win-ring"></progress>
</div>
<div id='searchResultsTemplate' data-win-control='WinJS.Binding.Template'>
  <div class='searchResultsItem'>
    <img src='#' data-win-bind='src: pictureThumb' />
    <div class='details'>
      <p data-win-bind='textContent: title'></p>
      <p data-win-bind='textContent: date'></p>
    </div>
  </div>
</div>
```

Finally, I add the code to worker.js that creates a new Worker thread and then populates the HTML based on the response. The code in worker.js is shown in **Figure 18**.

When you run the app and navigate to the page, you'll notice minimal lag between navigation and populating the ListView



Event name
Speculative downloadi... 9.56 ms
HTML parsing 6.17 ms
CSS parsing (ui-dark.css) 12.34 ms
Speculative downloadi... 0.0036 ms
HTML parsing 60.75 ms (2 ms)
HTTP request [7296] (/... 444.63 ms
HTML parsing 2.26 ms (0.64 ms)

Select an event within the timeline to view its details

Figure 17 **HTML UI Responsiveness After Using Scheduler.js**

Spreadsheets Made Easy.

WIN Windows Forms | Silverlight | WPF WPF

## Fastest Calculations

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

## Powerful Controls

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

## Comprehensive Charting

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

## Scalable Reporting

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

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

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

Figure 18 **Creating a Worker Thread and Then Populating the UI**

```
(function () {
  "use strict";
  WinJS.UI.Pages.define("/pages/worker/worker.html", {
    ready: function (element, options) {

      performance.mark("navigated to Worker");

      var getBaseballCards = new Worker('/js/LOC-worker.js'),
        baseballCards = new LOCPictures.Collection({
          title: "Baseball cards",
          thumbFeatured: null,
          code: "bbc"
      });

      getBaseballCards.onmessage = function (message) {
         createCollection(message.data);
         getBaseballCards.terminate();
      }
      getBaseballCards.postMessage(baseballCards);
    }

  // Other PageControl members ...
  });

  function createCollection(info) {
    var collection = new WinJS.Binding.List(info.pictures),
      collectionElement = $("# searchResultsTemplate")[0],
      collectionList = new WinJS.UI.ListView(collectionElement, {
        itemDataSource: collection.dataSource,
        itemTemplate: $('#collectionTemplate')[0],
        layout: {type: WinJS.UI.GridLayout}
      });

  }
})();
```

control with images. If you run this test case through the HTML UI responsiveness tool, you'll see output similar to what's shown in **Figure 19**.

Notice the app dropped very few frames after I navigated to the worker.html page (after the first user mark in the timeline). The UI remained incredibly responsive because the fetching of data was off-loaded to the Worker thread.

## When to Choose between the Scheduler and Worker APIs

Because both the Scheduler and the Worker APIs let you manage background tasks in your code, you might be wondering when to use one over the other. (Please note that the two code samples I've provided in this article aren't a fair "apples to apples" comparison of the two APIs.)

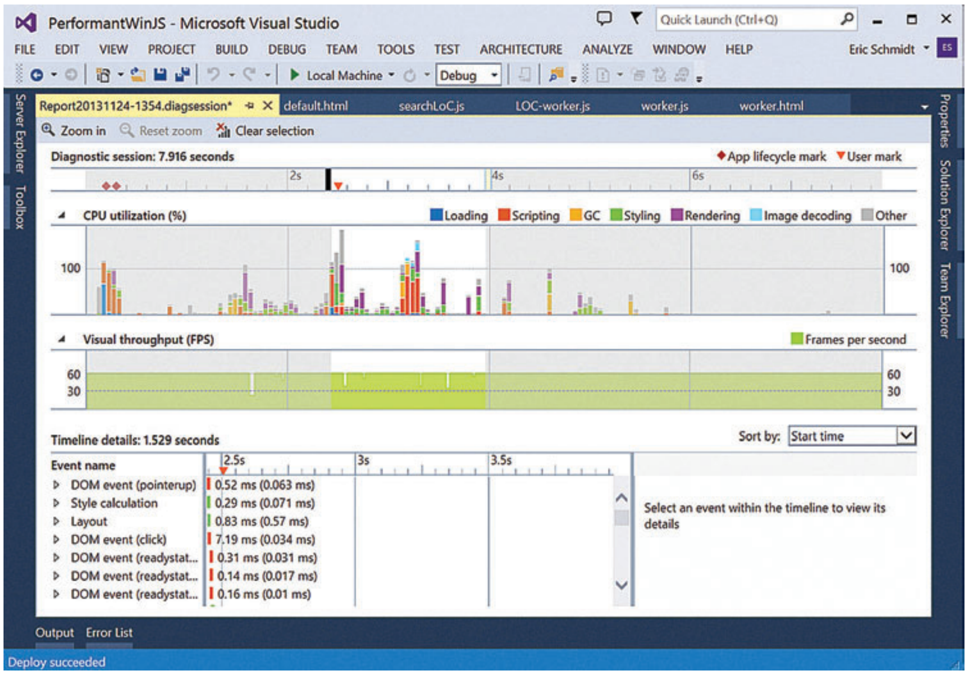
The Worker API, because it runs on a different thread, provides better performance than Scheduler in a head-to-head comparison. However, because the Scheduler uses timeslices of the UI thread, it has the context of the current page. You can use the Scheduler to update UI elements on a page or dynamically create new elements on the page.

If your background code needs to interact with the UI in any kind of meaningful way, you should use Scheduler. However, if your code doesn't rely on the context of the app and only passes simple data back and forth, consider using a Worker. The benefit of using a Worker thread is that UI responsiveness won't be affected by the background work.

The Scheduler and Web Worker APIs aren't your only choices for spawning multiple threads in a Windows Store app. You can also build a Windows Runtime Component in C++, C# or Visual Basic .NET that can create new threads. WinRT Components can expose APIs that the JavaScript code can call. For more information, see bit.ly/19DfFa0.

I doubt many developers set out to write buggy, glitchy or unresponsive apps (unless they're writing an article about buggy, glitchy and unresponsive apps). The trick is to find those bugs in your app code and fix them, preferably before the app goes in front of users. In this series of articles, I've demonstrated several tools for catching problems in your code and techniques for writing more efficient app code.

Of course, none of these techniques or tools is a silver bullet that will automatically fix a problem. They can help improve the experience, but they don't eliminate the need for good coding practices. The fundamentals of programming, in general, remain as true for Window Store apps built with JavaScript as for any other platform. ■



```
Event name
▷ DOM event (pointerup)    0.52 ms (0.063 ms)
▷ Style calculation        0.29 ms (0.071 ms)
▷ Layout                   0.83 ms (0.57 ms)
▷ DOM event (click)        7.19 ms (0.034 ms)
▷ DOM event (readystat...  0.31 ms (0.031 ms)
▷ DOM event (readystat...  0.14 ms (0.017 ms)
▷ DOM event (readystat...  0.16 ms (0.01 ms)
```

Figure 19 **HTML UI Responsiveness When Using a Worker Thread**

**Eric Schmidt** *is a content developer in the Microsoft Windows Developer Content team, writing about the Windows Library for JavaScript (WinJS). When previously in the Microsoft Office Division, he built code samples for the apps for Office platform. Otherwise, he spends time with his family, plays string bass, builds HTML5 video games, or blogs about plastic building toys (historybricks.com).*

**THANKS** to the following Microsoft technical experts for reviewing this article: *Kraig Brockschmidt, Greg Bulmash and Josh Williams*

# ALMFORUMSEATTLE

*expertise worth sharing*

Washington State Convention Center • April 1-3, 2014

Register by February 15th to save $300 follow us on

## keynote speakers

### Scott Ambler
ALM Thought Leader and Co-creator of DAD Framework

*Disciplined Agile Delivery: The Foundation for Scaling Agile*

### Steve Denning
Award-winning Author

*Transforming Management through Agile*

### Ken Schwaber
Industry Legend and Co-Creator of Scrum

*The State of Agile*

### Sam Guckenheimer
Product Owner, Microsoft Visual Studio

*Transforming software development in a world of devices and services*

## plenary speakers

### Mike Brittain
Director of Engineering, Etsy

*Principles and Practices of Continuous Deployment*

### James Whittaker
Distinguished Technical Evangelist, Microsoft

*Intent Commerce*

### Dave West
Chief Product Officer, Tasktop

*Lean ALM*

## sponsors

Scrum Alliance® Microsoft Scrum.org Tasktop NORTHWEST CADENCE avanade® Results Realized

diamond platinum

# Aspect-Oriented Programming with the RealProxy Class

Bruno Sonnino

A well-architected application has separate layers so different concerns don't interact more than needed. Imagine you're designing a loosely coupled and maintainable application, but in the middle of the development, you see some requirements that might not fit well in the architecture, such as:

- The application must have an authentication system, used before any query or update.
- The data must be validated before it's written to the database.
- The application must have auditing and logging for sensible operations.
- The application must maintain a debugging log to check if operations are OK.
- Some operations must have their performance measured to see if they're in the desired range.

Any of these requirements need a lot of work and, more than that, code duplication. You have to add the same code in many parts of the system, which goes against the "don't repeat yourself" (DRY) principle and makes maintenance more difficult. Any requirement change causes a massive change in the program. When I have to add something like that in my applications, I think, "Why can't the compiler add this repeated code in multiple places for me?" or, "I wish I had some option to 'Add logging to this method.'"

This article discusses:

- The basics of aspect-oriented programming (AOP)
- The Decorator design pattern
- Using the RealProxy class to create a dynamic proxy
- How to filter aspects
- The differences between RealProxy and PostSharp

Technologies discussed:

Microsoft .NET Framework, Aspect-Oriented Programming

The good news is that something like that does exist: aspect-oriented programming (AOP). It separates general code from aspects that cross the boundaries of an object or a layer. For example, the application log isn't tied to any application layer. It applies to the whole program and should be present everywhere. That's called a crosscutting concern.

AOP is, according to Wikipedia, "a programming paradigm that aims to increase modularity by allowing the separation of crosscutting concerns." It deals with functionality that occurs in multiple parts of the system and separates it from the core of the application, thus improving separation of concerns while avoiding duplication of code and coupling.

Figure 1 **The Repository<T> Class**

```
public class Repository<T> : IRepository<T>
{
  public void Add(T entity)
  {
    Console.WriteLine("Adding {0}", entity);
  }

  public void Delete(T entity)
  {
    Console.WriteLine("Deleting {0}", entity);
  }

  public void Update(T entity)
  {
    Console.WriteLine("Updating {0}", entity);
  }

  public IEnumerable<T> GetAll()
  {
    Console.WriteLine("Getting entities");
    return null;
  }

  public T GetById(int id)
  {
    Console.WriteLine("Getting entity {0}", id);
    return default(T);
  }
}
```

Figure 2 **The Main Program, with No Logging**

```
static void Main(string[] args)
{
  Console.WriteLine("***\r\n Begin program - no logging\r\n");
  IRepository<Customer> customerRepository =
    new Repository<Customer>();
  var customer = new Customer
  {
    Id = 1,
    Name = "Customer 1",
    Address = "Address 1"
  };
  customerRepository.Add(customer);
  customerRepository.Update(customer);
  customerRepository.Delete(customer);
  Console.WriteLine("\r\nEnd program - no logging\r\n***");
  Console.ReadLine();
}
```

In this article, I'll explain the basics of AOP and then detail how to make it easier by using a dynamic proxy via the Microsoft .NET Framework class RealProxy.

## Implementing AOP

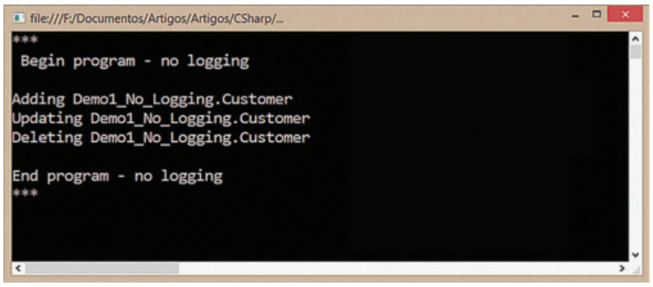
The biggest advantage of AOP is that you only have to worry about the aspect in one place, programming it once and applying it in all the places where needed. There are many uses for AOP, such as:

- Implementing logging in your application.
- Using authentication before an operation (such as allowing some operations only for authenticated users).
- Implementing validation or notification for property setters (calling the PropertyChanged event when a property has been changed for classes that implement the INotifyPropertyChanged interface).
- Changing the behavior of some methods.

As you can see, AOP has many uses, but you must wield it with care. It will keep some code out of your sight, but it's still there, running in every call where the aspect is present. It can have bugs and severely impact the performance of the application. A subtle bug in the aspect might cost you many debugging hours. If your aspect isn't used in many places, sometimes it's better to add it directly to the code.

AOP implementations use some common techniques:

- Adding source code using a pre-processor, such as the one in C++.
- Using a post-processor to add instructions on the compiled binary code.
- Using a special compiler that adds the code while compiling.
- Using a code interceptor at run time that intercepts execution and adds the desired code.



```
file:///F:/Documentos/Artigos/Artigos/CSharp/...
***
 Begin program - no logging

Adding Demo1_No_Logging.Customer
Updating Demo1_No_Logging.Customer
Deleting Demo1_No_Logging.Customer

End program - no logging
***
```

Figure 3 **Output of the Program with No Logging**

Figure 4 **The Logger Repository**

```
public class LoggerRepository<T> : IRepository<T>
{
  private readonly IRepository<T> _decorated;

  public LoggerRepository(IRepository<T> decorated)
  {
    _decorated = decorated;
  }

  private void Log(string msg, object arg = null)
  {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine(msg, arg);
    Console.ResetColor();
  }

  public void Add(T entity)
  {
    Log("In decorator - Before Adding {0}", entity);
    _decorated.Add(entity);
    Log("In decorator - After Adding {0}", entity);
  }

  public void Delete(T entity)
  {
    Log("In decorator - Before Deleting {0}", entity);
    _decorated.Delete(entity);
    Log("In decorator - After Deleting {0}", entity);
  }

  public void Update(T entity)
  {
    Log("In decorator - Before Updating {0}", entity);
    _decorated.Update(entity);
    Log("In decorator - After Updating {0}", entity);
  }

  public IEnumerable<T> GetAll()
  {
    Log("In decorator - Before Getting Entities");
    var result = _decorated.GetAll();
    Log("In decorator - After Getting Entities");
    return result;
  }

  public T GetById(int id)
  {
    Log("In decorator - Before Getting Entity {0}", id);
    var result = _decorated.GetById(id);
    Log("In decorator - After Getting Entity {0}", id);
    return result;
  }
}
```

In the .NET Framework, the most commonly used of these techniques are post-processing and code interception. The former is the technique used by PostSharp (postsharp.net) and the latter is used by dependency injection (DI) containers such as Castle DynamicProxy (bit.ly/JzE631) and Unity (unity.codeplex.com). These tools usually use a design pattern named Decorator or Proxy to perform the code interception.

## The Decorator Design Pattern

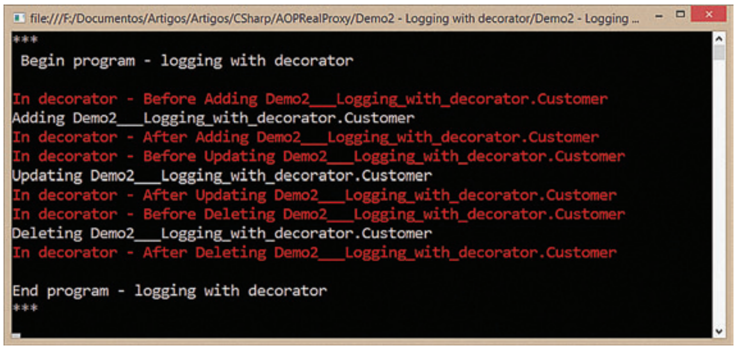
The Decorator design pattern solves a common problem: You have a class and want to add some functionality to it. You have several options for that:

- You could add the new functionality to the class directly. However, that gives the class another responsibility and hurts the "single responsibility" principle.
- You could create a new class that executes this functionality and call it from the old class. This brings a new problem: What if you also want to use the class without the new functionality?

Figure 5 **The Main Program Using The Logger Repository**

```
static void Main(string[] args)
{
  Console.WriteLine("***\r\n Begin program - logging with decorator\r\n");
  // IRepository<Customer> customerRepository =
  //   new Repository<Customer>();
  IRepository<Customer> customerRepository =
    new LoggerRepository<Customer>(new Repository<Customer>());

  var customer = new Customer
  {
    Id = 1,
    Name = "Customer 1",
    Address = "Address 1"
  };
  customerRepository.Add(customer);
  customerRepository.Update(customer);
  customerRepository.Delete(customer);
  Console.WriteLine("\r\nEnd program - logging with decorator\r\n***");
  Console.ReadLine();
}
```

- You could inherit a new class and add the new functionality, but that may result in many new classes. For example, let's say you have a repository class for create, read, update and delete (CRUD) database operations and you want to add auditing. Later, you want to add data validation to be sure the data is being updated correctly. After that, you might also want to authenticate the access to ensure that only authorized users can access the classes. These are big issues: You could have some classes that implement all three aspects, and some that implement only two of them or even only one. How many classes would you end up having?
- You can "decorate" the class with the aspect, creating a new class that uses the aspect and then calls the old one. That way, if you need one aspect, you decorate it once. For two aspects, you decorate it twice and so on. Let's say you order a toy (as we're all geeks, an Xbox or a smartphone is OK). It needs a package for display in the store and for protection. Then, you order it with gift wrap, the second decoration, to embellish the box with tapes, stripes, cards and gift paper. The store sends the toy with a third package, a box with Styrofoam balls for protection. You have three decorations, each one with a different functionality, and each one independent from one another. You can buy your toy with no gift packaging, pick it up at the store without the external box or even buy it with no box (with a special discount!). You can have your toy with any combination of the decorations, but they don't change its basic functionality.



```
file:///F:/Documentos/Artigos/Artigos/CSharp/AOPRealProxy/Demo2 - Logging with decorator/Demo2 - Logging ...
***
 Begin program - logging with decorator

In decorator - Before Adding Demo2___Logging_with_decorator.Customer
Adding Demo2___Logging_with_decorator.Customer
In decorator - After Adding Demo2___Logging_with_decorator.Customer
In decorator - Before Updating Demo2___Logging_with_decorator.Customer
Updating Demo2___Logging_with_decorator.Customer
In decorator - After Updating Demo2___Logging_with_decorator.Customer
In decorator - Before Deleting Demo2___Logging_with_decorator.Customer
Deleting Demo2___Logging_with_decorator.Customer
In decorator - After Deleting Demo2___Logging_with_decorator.Customer

End program - logging with decorator
***
```

Figure 6 **Execution of the Logging Program with a Decorator**

Figure 7 **The Dynamic Proxy Class**

```
class DynamicProxy<T> : RealProxy
{
  private readonly T _decorated;

  public DynamicProxy(T decorated)
    : base(typeof(T))
  {
    _decorated = decorated;
  }

  private void Log(string msg, object arg = null)
  {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine(msg, arg);
    Console.ResetColor();
  }

  public override IMessage Invoke(IMessage msg)
  {
    var methodCall = msg as IMethodCallMessage;
    var methodInfo = methodCall.MethodBase as MethodInfo;

    Log("In Dynamic Proxy - Before executing '{0}'",
      methodCall.MethodName);

    try
    {
      var result = methodInfo.Invoke(_decorated, methodCall.InArgs);
      Log("In Dynamic Proxy - After executing '{0}' ",
        methodCall.MethodName);

      return new ReturnMessage(result, null, 0,
        methodCall.LogicalCallContext, methodCall);
    }
    catch (Exception e)
    {
     Log(string.Format(
       "In Dynamic Proxy- Exception {0} executing '{1}'", e),
       methodCall.MethodName);
     return new ReturnMessage(e, methodCall);
    }
  }
}
```

Now that you know about the Decorator pattern, I'll show how to implement it in C#.

First, create an interface IRepository<T>:

```
public interface IRepository<T>
{
  void Add(T entity);
  void Delete(T entity);
  void Update(T entity);
  IEnumerable<T> GetAll();
  T GetById(int id);
}
```

Implement it with the Repository<T> class, shown in **Figure 1**. Use the Repository<T> class to add, update, delete and retrieve the elements of the Customer class:

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

The program could look something like **Figure 2**.

When you run this code, you'll see something like **Figure 3**.

Imagine your boss asks you to add logging to this class. You can create a new class that will decorate IRepository<T>. It receives the class to build and implements the same interface, as shown in **Figure 4**.

This new class wraps the methods for the decorated class and adds the logging feature. You must change the code a little to call the logging class, as shown in **Figure 5**.

Figure 8 **The Main Program with a Dynamic Proxy**

```
static void Main(string[] args)
{
  Console.WriteLine("***\r\n Begin program - logging with dynamic proxy\r\n");
  // IRepository<Customer> customerRepository =
  //   new Repository<Customer>();
  // IRepository<Customer> customerRepository =
  //   new LoggerRepository<Customer>(new Repository<Customer>());
  IRepository<Customer> customerRepository =
    RepositoryFactory.Create<Customer>();
  var customer = new Customer
  {
    Id = 1,
    Name = "Customer 1",
    Address = "Address 1"
   ;
  customerRepository.Add(customer);
  customerRepository.Update(customer);
  customerRepository.Delete(customer);
  Console.WriteLine("\r\nEnd program - logging with dynamic proxy\r\n***");
  Console.ReadLine();
}
```

You simply create the new class, passing an instance of the old class as a parameter for its constructor. When you execute the program, you can see it has the logging, as shown in **Figure 6**.

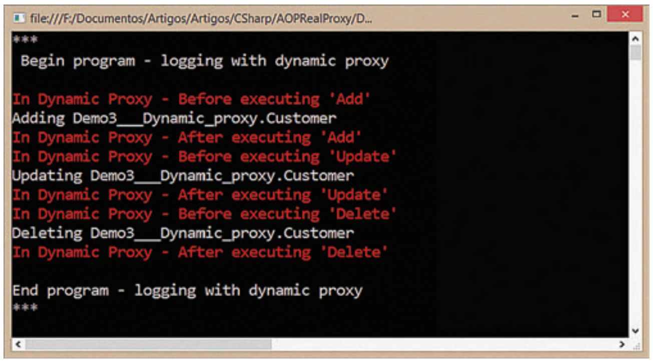
You might be thinking: "OK, the idea is good, but it's a lot of work: I have to implement all the classes and add the aspect to all the methods. That will be difficult to maintain. Is there another way to do it?" With the .NET Framework, you can use reflection to get all methods and execute them. The base class library (BCL) even has the RealProxy class (bit.ly/18MfxWo) that does the implementation for you.

## Creating a Dynamic Proxy with RealProxy

The RealProxy class gives you basic functionality for proxies. It's an abstract class that must be inherited by overriding its Invoke method and adding new functionality. This class is in the namespace System.Runtime.Remoting.Proxies. To create a dynamic proxy, you use code similar to **Figure 7**.

In the constructor of the class, you must call the constructor of the base class, passing the type of the class to be decorated. Then you must override the Invoke method that receives an IMessage parameter. It contains a dictionary with all the parameters passed for the method. The IMessage parameter is typecast to an IMethodCallMessage, so you can extract the parameter MethodBase (which has the MethodInfo type).

The next steps are to add the aspect you want before calling the method, call the original method with methodInfo.Invoke and then add the aspect after the call.



```
file:///F:/Documentos/Artigos/Artigos/CSharp/AOPRealProxy/D...
***
 Begin program - logging with dynamic proxy

In Dynamic Proxy - Before executing 'Add'
Adding Demo3___Dynamic_proxy.Customer
In Dynamic Proxy - After executing 'Add'
In Dynamic Proxy - Before executing 'Update'
Updating Demo3___Dynamic_proxy.Customer
In Dynamic Proxy - After executing 'Update'
In Dynamic Proxy - Before executing 'Delete'
Deleting Demo3___Dynamic_proxy.Customer
In Dynamic Proxy - After executing 'Delete'

End program - logging with dynamic proxy
***
```

Figure 9 **Program Execution with Dynamic Proxy**

Figure 10 **An Authentication Proxy**

```
class AuthenticationProxy<T> : RealProxy
{
  private readonly T _decorated;

  public AuthenticationProxy(T decorated)
    : base(typeof(T))
  {
    _decorated = decorated;
  }

  private void Log(string msg, object arg = null)
  {
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine(msg, arg);
    Console.ResetColor();
  }

  public override IMessage Invoke(IMessage msg)
  {
    var methodCall = msg as IMethodCallMessage;
    var methodInfo = methodCall.MethodBase as MethodInfo;

    if (Thread.CurrentPrincipal.IsInRole("ADMIN"))
    {
      try
      {
        Log("User authenticated - You can execute '{0}' ",
          methodCall.MethodName);
        var result = methodInfo.Invoke(_decorated, methodCall.InArgs);

        return new ReturnMessage(result, null, 0,
          methodCall.LogicalCallContext, methodCall);
      }
      catch (Exception e)
      {
        Log(string.Format(
          "User authenticated - Exception {0} executing '{1}'", e),
          methodCall.MethodName);
        return new ReturnMessage(e, methodCall);
      }
    }
    Log("User not authenticated - You can't execute '{0}' ",
      methodCall.MethodName);
    return new ReturnMessage(null, null, 0,
      methodCall.LogicalCallContext, methodCall);
  }
}
```

You can't call your proxy directly, because DynamicProxy<T> isn't an IRepository<Customer>. That means you can't call it like this:

```
IRepository<Customer> customerRepository =
  new DynamicProxy<IRepository<Customer>>(
  new Repository<Customer>());
```

To use the decorated repository, you must use the GetTransparentProxy method, which will return an instance of IRepository<Customer>. Every method of this instance that's called will go through the proxy's Invoke method. To ease this process, you can create a Factory class to create the proxy and return the instance for the repository:

```
public class RepositoryFactory
{
  public static IRepository<T> Create<T>()
  {
    var repository = new Repository<T>();
    var dynamicProxy = new DynamicProxy<IRepository<T>>(repository);
    return dynamicProxy.GetTransparentProxy() as IRepository<T>;
  }
}
```

That way, the main program will be similar to **Figure 8**.

When you execute this program, you get a similar result as before, as shown in **Figure 9**.

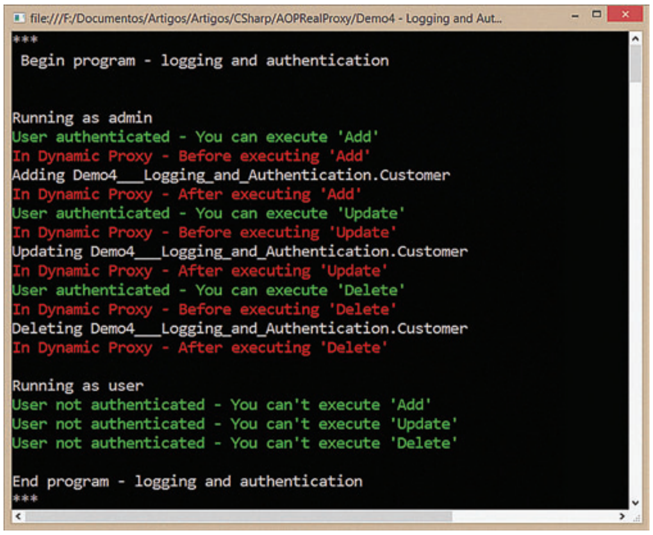
As you can see, you've created a dynamic proxy that allows adding aspects to the code, with no need to repeat it. If you wanted to

Figure 11 **The Repository Factory Decorated by Two Proxies**

```
public class RepositoryFactory
{
  public static IRepository<T> Create<T>()
  {
    var repository = new Repository<T>();
    var decoratedRepository =
      (IRepository<T>)new DynamicProxy<IRepository<T>>(
      repository).GetTransparentProxy();
    // Create a dynamic proxy for the class already decorated
    decoratedRepository =
      (IRepository<T>)new AuthenticationProxy<IRepository<T>>(
      decoratedRepository).GetTransparentProxy();
    return decoratedRepository;
  }
}
```

Figure 12 **The Main Program Calling the Repository with Two Users**

```
static void Main(string[] args)
{
  Console.WriteLine(
    "***\r\n Begin program - logging and authentication\r\n");
  Console.WriteLine("\r\nRunning as admin");
  Thread.CurrentPrincipal =
    new GenericPrincipal(new GenericIdentity("Administrator"),
    new[] { "ADMIN" });
  IRepository<Customer> customerRepository =
    RepositoryFactory.Create<Customer>();
  var customer = new Customer
  {
    Id = 1,
    Name = "Customer 1",
    Address = "Address 1"
  };
  customerRepository.Add(customer);
  customerRepository.Update(customer);
  customerRepository.Delete(customer);
  Console.WriteLine("\r\nRunning as user");
  Thread.CurrentPrincipal =
    new GenericPrincipal(new GenericIdentity("NormalUser"),
    new string[] { });
  customerRepository.Add(customer);
  customerRepository.Update(customer);
  customerRepository.Delete(customer);
  Console.WriteLine(
    "\r\nEnd program - logging and authentication\r\n***");
  Console.ReadLine();

}
```



```
file:///F:/Documentos/Artigos/Artigos/CSharp/AOPRealProxy/Demo4 - Logging and Aut...
***
 Begin program - logging and authentication


Running as admin
User authenticated - You can execute 'Add'
In Dynamic Proxy - Before executing 'Add'
Adding Demo4___Logging_and_Authentication.Customer
In Dynamic Proxy - After executing 'Add'
User authenticated - You can execute 'Update'
In Dynamic Proxy - Before executing 'Update'
Updating Demo4___Logging_and_Authentication.Customer
In Dynamic Proxy - After executing 'Update'
User authenticated - You can execute 'Delete'
In Dynamic Proxy - Before executing 'Delete'
Deleting Demo4___Logging_and_Authentication.Customer
In Dynamic Proxy - After executing 'Delete'

Running as user
User not authenticated - You can't execute 'Add'
User not authenticated - You can't execute 'Update'
User not authenticated - You can't execute 'Delete'

End program - logging and authentication
***
```

Figure 13 **Output of the Program Using Two Proxies**

Figure 14 **Filtering Methods for the Aspect**

```
public override IMessage Invoke(IMessage msg)
{
  var methodCall = msg as IMethodCallMessage;
  var methodInfo = methodCall.MethodBase as MethodInfo;

  if (!methodInfo.Name.StartsWith("Get"))
    Log("In Dynamic Proxy - Before executing '{0}'",
      methodCall.MethodName);

  try
  {
    var result = methodInfo.Invoke(_decorated, methodCall.InArgs);
    if (!methodInfo.Name.StartsWith("Get"))
      Log("In Dynamic Proxy - After executing '{0}' ",
        methodCall.MethodName);

      return new ReturnMessage(result, null, 0,
       methodCall.LogicalCallContext, methodCall);
  }
  catch (Exception e)
  {
    if (!methodInfo.Name.StartsWith("Get"))
      Log(string.Format(
        "In Dynamic Proxy- Exception {0} executing '{1}'", e),
        methodCall.MethodName);
      return new ReturnMessage(e, methodCall);
  }
}
```

add a new aspect, you'd only need to create a new class, inherit from RealProxy and use it to decorate the first proxy.

If your boss comes back to you and asks you to add authorization to the code, so only administrators can access the repository, you could create a new proxy as shown in **Figure 10**.

The repository factory must be changed to call both proxies, as shown in **Figure 11**.

When you change the main program to **Figure 12** and run it, you'll get the output shown in **Figure 13**.

The program executes the repository methods twice. The first time, it runs as an admin user and the methods are called. The second time, it runs as a normal user and the methods are skipped.

That's much easier, isn't it? Note that the factory returns an instance of IRepository<T>, so the program doesn't know if it's using the decorated version. This respects the Liskov Substitution Principle, which says that if S is a subtype of T, then objects of type T may be replaced with objects of type S. In this case, by using an IRepository<Customer> interface, you could use any class that implements this interface with no change in the program.

## Filtering Functions

Until now, there was no filtering in the functions; the aspect is applied to every class method that's called. Often this isn't the desired behavior. For example, you might not want to log the retrieval methods (GetAll and GetById). One way to accomplish this is to filter the aspect by name, as in **Figure 14**.

The program checks if the method starts with "Get." If it does, the program doesn't apply the aspect. This works, but the filtering code is repeated three times. Besides that, the filter is inside the proxy, which will make you change the class every time you want to change the proxy. You can improve this by creating an IsValidMethod predicate:

```
private static bool IsValidMethod(MethodInfo methodInfo)
{
  return !methodInfo.Name.StartsWith("Get");
}
```

Figure 15 **A Filtering Proxy**

```
class DynamicProxy<T> : RealProxy
{
  private readonly T _decorated;
  private Predicate<MethodInfo> _filter;

  public DynamicProxy(T decorated)
    : base(typeof(T))
  {
    _decorated = decorated;
    _filter = m => true;
  }

  public Predicate<MethodInfo> Filter
  {
    get { return _filter; }
    set
    {
      if (value == null)
        _filter = m => true;
      else
        _filter = value;
    }
  }

  private void Log(string msg, object arg = null)
  {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine(msg, arg);
    Console.ResetColor();
  }

  public override IMessage Invoke(IMessage msg)
  {
    var methodCall = msg as IMethodCallMessage;
    var methodInfo = methodCall.MethodBase as MethodInfo;

    if (_filter(methodInfo))
      Log("In Dynamic Proxy - Before executing '{0}'",
        methodCall.MethodName);

    try
    {
      var result = methodInfo.Invoke(_decorated, methodCall.InArgs);
      if (_filter(methodInfo))
        Log("In Dynamic Proxy - After executing '{0}' ",
          methodCall.MethodName);

        return new ReturnMessage(result, null, 0,
          methodCall.LogicalCallContext, methodCall);
    }
    catch (Exception e)
    {
      if (_filter(methodInfo))
        Log(string.Format(
          "In Dynamic Proxy- Exception {0} executing '{1}'", e),
          methodCall.MethodName);
      return new ReturnMessage(e, methodCall);
    }
  }
}
```

Now you need to make the change in only one place, but you still need to change the class every time you want to change the filter. One solution to this would be to expose the filter as a class property, so you can assign the responsibility of creating a filter to the caller. You can create a Filter property of type Predicate<MethodInfo> and use it to filter the data, as shown in **Figure 15**.

The Filter property is initialized with Filter = m => true. That means there's no filter active. When assigning the Filter property, the program verifies if the value is null and, if so, it resets the filter. In the Invoke method execution, the program checks the filter result and, if true, it applies the aspect. Now the proxy creation in the factory class looks like this:

```
public class RepositoryFactory
{
  public static IRepository<T> Create<T>()
  {
    var repository = new Repository<T>();
    var dynamicProxy = new DynamicProxy<IRepository<T>>(repository)
    {
      Filter = m => !m.Name.StartsWith("Get")
      };
    return dynamicProxy.GetTransparentProxy() as IRepository<T>;
  }
}
}
```

The responsibility of creating the filter has been transferred to the factory. When you run the program, you should get something like **Figure 16**.

Notice in **Figure 16** that the last two methods, GetAll and GetById (represented by "Getting entities" and "Getting entity 1") don't have logging around them. You can enhance the class even further by exposing the aspects as events. That way, you don't have to change the class every time you want to change the aspect. This is shown in **Figure 17**.

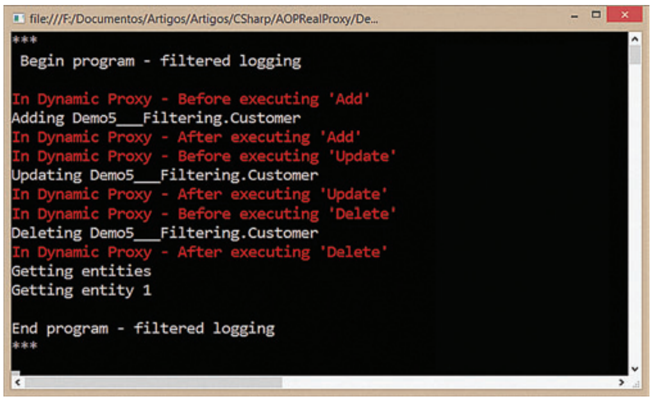
In **Figure 17**, three events, BeforeExecute, AfterExecute and ErrorExecuting, are called by the methods OnBeforeExecute, OnAfterExecute and OnErrorExecuting. These methods verify whether the event handler is defined and, if it is, they check if the called method passes the filter. If so, they call the event handler that applies the aspect. The factory class now becomes something like **Figure 18**.

Now you have a flexible proxy class, and you can choose the aspects to apply before executing, after executing or when there's an error, and only for selected methods. With that, you can apply many aspects in your code with no repetition, in a simple way.

## Not a Replacement

With AOP you can add code to all layers of your application in a centralized way, with no need to repeat code. I showed how to create a generic dynamic proxy based on the Decorator design pattern that applies aspects to your classes using events and a predicate to filter the functions you want.

As you can see, the RealProxy class is a flexible class and gives you full control of the code, with no external dependencies. How-


file:///F:/Documentos/Artigos/Artigos/CSharp/AOPRealProxy/De...

```
***
 Begin program - filtered logging

In Dynamic Proxy - Before executing 'Add'
Adding Demo5___Filtering.Customer
In Dynamic Proxy - After executing 'Add'
In Dynamic Proxy - Before executing 'Update'
Updating Demo5___Filtering.Customer
In Dynamic Proxy - After executing 'Update'
In Dynamic Proxy - Before executing 'Delete'
Deleting Demo5___Filtering.Customer
In Dynamic Proxy - After executing 'Delete'
Getting entities
Getting entity 1

End program - filtered logging
***
```

Figure 16 **Output with a Filtered Proxy**

Figure 17 **A Flexible Proxy**

```
class DynamicProxy<T> : RealProxy
{
  private readonly T _decorated;
  private Predicate<MethodInfo> _filter;

  public event EventHandler<IMethodCallMessage> BeforeExecute;
  public event EventHandler<IMethodCallMessage> AfterExecute;
  public event EventHandler<IMethodCallMessage> ErrorExecuting;
  public DynamicProxy(T decorated)
    : base(typeof(T))
  {
    _decorated = decorated;
    Filter = m => true;
  }

  public Predicate<MethodInfo> Filter
  {
    get { return _filter; }
    set
    {
      if (value == null)
        _filter = m => true;
      else
        _filter = value;
    }
  }

  private void OnBeforeExecute(IMethodCallMessage methodCall)
  {
    if (BeforeExecute != null)
    {
      var methodInfo = methodCall.MethodBase as MethodInfo;
      if (_filter(methodInfo))
        BeforeExecute(this, methodCall);
    }
  }

  private void OnAfterExecute(IMethodCallMessage methodCall)
  {
    if (AfterExecute != null)
    {
      var methodInfo = methodCall.MethodBase as MethodInfo;
      if (_filter(methodInfo))
        AfterExecute(this, methodCall);
    }
  }

  private void OnErrorExecuting(IMethodCallMessage methodCall)
  {
    if (ErrorExecuting != null)
    {
      var methodInfo = methodCall.MethodBase as MethodInfo;
      if (_filter(methodInfo))
        ErrorExecuting(this, methodCall);
    }
  }

  public override IMessage Invoke(IMessage msg)
  {
    var methodCall = msg as IMethodCallMessage;
    var methodInfo = methodCall.MethodBase as MethodInfo;

    OnBeforeExecute(methodCall);
    try
    {
      var result = methodInfo.Invoke(_decorated, methodCall.InArgs);
      OnAfterExecute(methodCall);

      return new ReturnMessage(
        result, null, 0, methodCall.LogicalCallContext, methodCall);
    }
    catch (Exception e)
    {
      OnErrorExecuting(methodCall);
      return new ReturnMessage(e, methodCall);
    }
  }
}
```

ever, note that RealProxy isn't a replacement for other AOP tools, such as PostSharp. PostSharp uses a completely different method. It will add intermediate language (IL) code in a post-compilation step and won't use reflection, so it should have better performance than RealProxy. You'll also have to do more work to implement an aspect with RealProxy than with PostSharp. With PostSharp, you need only create the aspect class and add an attribute to the class (or the method) where you want the aspect added, and that's all.

Figure 18 **A Repository Factory that Sets the Aspect Events and Filter**

```
public class RepositoryFactory
{
  private static void Log(string msg, object arg = null)
  {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine(msg, arg);
    Console.ResetColor();
  }
  public static IRepository<T> Create<T>()
  {
    var repository = new Repository<T>();
    var dynamicProxy = new DynamicProxy<IRepository<T>>(repository);
    dynamicProxy.BeforeExecute += (s, e) => Log(
      "Before executing '{0}'", e.MethodName);
    dynamicProxy.AfterExecute += (s, e) => Log(
      "After executing '{0}'", e.MethodName);
    dynamicProxy.ErrorExecuting += (s, e) => Log(
      "Error executing '{0}'", e.MethodName);
    dynamicProxy.Filter = m => !m.Name.StartsWith("Get");
    return dynamicProxy.GetTransparentProxy() as IRepository<T>;
  }
}
```

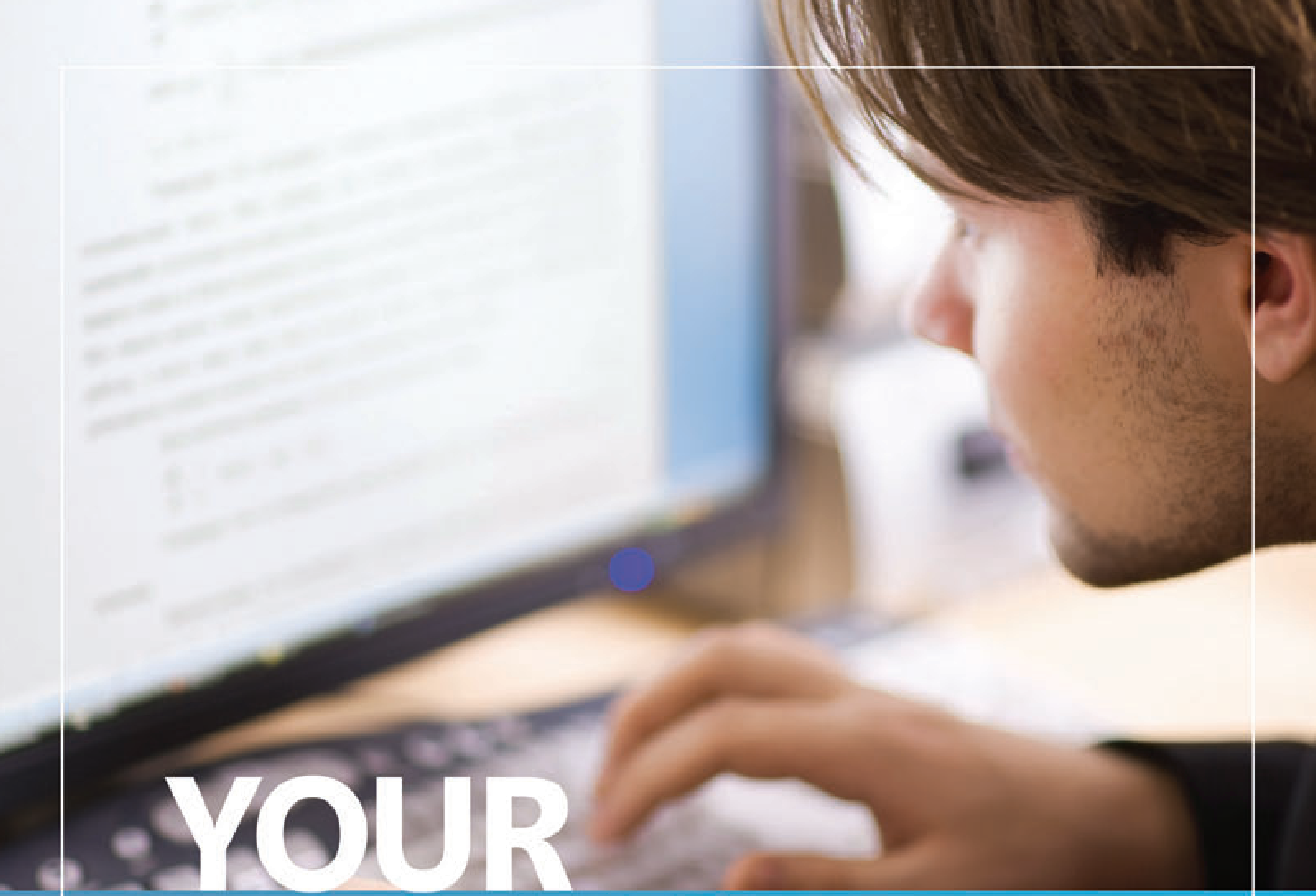
On the other hand, with RealProxy, you'll have full control of your source code, with no external dependencies, and you can extend and customize it as much as you want. For example, if you want to apply an aspect only on methods that have the Log attribute, you could do something like this:

```
public override IMessage Invoke(IMessage msg)
{
  var methodCall = msg as IMethodCallMessage;
  var methodInfo = methodCall.MethodBase as MethodInfo;
  if (!methodInfo.CustomAttributes
    .Any(a => a.AttributeType == typeof (LogAttribute)))
  {
    var result = methodInfo.Invoke(_decorated, methodCall.InArgs);
    return new ReturnMessage(result, null, 0,
      methodCall.LogicalCallContext, methodCall);
  }
    ...
```

Besides that, the technique used by RealProxy (intercept code and allow the program to replace it) is powerful. For example, if you want to create a mocking framework, for creating generic mocks and stubs for testing, you can use the RealProxy class to intercept all calls and replace them with your own behavior, but that's a subject for another article! ■

**BRUNO SONNINO** *is a Microsoft Most Valuable Professional (MVP) located in Brazil. He's a developer, consultant, and author, having written five Delphi books, published in Portuguese by Pearson Education Brazil, and many articles for Brazilian and U.S. magazines and Web sites.*

**THANKS** to the following Microsoft Research technical experts for reviewing this article: *James McCaffrey, Carlos Suarez and Johan Verwey*



Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

PLATINUM SPONSOR esri

SUPPORTED BY Microsoft · Visual Studio · msdn magazine · Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA INC

## LAS VEGAS 2014

March 10 - 14 | Planet Hollywood Resort & Casino

LIVE! 360 DEV

Visual Studio LIVE!

YOUR GUIDE TO THE .NET DEVELOPMENT UNIVERSE

**Visual Studio Live!** is your guide to the .NET Development universe, featuring code-filled days, networking nights and independent education. Whether you are a .NET developer, software architect or a designer, Visual Studio Live!'s multi-track events include focused, cutting-edge education on the .NET platform that you'll be ready to implement as soon as you get back to the office.

```
nstore

var nstore = require('nstore');
var messages = nstore.new('messages.db', ...);

messages.save(msg.id, msg, function (err) ...);

messages.all(function (err, results) {
  ...
});

messages.find({ user: 'name' }, ...);
```

## COMPREHENSIVE TRAINING FOR THE DEVELOPER WORLD.

**Visual Studio Live!** Las Vegas is part of Live! 360 DEV, which means you'll have access to four (4) other co-located events **at no additional cost:**

SQL Server LIVE!
SQL SERVER FOR MODERN DEVELOPERS

SharePoint LIVE!
TRAINING FOR COLLABORATION

Modern Apps LIVE!
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

AND INTRODUCING Web Dev Live!

Web Dev LIVE!
HTML5, JAVASCRIPT & ASP.NET TRAINING

**Five (5) events** means over a hundred sessions to choose from – mix and match sessions to create your own, custom event line-up - it's like no other dev conference available today!

**Register by February 12 and Save $300!**

Use promo code VSLFEB4

vslive.com/lasvegas

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

TURN THE PAGE FOR MORE EVENT DETAILS →

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

LIVE! 360 DEV

| VISUAL STUDIO LIVE! TRACKS | | | | | | WEB DEV LIVE! TRACKS | |
|---|---|---|---|---|---|---|---|
| Visual Studio / .NET Framework | Windows 8.1 / WinRT | WPF / Silverlight | Cloud Computing | Windows Phone | Cross-Platform Mobile Development | ASP.NET | HTML5 |

| | | VISUAL STUDIO LIVE! | | | WEB DEV LIVE! |
|---|---|---|---|---|---|
| **START TIME** | **END TIME** | Pre-Conference Workshops: Monday, March 10, 2014 (Separate entry fee required) | | | |
| 7:30 AM | 9:00 AM | Pre-Conference Workshop Registration - Coffee and Morning Pastries | | | |
| 9:00 AM | 6:00 PM | **LWM01** Workshop: Building Distributed Modern Apps in C# and XAML - *Rockford Lhotka* | **LWM02** Workshop: Modern UX Design - *Billy Hollis* | | **LWM03** Workshop: Data-Centric Single Page Applications with Knockout, Durandal, Breeze, and Web API - *Brian Noyes* |
| 7:00 PM | 9:00 PM | Dine-A-Round Dinner | | | |
| **START TIME** | **END TIME** | Day 1: Tuesday, March 11, 2014 | | | |
| 7:00 AM | 8:00 AM | Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 9:00 AM | Keynote: Modern Development with Visual Studio. Nicole Herskowitz, Sr. Director, Visual Studio, Microsoft | | | |
| 9:15 AM | 10:30 AM | **LT01** Visual Studio 2013 New IDE Features Part I - *Deborah Kurata* | **LT02** What's New for HTML/WinJS Windows Store Apps - *Ben Dewey* | **LT03** Introduction to Windows Azure - *Vishwas Lele* | **LT04** HTML5 for Better Web Sites - *Robert Boedigheimer* |
| 10:45 AM | 12:00 PM | **LT08** Visual Studio 2013 New IDE Features Part II - *Deborah Kurata* | **LT09** What's New for XAML Windows Store Apps - *Ben Dewey* | **LT10** Windows Azure Cloud Services - *Vishwas Lele* | **LT11** 12 Things Every Developer Really Needs to Know About JavaScript - *Rachel Appel* |
| 12:00 PM | 1:15 PM | Lunch • Visit the EXPO | | | |
| 1:15 PM | 2:15 PM | **LT15** - Chalk Talk: To Be Announced | | | **LT16** Chalk Talk: Introduction to Speech Recognition with C# - *James McCaffrey* |
| 2:30 PM | 3:45 PM | **LT19** What's New in Team Foundation Server 2013 - *Benjamin Day* | **LT20** Controlling Hardware Using Windows 8.1 - *Brian Peek* | **LT21** WCF & Web API: Can We All Just Get Along?!? - *Miguel Castro* | **LT22** Great User Experiences with CSS 3 - *Robert Boedigheimer* |
| 3:45 PM | 4:15 PM | Networking Break • Visit the EXPO | | | |
| 4:15 PM | 5:30 PM | **LT26** Visual Studio Online: Cloud-y Visual Studio & TFS for Your Teams - *Benjamin Day* | **LT27** Performance and Diagnostics Hub in Visual Studio 2013 - *Brian Peek* | **LT28** Windows Azure Web Sites - *Vishwas Lele* | **LT29** Writing Next Generation Javascript with TypeScript - *Rachel Appel* |
| 5:30 PM | 7:00 PM | Welcome Reception | | | |
| **START TIME** | **END TIME** | Day 2: Wednesday, March 12, 2014 | | | |
| 7:00 AM | 8:00 AM | Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 9:00 AM | Keynote: To Be Announced | | | |
| 9:15 AM | 10:30 AM | **LW01** Getting Started with Windows Phone Development - *Nick Landry* | **LW02** Test Your XAML-based Windows Store Apps with VS2013 - *Benjamin Day* | **LW03** Overview of IaaS in Windows Azure with Virtual Machines - *Eric D. Boyd* | **LW04** Using jQuery to Replace the Ajax Control Toolkit - *Robert Boedigheimer* |
| 10:45 AM | 12:00 PM | **LW08** Developing Windows Phone Apps with Azure - *Rockford Lhotka* | **LW09** Accelerate Your App Design with Blend for Visual Studio 2013 - *Brian Noyes* | **LW10** Building Real-time, Multi-user Interactive Web and Mobile Applications Using SignalR - *Marcel de Vries* | **LW11** Building Web Applications Using Kendo UI and the MVVM Pattern - *Ben Hoelting* |
| 12:00 PM | 1:15 PM | Birds-of-a-Feather Lunch • Visit the EXPO | | | |
| 1:15 PM | 2:30 PM | **LW15** Cross Platform Native Mobile App Development for iOS, Android and Windows Using the Power of C# - *Marcel de Vries* | **LW16** Holy Updates! (or What's New in Windows 8.1) - *Philip Japikse* | **LW17** Persistence In The Cloud: How to Use Azure Storage - *David Giard* | **LW18** Angular for .NET Developers - *Jesse Liberty* |
| 2:45 PM | 4:00 PM | **LW22** Mobile App Development for the Web Developer Using PhoneGap - *Eric D. Boyd* | **LW23** WinRT Business Application Architecture - *Rockford Lhotka* | **LW24** Building Services with ASP.NET MVC Web API Deep Dive - *Marcel de Vries* | **LW25** Build Your First Angular Web Application - *Matthew DiFranco* |
| 4:00 PM | 4:30 PM | Sponsored Break • Visit the EXPO • Expo Raffle @ 4:15PM | | | |
| 4:30 PM | 5:45 PM | **LW29** iOS Development Survival Guide for the .NET Guy - *Nick Landry* | **LW30** Overview of the Bing Platform - *Brian Peek* | **LW31** Debugging and Monitoring Windows Azure Cloud Services - *Eric D. Boyd* | **LW32** Connecting the Dots: Using HTML5, jQuery, and Web API Together - *David Giard* |
| 7:00 PM | 9:00 PM | Live! 360 Evening Event | | | |
| **START TIME** | **END TIME** | Day 3: Thursday, March 13, 2014 | | | |
| 7:00 AM | 8:00 AM | Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 9:15 AM | **LTH01** What's New in .NET 4.5.1 - *Jason Bock* | **LTH02** Interaction Design Principles and Patterns - *Billy Hollis* | **LTH03** To Be Announced | **LTH04** Knocking it Out of the Park with Knockout.JS - *Miguel Castro* |
| 9:30 AM | 10:45 AM | **LTH08** Building Shared Mobile Apps for Windows Store and Windows Phone - *Nick Landry* | **LTH09** Applying UX Design in XAML - *Billy Hollis* | **LTH10** Learning Entity Framework 6 - *Leonard Lobel* | **LTH11** Structuring Your Single Page Application with Durandal - *Brian Noyes* |
| 11:00 AM | 12:15 PM | **LTH15** Visual Studio Online—It's More Than You Know - *Brian Randell* | **LTH16** Creating Games for Windows 8 with Unity - *Brian Lagunas* | **LTH17** To Be Announced | **LTH18** Create an API For Your Application With Service Stack - *Jesse Liberty* |
| 12:15 PM | 1:30 PM | Lunch | | | |
| 1:30 PM | 2:45 PM | **LTH22** App Insights - App Performance Monitoring - *Brian Randell* | **LTH23** Building Composite XAML Applications with Prism - *Brian Lagunas* | **LTH24** Asynchronous Debugging in .NET - *Jason Bock* | **LTH25** Busy Developer's Guide to Node.js - *Ted Neward* |
| 3:00 PM | 4:15 PM | **LTH29** App Insights - Global System Monitoring - *Brian Randell* | **LTH30** Implementing MVVM (Model-View-View Model) for WPF - *Philip Japikse* | **LTH31** Browser as Code Editor: A Tour of Monaco - *Jason Bock* | **LTH32** Busy Developer's Guide to Everything Not-JavaScript - *Ted Neward* |
| **START TIME** | **END TIME** | POST-CONFERENCE WORKSHOPS: FRIDAY, MARCH 14, 2014 (SEPARATE ENTRY FEE REQUIRED) | | | |
| 7:30 AM | 8:00 AM | Post-Conference Workshop Registration - Coffee and Morning Pastries | | | |
| 8:00 AM | 5:00 PM | **LWF01** Workshop: Deep Dive into Visual Studio 2013, TFS, and Visual Studio Online - *Brian Randell* | **LWF02** Workshop: Build and Submit a Game to the Windows App Store - *David Giard* | | **LWF03** Workshop: Busy Developer's Guide to AngularJS - *Ted Neward* |

Speakers and sessions subject to change

vslive.com/lasvegas

# LAS VEGAS 2014

**March 10 – 14 | Planet Hollywood Resort & Casino**

| | | SHAREPOINT LIVE! & SQL SERVER LIVE! TRACKS | | MODERN APPS LIVE! TRACK |
|---|---|---|---|---|
| Mobile Web | JavaScript | Developing Apps and Solutions for SharePoint | SQL Server for the Developer | Modern Apps Live! \| Magenic |
| | | SHAREPOINT LIVE! / SQL SERVER LIVE! | | MODERN APPS LIVE! |
| | | **LWM04** Workshop: Building Apps with SharePoint 2013 - *Darrin Bishop* | | **LWM05** Crash Course on Technologies for Modern App Development - *Nick Landry & Kevin Ford* |
| **LT05** One ASP.NET - Game Changers for 2013 - *Jesse Liberty* | | **LT06** - Introduction to In-Memory OLTP using Hekaton in SQL Server 2014 - *Kevin Goff* | | **LT07** Introduction to Modern Apps Development - *Rockford Lhotka* |
| **LT12** What's New in MVC 5 - *Miguel Castro* | | **LT13** SQL Server 2012/2014 Columnstore Indexes - *Kevin Goff* | | **LT14** Modern Apps Architecture - *Rockford Lhotka* |
| | | **LT17** Chalk Talk: Access Services in SharePoint 2013 - What is it? - *Darrin Bishop* | | **LT18** Chalk Talk: Using Git for Cross-Platform Source Control - *Benjamin Day* |
| **LT23** Build Real Time Websites with SignalR - *Rachel Appel* | | **LT24** SQL Server in Windows Azure Virtual Machines - *Eric D. Boyd* | | **LT25** ALM with Visual Studio Online and Git - *Brian Randell* |
| **LT30** What's New in Web API 2 - *Miguel Castro* | | **LT31** T-SQL for Application Developers - Attendees Choose - *Kevin Goff* | | **LT32** Managing a Distributed Project Team - *Kevin Ford* |
| **LW05** Going Native with KendoUI Mobile - *Keith Burnell* | | **LW06** How SharePoint Apps are Going to Save Us! - *Bill Ayers* | | **LW07** Application Servers and Windows Azure - *Kevin Ford* |
| **LW12** Building Cross-Platform Phone Apps & Sites with jQuery Mobile - *Nick Landry* | | **LW13** Developing Using the SharePoint 2013 REST Services - *Matthew DiFranco* | | **LW14** Modern App Design - *Billy Hollis* |
| **LW19** Tips for Building Multi-Touch Enabled Web Sites - *Ben Hoelting* | | **LW20** Introduction to Office 365 SharePoint Online Development - *Matthew DiFranco* | | **LW21** User Expperience Nuts and Bolts - *Billy Hollis & Anthony Handley* |
| **LW26** Build Once Deploy Everywhere with Icenium from Telerik - *Ben Hoelting* | | **LW27** Set Your Code Free! SharePoint Apps for the Mobile World - *Bill Ayers* | | **LW28** Data Access and SQL Server - *Steve Hughes & Dave Stienessen* |
| **LW33** Building OData Services with ASP.NET Web API - *Brian Noyes* | | **LW34** Test-Driven SharePoint 2013 App Development - *Bill Ayers* | | **LW35** Unit Testing Modern Apps - *Jason Bock* |
| **LTH05** Slice Development Time with ASP.NET MVC, Visual Studio, and Razor - *Philip Japikse* | | **LTH06** To Be Announced | | **LTH07** Building an App for Android with Xamarin - *Nick Landry* |
| **LTH12** OWIN and Katana: The New HTTP Host for .NET Web Applications - *Brock Allen* | | **LTH13** Excel, Power BI and You: An Analytics Superhub - *Andrew Brust* | | **LTH14** Leveraging Your Data Through Business Intelligence -*Steve Hughes* |
| **LTH19** Creating Map Centric Web Applications with HTML5 and JavaScript - *Ben Ramseth* | | **LTH20** Big Data 101 with HDInsight - *Andrew Brust* | | **LTH21** Building a WinRT and WinPRT App with C# and XAML -*Brent Edwards* |
| **LTH26** Internals of Security for Web Applications in Visual Studio 2013 - *Brock Allen* | | **LTH27** Database Development with SQL Server Data Tools - *Leonard Lobel* | | **LTH28** Building a Modern iPad (iOS) -*Lou Miranda* |
| **LTH33** Identity Management in ASP.NET - *Brock Allen* | | **LTH34** Windows Azure SQL Database – SQL Server in the Cloud - *Leonard Lobel* | | **LTH35** Building a 'Smart Client' HTML5 and JavaScript App - *Aiden Ryan* |
| | | **LWF04** Workshop: SQL Server for Developers - *Andrew Brust & Leonard Lobel* | | **LWF05** Workshop: Deep Dive into Modern App Design and Development - *Brent Edwards, Anthony Handley, Lou Miranda, & Aidan Ryan* |

VISUAL STUDIO LIVE! HAS PARTNERED WITH PLANET HOLLYWOOD RESORT & CASINO ON A SPECIAL REDUCED RATE OF $109/NIGHT.

CONNECT WITH VISUAL STUDIO LIVE!

**twitter.com/vslive – @VSLive**

**facebook.com – Search "VSLive"**

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

**Register by February 12 and Save $300!**

Use promo code VSLFEB4

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

# Intercepting Asynchronous Methods Using Unity Interception

Fernando Simonazzi and Grigori Melnik

Unity (not to be confused with the Unity3D game engine) is a general-purpose, extensible, dependency injection container with interception support for use in any type of Microsoft .NET Framework-based application. Unity is designed and maintained by the Microsoft patterns & practices team (microsoft.com/practices). It can be easily added to your application via NuGet, and you'll find the main hub of learning resources related to Unity at msdn.com/unity.

This article focuses on Unity interception. Interception is a technique that's useful when you want to modify the behavior of individual objects without affecting the behavior of other objects from the same class, very much as you'd do when using the Decorator pattern (the Wikipedia definition of the Decorator pattern can be found here: bit.ly/1gZZUQu). Interception provides a flexible approach for adding new behaviors to an object at run time. These behaviors typically address some crosscutting concerns, such as logging or data validation. Interception is often used as the underlying mechanism for aspect-oriented programming (AOP). The runtime interception feature in Unity allows you to effectively intercept method calls to objects and perform pre- and post-processing of these calls.

Interception in the Unity container has two main components: interceptors and interception behaviors. Interceptors determine the mechanism used to intercept the calls to methods in the intercepted object, while the interception behaviors determine the actions that are performed on the intercepted method calls. An intercepted object is supplied with a pipeline of interception behaviors. When a method call is intercepted, each behavior in the pipeline is allowed to inspect and even modify the parameters of the method call, and eventually the original method implementation is invoked. Upon return, each behavior can inspect or replace the values returned or exceptions thrown by the original implementation or the previous behavior in the pipeline. Finally, the original caller gets the resulting return value, if any, or the resulting exception. **Figure 1** depicts the interception mechanism.

There are two types of interception techniques: instance interception and type interception. With instance interception, Unity dynamically creates a proxy object that's inserted between the client and the target object. The proxy object is then responsible for passing the calls made by the client to the target object through the behaviors. You can use Unity instance interception to intercept objects created both by the Unity container and outside of the container, and you can use it to intercept both virtual and non-virtual methods. However, you can't cast the dynamically created proxy type to the type of the target object. With type interception, Unity dynamically creates a new type that derives from the type of the target object and that includes the behaviors that handle the

This article discusses:

- The runtime interception feature in Unity
- Intercepting TAP asynchronous methods
- Dealing with generics
- An alternative to using await

Technologies discussed:

C# 5.0, Unity 3

Code download available at:

msdn.microsoft.com/magazine/msdnmag0214

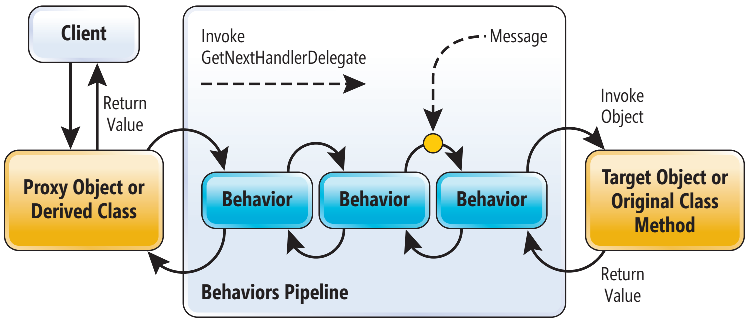

Client
Return Value
Proxy Object or Derived Class
Invoke GetNextHandlerDelegate
Message
Behavior
Behavior
Behavior
Behaviors Pipeline
Invoke Object
Target Object or Original Class Method
Return Value

Figure 1 **Unity Interception Mechanism**

crosscutting concerns. The Unity container instantiates objects of the derived type at run time. Instance interception can only intercept public instance methods. Type interception can intercept both public and protected virtual methods. Keep in mind that, due to platform constraints, Unity interception does not support Windows Phone and Windows Store app development, though the core Unity container does.

For a primer on Unity, see "Dependency Injection with Unity" (Microsoft patterns & practices, 2013) at amzn.to/16rfy0B. For more information about interception in the Unity container, see the MSDN Library article, "Interception using Unity," at bit.ly/1cWCnwM.

## Intercepting Task-Based Asynchronous Pattern (TAP) Asynchronous Methods

The interception mechanism is simple enough, but what happens if the intercepted method represents an asynchronous operation that returns a Task object? In a way, nothing really changes: A method is invoked and returns a value (the Task object) or throws an exception, so it can be intercepted just like any other method. But you're probably interested in dealing with the actual outcome of the asynchronous operation rather than the Task representing it. For example, you might want to log the Task's return value, or handle any exception the Task might produce.

Figure 2 **Simple Interception**

```
public class LoggingAsynchronousOperationInterceptionBehavior : IInterceptionBehavior
{
  public IMethodReturn Invoke(IMethodInvocation input,
    GetNextInterceptionBehaviorDelegate getNext)
  {
    // Execute the rest of the pipeline and get the return value
    IMethodReturn value = getNext()(input, getNext);

    return value;
  }

  #region additional interception behavior methods

  public IEnumerable<Type> GetRequiredInterfaces()
  {
    return Type.EmptyTypes;
  }

  public bool WillExecute
  {
    get { return true; }
  }

  #endregion
}
```

Fortunately, having an actual object representing the outcome of the operation makes interception of this asynchronous pattern relatively simple. Other asynchronous patterns are quite a bit more difficult to intercept: In the Asynchronous Programming Model (bit.ly/ICl8aH) two methods represent a single asynchronous operation, while in the Event-based Asynchronous Pattern (bit.ly/19VdUWu) asynchronous operations are represented by a method to initiate the operation and an associated event to signal its completion.

In order to accomplish the interception of the asynchronous TAP operation, you can replace the Task returned by the method with a new Task that performs the necessary post-processing after the original Task completes. Callers of the intercepted method will receive the new Task matching the method's signature, and will observe the result of the intercepted method's implementation, plus whatever extra processing the interception behavior performs.

Fortunately, having an actual object representing the outcome of the operation makes interception of this asynchronous pattern relatively simple.

We'll develop a sample implementation of the basic approach to intercept TAP asynchronous operations in which we want to log the completion of asynchronous operations. You can adapt this sample to create your own behaviors that can intercept asynchronous operations.

## Simple Case

Let's start with a simple case: intercepting asynchronous methods that return a non-generic Task. We need to be able to detect that the intercepted method returns a Task and replace that Task with a new one that performs the appropriate logging.

Figure 3 **Returning a Task**

```
public IMethodReturn Invoke(IMethodInvocation input,
  GetNextInterceptionBehaviorDelegate getNext)
{
  // Execute the rest of the pipeline and get the return value
  IMethodReturn value = getNext()(input, getNext);

  // Deal with tasks, if needed
  var method = input.MethodBase as MethodInfo;
  if (value.ReturnValue != null
    && method != null
    && typeof(Task) == method.ReturnType)
  {
    // If this method returns a Task, override the original return value
    var task = (Task)value.ReturnValue;
    return input.CreateMethodReturn(this.CreateWrapperTask(task, input),
      value.Outputs);
  }

  return value;
}
```

Figure 4 **Logging the Outcome**

```
private async Task CreateWrapperTask(Task task,
  IMethodInvocation input)
{
  try
  {
    await task.ConfigureAwait(false);
    Trace.TraceInformation("Successfully finished async operation {0}",
      input.MethodBase.Name);
  }
  catch (Exception e)
  {
    Trace.TraceWarning("Async operation {0} threw: {1}",
      input.MethodBase.Name, e);
    throw;
  }
}
```

We can take the "no op" interception behavior shown in **Figure 2** as a starting point.

Next, we add the code to detect the task-returning methods and replace the returned Task with a new wrapper Task that logs the outcome. To accomplish this, the CreateMethodReturn on the input object is called to create a new IMethodReturn object representing a wrapper Task created by the new CreateWrapperTask method in the behavior, as shown in **Figure 3**.

The new CreateWrapperTask method returns a Task that waits for the original Task to complete and logs its outcome, as shown in **Figure 4**. If the task resulted in an exception, the method will rethrow it after logging it. Note that this implementation doesn't change the original Task's outcome, but a different behavior could replace or ignore the exceptions the original Task might introduce.

Figure 5 **A Generic Method to Handle Task<T>**

```
private async Task<T> CreateGenericWrapperTask<T>(Task<T> task,
  IMethodInvocation input)
{
  try
  {
    T value = await task.ConfigureAwait(false);
    Trace.TraceInformation("Successfully finished async operation {0}
with value: {1}",
      input.MethodBase.Name, value);
    return value;
  }
  catch (Exception e)
  {
    Trace.TraceWarning("Async operation {0} threw: {1}", input.MethodBase.Name, e);
    throw;
  }
}
```

Figure 6 **Splitting the Delegate Creation Method**

```
private Task CreateGenericWrapperTask<T>(Task task, IMethodInvocation input)
{
  return this.DoCreateGenericWrapperTask<T>((Task<T>)task, input);
}

private async Task<T> DoCreateGenericWrapperTask<T>(Task<T> task,
  IMethodInvocation input)
{
  try
  {
    T value = await task.ConfigureAwait(false);
    Trace.TraceInformation("Successfully finished async operation {0}
with value: {1}",
      input.MethodBase.Name, value);
    return value;
  }
  catch (Exception e)
  {
    Trace.TraceWarning("Async operation {0} threw: {1}", input.MethodBase.Name, e);
    throw;
  }
}
```

## Dealing with Generics

Dealing with methods that return Task<T> is a bit more complex, particularly if you want to avoid taking a performance hit. Let's ignore for now the problem of figuring out what "T" is and assume it's already known. As **Figure 5** shows, we can write a generic method that can handle Task<T> for a known "T," taking advantage of the asynchronous language features available in C# 5.0.

As with the simple case, the method just logs without changing the original behavior. But because the wrapped Task now returns a value, the behavior could also replace this value, if needed.

How can we invoke this method to obtain the replacement Task? We need to resort to reflection, extracting the T from the intercepted method's generic return type, creating a closed version of this generic method for that T and creating a delegate out of it and, finally, invoking the delegate. This process can be quite expensive, so it's a good idea to cache these delegates. If the T is part of the method's signature, we wouldn't be able to create a delegate of a method and invoke it without knowing the T, so we'll split our earlier method into two methods: one with the desired signature, and one that benefits from the C# language features, as shown in **Figure 6**.

Next, we change the interception method so we use the correct delegate to wrap the original task, which we get by invoking the new GetWrapperCreator method and passing the expected task type. We don't need a special case for the non-generic Task, because it can fit the delegate approach just like the generic Task<T>. **Figure 7** shows the updated Invoke method.

All that's left is implementing the GetWrapperCreator method. This method will perform the expensive reflection calls to create the delegates and use a ConcurrentDictionary to cache them. These wrapper creator delegates are of type Func<Task, IMethodInvocation, Task>; we want to get the original task and the IMethodInvocation object representing the call to the invocation to the asynchronous method and return a wrapper Task. This is shown in **Figure 8**.

For the non-generic Task case, no reflection is needed and the existing non-generic method can be used as the desired delegate as is. When dealing with Task<T>, the necessary reflection calls are

Figure 7 **The Updated Invoke Method**

```
public IMethodReturn Invoke(IMethodInvocation input,
  GetNextInterceptionBehaviorDelegate getNext)
{
  IMethodReturn value = getNext()(input, getNext);
  var method = input.MethodBase as MethodInfo;

  if (value.ReturnValue != null
    && method != null
    && typeof(Task).IsAssignableFrom(method.ReturnType))
  {
    // If this method returns a Task, override the original return value
    var task = (Task)value.ReturnValue;
    return input.CreateMethodReturn(
      this.GetWrapperCreator(method.ReturnType)(task, input), value.Outputs);
  }

  return value;
}
```

Figure 8 **Implementing the GetWrapperCreator Method**

```
private readonly ConcurrentDictionary<Type, Func<Task, IMethodInvocation, Task>>
  wrapperCreators = new ConcurrentDictionary<Type, Func<Task,
  IMethodInvocation, Task>>();

private Func<Task, IMethodInvocation, Task> GetWrapperCreator(Type taskType)
{
  return this.wrapperCreators.GetOrAdd(
    taskType,
    (Type t) =>
    {
      if (t == typeof(Task))
      {
        return this.CreateWrapperTask;
      }
      else if (t.IsGenericType && t.GetGenericTypeDefinition() == typeof(Task<>))
      {
        return (Func<Task, IMethodInvocation, Task>)this.GetType()
          .GetMethod("CreateGenericWrapperTask",
             BindingFlags.Instance | BindingFlags.NonPublic)
          .MakeGenericMethod(new Type[] { t.GenericTypeArguments[0] })
          .CreateDelegate(typeof(Func<Task, IMethodInvocation, Task>), this);

      }
      else
      {
        // Other cases are not supported
        return (task, _) => task;
      }
    });
}
```

performed to create the corresponding delegate. Finally, we can't support any other Task type, as we wouldn't know how to create it, so a no-op delegate that just returns the original task is returned.

This behavior can now be used on an intercepted object and will log the results of the tasks returned by the intercepted object's methods for the cases where a value is returned and where an exception is thrown. The example in **Figure 9** shows how a container can be configured to intercept an object and use this new behavior, and the resulting output when different methods are invoked.

Figure 9 **Configuring a Container to Intercept an Object and Use the New Behavior**

```
using (var container = new UnityContainer())
{
  container.AddNewExtension<Interception>();
  container.RegisterType<ITestObject, TestObject>(
    new Interceptor<InterfaceInterceptor>(),
    new InterceptionBehavior<LoggingAsynchronousOperationInterceptionBehavior>());

  var instance = container.Resolve<ITestObject>();

  await instance.DoStuffAsync("test");

  // Do some other work
}

Output:
vstest.executionengine.x86.exe Information: 0 :
  Successfully finished async operation DoStuffAsync with value: test

vstest.executionengine.x86.exe Warning: 0 :
  Async operation DoStuffAsync threw:
    System.InvalidOperationException: invalid
   at AsyncInterception.Tests.AsyncBehaviorTests2.TestObject.<
     DoStuffAsync>d__38.MoveNext() in d:\dev\interceptiontask\
       AsyncInterception\AsyncInterception.Tests\
         AsyncBehaviorTests2.cs:line 501
--- End of stack trace from previous location where exception was thrown ---
   at System.Runtime.CompilerServices.TaskAwaiter.ThrowForNonSuccess(Task task)
   at System.Runtime.CompilerServices.TaskAwaiter.
     HandleNonSuccessAndDebuggerNotification(Task task)
   at System.Runtime.CompilerServices.TaskAwaiter.GetResult()
   at AsyncInterception.LoggingAsynchronousOperationInterceptionBehavior.<
     CreateWrapperTask>d__3.MoveNext() in d:\dev\interceptiontask\
       AsyncInterception\AsyncInterception\
         LoggingAsynchronousOperationInterceptionBehavior.cs:line 63
```

Figure 10 **Using ContinueWith instead of the Await Keyword**

```
private Task CreateWrapperTask(Task task, IMethodInvocation input)
{
  var tcs = new TaskCompletionSource<bool>();

  task.ContinueWith(
    t =>
    {
      if (t.IsFaulted)
      {
        var e = t.Exception.InnerException;
        Trace.TraceWarning("Async operation {0} threw: {1}",
          input.MethodBase.Name, e);
        tcs.SetException(e);
      }
      else if (t.IsCanceled)
      {
        tcs.SetCanceled();
      }
      else
      {
        Trace.TraceInformation("Successfully finished async operation {0}",
          input.MethodBase.Name);
        tcs.SetResult(true);
      }
    },
    TaskContinuationOptions.ExecuteSynchronously);

  return tcs.Task;
}
```

## Covering Our Tracks

As you can see in the resulting output in **Figure 9**, the approach used in this implementation results in a light change in the exception's stack trace, reflecting the way the exception was rethrown when awaiting for the task. An alternative approach can use the ContinueWith method and a TaskCompletionSource<T> instead of the await keyword to avoid this issue, at the expense of having a more complex (and potentially more expensive) implementation such as what's shown in **Figure 10**.

## Wrapping Up

We discussed several strategies for intercepting asynchronous methods and demonstrated them on an example that logs the completion of asynchronous operations. You can adapt this sample to create your own intercepting behaviors that would support asynchronous operations. Full source code for the example is available at msdn.microsoft.com/magazine/msdnmag0214. ■

**Fernando Simonazzi** *is a software developer and architect with more than 15 years of professional experience. He has been a contributor to Microsoft patterns & practices projects, including several releases of the Enterprise Library, Unity, CQRS Journey and Prism. Simonazzi is also an associate at Clarius Consulting.*

**Dr. Grigori Melnik** *is a principal program manager on the Microsoft patterns & practices team. These days he drives the Microsoft Enterprise Library, Unity, CQRS Journey and NUI patterns projects. Prior to that, he was a researcher and software engineer long enough ago to remember the joy of programming in Fortran. Dr. Melnik blogs at blogs.msdn.com/agile.*

**Thanks** to the following technical expert for reviewing this article: *Stephen Toub (Microsoft)*

# A 2D Portal into a 3D World

If you're well-versed in 2D graphics, you might assume that 3D is similar except for the extra dimension. Not quite! Anyone who's dabbled in 3D graphics programming knows how difficult it is. 3D graphics programming requires you to master new and exotic concepts beyond anything encountered in the conventional 2D world. A lot of preliminaries are required to get just a little 3D on the screen, and even then a slight miscalculation can render it invisible. Consequently, the visual feedback so important to learning graphics programming is delayed until all the programming pieces are in place and working in harmony.

DirectX acknowledges the profound difference between 2D and 3D graphics programming with the division between Direct2D and Direct3D. Although you can mix 2D and 3D content on the same output device, these are very distinct and different programming interfaces, and there's no middle ground. DirectX doesn't allow you to be a little bit country, a little bit rock-and-roll.

Or does it?

Interestingly, Direct2D includes some concepts and facilities that originated in the 3D programming universe. Through features such as geometry tessellation (the decomposition of complex geometries into triangles) and 2D effects using shaders (which consist of special code that runs on the graphics processing unit, or GPU), it's possible to exploit some powerful 3D concepts while still remaining within the context of Direct2D.

Moreover, these 3D concepts can be encountered and explored gradually, and you get the satisfaction of actually seeing the results on the screen. You can get your 3D feet wet in Direct2D so a later plunge into Direct3D programming is a little less shocking.

I guess it shouldn't be all that surprising that Direct2D incorporates some 3D features. Architecturally, Direct2D is built on top of Direct3D, which allows Direct2D to also take advantage of the hardware acceleration of the GPU. This relationship between Direct2D and Direct3D becomes more apparent as you begin exploring the nether regions of Direct2D.

Code download available at msdn.microsoft.com/magazine/msdnmag0214.

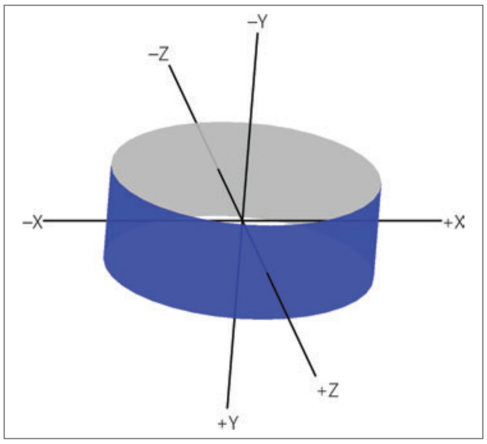


Figure 1 **The Coordinate System Used for the Programs in This Article**

I'll commence this exploration with a review of 3D coordinates and coordinate systems.

## The Big Leap Outward

If you've been following this column in recent months, you know it's possible to call the GetGlyphRunOutline method of an object that implements the IDWriteFontFace interface to obtain an ID2D1PathGeometry instance that describes the outlines of text characters in terms of straight lines and Bézier curves. You can then manipulate the coordinates of these lines and curves to distort the text characters in various ways.

It's also possible to convert the 2D coordinates of a path geometry into 3D coordinates, and then manipulate these 3D coordinates before converting them back into 2D to display the path geometry normally. Does that sound like fun?

Coordinates in two-dimensional space are expressed as number pairs (X, Y), which correspond to a location on the screen; 3D coordinates are in the form (X, Y, Z) and, conceptually, the Z axis is orthogonal to the screen. Unless you're dealing with a holographic display or a 3D printer, these Z coordinates aren't nearly as real as X and Y coordinates.

There are other differences between 2D and 3D coordinate systems. Conventionally the 2D origin—the point (0, 0)—is the upper-left corner of the display device. The X coordinates increase to the right and Y coordinates increase going down. In 3D, very often the origin is in the center of the screen, and it's more akin to a standard Cartesian coordinate system: The X coordinates still increase going to the right, but the Y coordinates increase going up, and there are negative coordinates as well. (Of course, the origin, scale, and orientation of these axes can be altered with matrix transforms, and usually are.)

Conceptually, the positive Z axis can either point out of the screen or point into the screen. These two conventions are known as "right-hand" and "left-hand" coordinate systems, referring to a technique to distinguish them: With a right-hand coordinate system, if you point the index finger of your right hand in the direction of the positive X axis, and the middle finger in the direction of positive Y, your thumb points to positive Z. Also, if you curve the fingers of your right hand from the positive X axis to the

Figure 2 **The Polygon Class for Storing Closed Path Figures**

```
struct Polygon
{
  // Constructors
  Polygon()
  {
  }

  Polygon(size_t pointCount)
  {
    Points = std::vector<D2D1_POINT_2F>(pointCount);
  }

  // Move constructor
  Polygon(Polygon && other) : Points(std::move(other.Points))
  {
  }

  std::vector<D2D1_POINT_2F> Points;

  static HRESULT CreateGeometry(ID2D1Factory* factory,
                                const std::vector<Polygon>& polygons,
                                ID2D1PathGeometry** pathGeometry);
};

HRESULT Polygon::CreateGeometry(ID2D1Factory* factory,
                                const std::vector<Polygon>& polygons,
                                ID2D1PathGeometry** pathGeometry)
{
  HRESULT hr;

  if (FAILED(hr = factory->CreatePathGeometry(pathGeometry)))
    return hr;

  Microsoft::WRL::ComPtr<ID2D1GeometrySink> geometrySink;

  if (FAILED(hr = (*pathGeometry)->Open(&geometrySink)))
    return hr;

  for (const Polygon& polygon : polygons)
  {
    if (polygon.Points.size() > 0)
    {
      geometrySink->BeginFigure(polygon.Points[0],
                                D2D1_FIGURE_BEGIN_FILLED);

      if (polygon.Points.size() > 1)
      {
        geometrySink->AddLines(polygon.Points.data() + 1,
                               polygon.Points.size() - 1);
      }
      geometrySink->EndFigure(D2D1_FIGURE_END_CLOSED);
    }
  }
  return geometrySink->Close();
}
```

positive Y axis, your thumb points to positive Z. With a left-hand coordinate system, it's the same except using the left hand.

My goal here is to obtain a 2D path geometry of a short text string, and then twist it around the origin into a 3D ring so the beginning meets the end, similar to the illustration shown in **Figure 1**. Because I'll be converting 2D coordinates to 3D coordinates and then back to 2D, I've chosen to use a 3D coordinate system with Y coordinates increasing going down, just like in 2D. The positive Z axis comes out of the screen, but it's really a left-hand coordinate system.

To make this whole job as easy as possible, I've used a font file stored as a program resource, and created an IDWriteFontFile object for obtaining the IDWriteFontFace object. Alternatively, you could obtain an IDWriteFontFace through a more roundabout method from the system font collection.

The ID2D1PathGeometry object generated from the GetGlyphRunOutline method is then passed through the Simplify method using the D2D1_GEOMETRY_SIMPLIFICATION_OPTION_LINES argument to flatten all Bézier curves into sequences of short lines. That simplified geometry is passed into a custom ID2D1GeometrySink implementation named FlattenedGeometrySink to further decompose all the straight lines into much shorter straight lines. The result is a completely malleable geometry consisting only of lines.

To ease the manipulation of these coordinates, FlattenedGeometrySink generates a collection of Polygon objects. **Figure 2** shows the definition of the Polygon structure. It's basically just a collection of connected 2D points. Each Polygon object corresponds to a closed figure in the path geometry. Not all figures in path geometries are closed, but those in text glyphs are always closed, so this structure is fine for that purpose. Some characters (such as C, E and X) are described by just one Polygon; some (A, D and O) consist of two Polygon objects for the inside and outside; some (B, for example) consist of three; and some symbol characters may have many more.

Among the downloadable code for this column is a Windows Store program named CircularText that creates a collection of Polygon objects based on the text "Text in an Infinite Circle of," where the end is intended to connect back to the beginning in a circle. The text string is actually specified in the program as "ext in an Infinite Circle of T" to avoid a space at the beginning or end that would disappear when a path geometry is generated from the glyphs.

The CircularTextRenderer class in the CircularText project contains two std::vector objects of type Polygon called m_srcPolygons (the original Polygon objects generated from the path geometry) and m_dstPolygons (the Polygon objects used to generate the rendered path geometry). **Figure 3** shows the method

Figure 3 **From 2D to 3D to 2D in the CircularText Program**

```
void CircularTextRenderer::CreateWindowSizeDependentResources()
{
  // Get window size and geometry size
  Windows::Foundation::Size logicalSize = m_deviceResources->GetLogicalSize();
  float geometryWidth = m_geometryBounds.right - m_geometryBounds.left;
  float geometryHeight = m_geometryBounds.bottom - m_geometryBounds.top;

  // Calculate a few factors for converting 2D to 3D
  float radius = logicalSize.Width / 2 - 50;
  float circumference = 2 * 3.14159f * radius;
  float scale = circumference / geometryWidth;
  float height = scale * geometryHeight;

  for (size_t polygonIndex = 0; polygonIndex < m_srcPolygons.size(); polygonIndex++)
  {
    const Polygon& srcPolygon = m_srcPolygons.at(polygonIndex);
    Polygon& dstPolygon = m_dstPolygons.at(polygonIndex);

    for (size_t pointIndex = 0; pointIndex < srcPolygon.Points.size(); pointIndex++)
    {
      const D2D1_POINT_2F pt = srcPolygon.Points.at(pointIndex);
      float radians = 2 * 3.14159f * (pt.x - m_geometryBounds.left) / geometryWidth;
      float x = radius * sin(radians);
      float z = radius * cos(radians);
      float y = height * ((pt.y - m_geometryBounds.top) / geometryHeight - 0.5f);
      dstPolygon.Points.at(pointIndex) = Point2F(x, y);
    }
  }

  // Create path geometry from Polygon collection
  DX::ThrowIfFailed(
    Polygon::CreateGeometry(m_deviceResources->GetD2DFactory(),
                            m_dstPolygons,
                            &m_pathGeometry)
    );
}
```

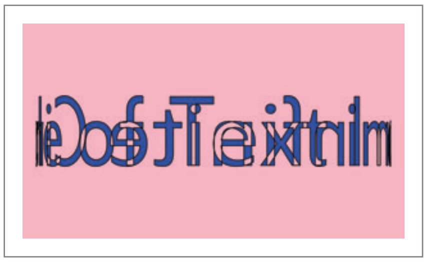


Figure 4 **The CircularText Display**

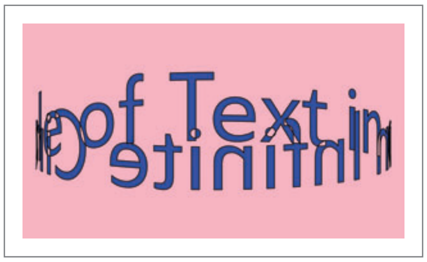


Figure 5 **The Tilted CircularText Display**

CreateWindowSizeDependentResources that converts the source polygons to the destination polygons based on the size of the screen.

In the inner loop, you'll see x, y and z values calculated. This is a 3D coordinate but it's not even saved. Instead, it's immediately collapsed back into 2D by simply ignoring the z value. To calculate those 3D coordinates, the code first converts a horizontal position on the original path geometry to an angle in radians from 0 to 2π. The sin and cos functions calculate a position on a unit circle on the XZ plane. The y value is a more direct conversion from the vertical coordinates of the original path geometry.

The CreateWindowSizeDependentResources method concludes by obtaining a new ID2D1PathGeometry object from the destination Polygon collection. The Render method then sets a matrix transform to put the origin in the center of the screen, and both fills and outlines this path geometry, with the result shown in **Figure 4**.

Is the program working? It's hard to tell! Look closely and you can see some wide characters in the center and narrower characters at the left and right. But the big problem is that I started out with a path geometry with no intersecting lines, and now the geometry is displayed back over itself, with the result that overlapping areas are not filled. This effect is characteristic of geometries, and it happens whether the path geometry created by the Polygon structure has a fill mode of alternate or winding.

## Getting Some Perspective

Three-dimensional graphics programming is not just about coordinate points. Visual cues are necessary for the viewer to interpret an image on a 2D screen as representing an object in 3D space. In the real world, you rarely view objects from a constant vantage point. You'd get a better view of the 3D text in **Figure 4** if you could tilt it somewhat so it looks more like the ring in **Figure 1**.

To get some perspective on the three-dimensional text, the coordinates need to be rotated in space. As you know, Direct2D supports a matrix transform structure named D2D1_MATRIX_3x2_F that you can use to define 2D transforms, which you can apply to your 2D graphics output by first calling the SetTransform method of ID2D1RenderTarget.

Most commonly, you would use a class named Matrix3x2F from the D2D1 namespace for this purpose. This class derives from D2D1_MATRIX_3x2F_F and provides methods for defining various types of standards for translation, scaling, rotation and skew.

The Matrix3x2F class also defines a method named TransformPoint that allows you to apply the transform "manually" to individual D2D1_POINT_2F objects. This is useful for manipulating points before they're rendered.

You may think I need a 3D rotation matrix to tilt the displayed text. I'll certainly be exploring 3D matrix transforms in future columns, but for now I can make do with 2D rotation. Imagine yourself situated somewhere on the negative X axis of **Figure 1**, looking toward the origin. The positive Z and Y axes are situated just like the X and Y axes in a conventional 2D coordinate system, so it seems plausible that by applying a 2D rotation matrix to the Z and Y values, I can rotate all the coordinates around the three-dimensional X axis.

Figure 6 **The Update Method of SpinningCircularText**

```
void SpinningCircularTextRenderer::Update(DX::StepTimer const& timer)
{
  // Get window size and geometry size
  Windows::Foundation::Size logicalSize = m_deviceResources->GetLogicalSize();
  float geometryWidth = m_geometryBounds.right - m_geometryBounds.left;
  float geometryHeight = m_geometryBounds.bottom - m_geometryBounds.top;

  // Calculate a few factors for converting 2D to 3D
  float radius = logicalSize.Width / 2 - 50;
  float circumference = 2 * 3.14159f * radius;
  float scale = circumference / geometryWidth;
  float height = scale * geometryHeight;

  // Calculate rotation matrix
  float rotateAngle = -360 * float(fmod(timer.GetTotalSeconds(), 10)) / 10;
  Matrix3x2F rotateMatrix = Matrix3x2F::Rotation(rotateAngle);

  // Calculate tilt matrix
  Matrix3x2F tiltMatrix = Matrix3x2F::Rotation(m_tiltAngle);

  for (size_t polygonIndex = 0; polygonIndex < m_srcPolygons.size(); polygonIndex++)
  {
    const Polygon& srcPolygon = m_srcPolygons.at(polygonIndex);
    Polygon& dstPolygon = m_dstPolygons.at(polygonIndex);

    for (size_t pointIndex = 0; pointIndex < srcPolygon.Points.size(); pointIndex++)
    {
      const D2D1_POINT_2F pt = srcPolygon.Points.at(pointIndex);
      float radians = 2 * 3.14159f * (pt.x - m_geometryBounds.left) / geometryWidth;
      float x = radius * sin(radians);
      float z = radius * cos(radians);
      float y = height * ((pt.y - m_geometryBounds.top) / geometryHeight - 0.5f);

      // Apply rotation to X and Z
      D2D1_POINT_2F rotatedPoint = rotateMatrix.TransformPoint(Point2F(x, z));
      x = rotatedPoint.x;
      z = rotatedPoint.y;

      // Apply tilt to Y and Z
      D2D1_POINT_2F tiltedPoint = tiltMatrix.TransformPoint(Point2F(y, z));
      y = tiltedPoint.x;
      z = tiltedPoint.y;

      dstPolygon.Points.at(pointIndex) = Point2F(x, y);
    }
  }

  // Create path geometry from Polygon collection
  DX::ThrowIfFailed(
    Polygon::CreateGeometry(m_deviceResources->GetD2DFactory(),
    m_dstPolygons,
    &m_pathGeometry)
    );

    // Update FPS display text
    uint32 fps = timer.GetFramesPerSecond();

    m_text = (fps > 0) ? std::to_wstring(fps) + L" FPS" : L" - FPS";
}
```

You can experiment with this with the CircularText program. Create a 2D rotation matrix in the CreateWindowSizeDependentResources method sometime before the Polygon coordinates are manipulated:

```
Matrix3x2F tiltMatrix = Matrix3x2F::Rotation(-8);
```

That's a rotation of -8 degrees, and the negative sign indicates a counterclockwise rotation. In the inner loop, after x, y, and z have been calculated, apply that transform to the z and y values as if they were x and y values:

```
D2D1_POINT_2F tiltedPoint =
    tiltMatrix.TransformPoint(Point2F(z, y));
z = tiltedPoint.x;
y = tiltedPoint.y;
```

**Figure 5** shows what you'll see.

This is much better, but it still has issues. Ugly things happen when the geometry overlaps, and there's nothing to suggest which part of the geometry is nearer to you and which is further away. Stare at it, and you might experience some perspective shift.

Still, the ability to apply 3D transforms to this object suggests that it might also be easy to rotate the object around the Y axis—and it is. If you imagine viewing the origin from the positive Y axis, you'll see that the X and Z axes are oriented the same way as the X and Y axes in a 2D coordinate system.

The SpinningCircularText project implements two rotation transforms to spin the text and tilt it. All the computational logic previously in CreateWindowSizeDependentResources has been moved into the Update method. The 3D points are rotated twice: once around the X axis based on elapsed time, and then around the Y axis based on the user sweeping a finger up and down the screen. This Update method is shown in **Figure 6**.

It's well-known that composite matrix transforms are equivalent to matrix multiplications, and because matrix multiplications aren't commutative, neither are composite transforms. Try switching around the application of the tilt and rotate transforms for a different effect (which you might actually prefer).

When creating the SpinningCircularText program, I adapted the SampleFpsTextRenderer class created by the Visual Studio template to create the SpinningCircularTextRenderer class, but I left the display of the rendering rate. This allows you to see how bad the performance is. On my Surface Pro, I see a frames per second (FPS) figure of 25 in Debug mode, which indicates the code is not keeping up with the refresh rate of the video display.

If you don't like that performance, I'm afraid I have some bad news: I'm going to make it even worse.

## Separating Foreground from Background

The biggest problem with the path geometry approach to 3D is the effect of overlapping areas. Is it possible to avoid those overlaps? The image this program is attempting to draw is not all that complex. At any time, there's a front view of part of the text and a back view of the rest of the text, and the front view should always be displayed on top of the back view. If it were possible to separate the path geometry into two path geometries—one for the background and one for the foreground—you could render those path geometries with separate FillGeometry calls so the foreground would be on top of the background. These two path geometries could even be rendered with different brushes.

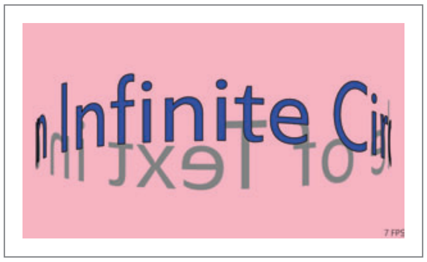


Figure 7 **The OccludedCircularText Display**

Consider the original path geometry created by the GetGlyphRunOutline method. That's just a flat 2D path geometry occupying a rectangular area. Eventually, half of that geometry is displayed in the foreground, and the other half is displayed in the background. But by the time the Polygon objects are obtained, it's too late to make that split with anything like computational ease.

Instead, the original path geometry needs to be broken in half before the Polygon objects are obtained. This break is dependent on the rotation angle, which means that much more logic must be moved into the Update method.

The original path geometry can be split in half with two calls to the CombineWithGeometry method. This method combines two geometries in various ways to make a third geometry. The two geometries that are combined are the original path geometry that describes the text outlines and a rectangle geometry that defines a subset of the path geometry. This subset appears in either the foreground or background, depending on the rotation angle.

For example, if the rotation angle is 0, the rectangle geometry must cover the central half of the path geometry of the text outlines. This is the part of the original geometry that appears in the foreground. Calling CombineWithGeometry with the D2D1_COMBINE_MODE_INTERSECT mode returns a path geometry consisting only of that central area, while calling CombineWithGeometry with D2D1_COMBINE_MODE_EXCLUDE obtains a path geometry of the remainder—the parts on the left and right. These two path geometries can then be converted to Polygon objects separately for manipulation of the coordinates, followed by a conversion back to separate path geometries for rendering.

This logic is part of a project named OccludedCircularText, which implements the Render method by filling the two geometries with different brushes, as shown in **Figure 7**.

Now it's much more obvious what's in the foreground and what's in the background. Yet, so much computation has been moved to the Update method that performance is very poor.

In a conventional 2D programming environment, I would've exhausted all the 2D programming tools at my disposal and now be stuck with this terrible performance. Direct2D, however, offers an alternative approach to rendering the geometry that simplifies the logic and improves performance immensely. This solution makes use of the most basic 2D polygon—which is a polygon that also plays a major role in 3D programming.

I speak, of course, of the humble triangle. ■

**Charles Petzold** *is a longtime contributor to* MSDN Magazine *and the author of "Programming Windows, 6th edition" (O'Reilly Media, 2012), a book about writing applications for Windows 8. His Web site is charlespetzold.com.*

**Thanks** to the following technical expert for reviewing this article: *Jim Galasyn (Microsoft)*

# Jose, Can You C?

To start my fifth year in this space, I want to tell you about a student who came to Boston for one of my classes. He lives in Miami and I doubt it violates his privacy to tell you that his name is Jose. Over drinks at the Harvard Faculty Club (see msdn.microsoft.com/magazine/dn532211), I asked him how he'd come to the United States. "From Cuba. On a raft. And now here I am at Harvard having drinks with the software legend."

I was (uncharacteristically, you must admit) speechless. But another student of mine, who left Cuba as an infant on a plane to Spain, summed up Jose's journey well: "That took *cojones*."

America has always been a refuge for immigrants. It holds an allure and offers a welcome like nowhere else, especially in technological fields. Think of all the scientists and engineers we've welcomed from abroad, and how they've enriched our country and the world: Bell. Fermi. Einstein. Tesla. Von Neumann.

Andy Grove, another immigrant, writes: "By the time I was 20, I had lived through a Hungarian Fascist dictatorship, German military occupation, the Nazi 'Final Solution,' the siege of Budapest by the Soviet Red Army, a period of chaotic democracy in the years immediately after the war, a variety of repressive Communist regimes, and a popular uprising that was put down at gunpoint." (Busy guy, no?) Grove escaped to the United States in 1957 and co-founded Intel in 1968.

Sometimes our global welcome generates profound ironies. While welcoming Grove, we also welcomed a prime representative of his persecutors: rocket scientist Werner von Braun, member of the Nazi party and the SS. His 1960 biographical movie carried the title, "I Aim at the Stars." Cynics suggested a subtitle: "… But Sometimes I Hit London." Or as Tom Lehrer sang: "Don't say that he's hypocritical / Say rather that he's apolitical. / 'Once the rockets are up, who cares where they come down? / That's not my department,' says Werner von Braun." (You can listen here: bit.ly/1lJpABs.) We overlooked von Braun's past; he built us a moon rocket. Good trade? Bad trade?

Immigrants drive our technological leadership today as well. Think of Sergey Brin from Russia, co-founder of Google. Or Jerry Yang from Taipei, co-founder of Yahoo (although based on the performance of Yahoo lately, maybe we shouldn't count him.)

And it's not just the Nobel-caliber immigrants that enrich our lives. Just think of really smart geeks here in the Windows community: Juval Lowy. Anders Hejlsberg. I'm sure you know plenty yourself.

Immigrants bring us an energy, a zest, that we can't easily duplicate in-house. Having the *chutzpah* to leave your known world behind for something you hope is better requires enormous levels of drive and competence. Think of Jose building his raft and shoving off into the Florida Straits, and the courage that must have required. The Miami Herald in December reported that one person died and two others went missing attempting the very same feat Jose managed so many years ago.

Immigrants appreciate what we have here, things that we natives take for granted. They don't sweat small stuff, such as shopkeepers saying "Happy Holidays" instead of "Merry Christmas." Most of them fled bad governments, usually of a collectivist or totalitarian bent. They especially admire the U.S. Constitution's limits on government. They're quick to smell a rat and sound the alarm when they see it being violated.

Immigrants bring us an energy, a zest, that we can't easily duplicate in-house.

So I say let them in. Throw open the doors to the serious geeks. Maybe we could hold programming contests every year and take the top 1,000 entries. Maybe the top 10 teams at the Microsoft Imagine Cup competition should be offered green cards. (See my October 2011 column at msdn.microsoft.com/magazine/hh456410.) Maybe anyone who earned a Ph.D. in a STEM field could stay here after school, and if they kept their noses clean for 5 years could make that status permanent. You can probably think of other criteria that would work. This idea is so obvious and sensible, though, that it has no chance whatsoever of becoming government policy.

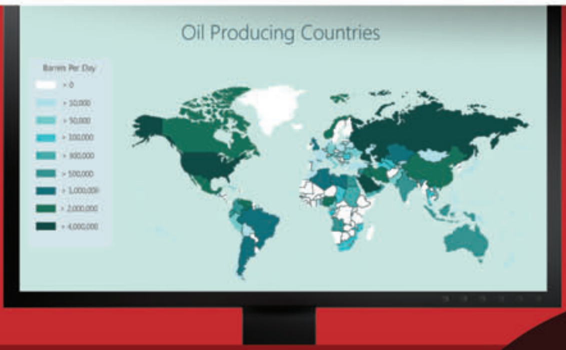
"My sons are born here, Americans for life," said Jose. "They'll never have to do what I did."

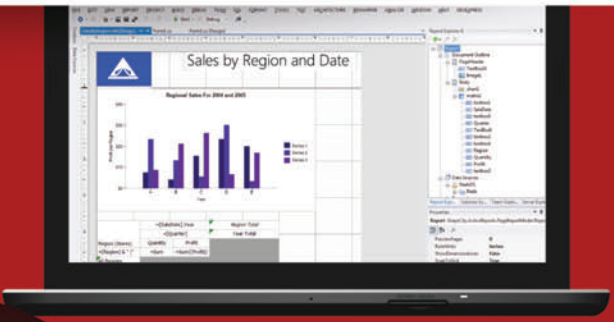
May we forever continue to be a refuge for immigrants. They want our freedom and opportunity. We need their brains. And we need their *cojones*, too. ■

**DAVID S. PLATT** *teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

# REPORTS ANYWHERE

We are the Reporting experts and we care about performance and Visual Studio integration as much as you. The all new ActiveReports 8 reporting engine delivers a granular API that allows you to create concise and beautiful reports. Whether you need multiple outputs, the ability to create maps, barcodes, or the power of HTML5 to display your reports, ActiveReports 8 can handle all of your heavy reporting requirements.





**ALL NEW ACTIVEREPORTS 8**

Sophisticated, Fast & Powerful Reports

Flexible Layouts Using Section, Region & Fixed Page Designers

HTML5 & Touch Optimized Report Viewers for Mobile Devices

A New Scalable, Distributed & Load Balanced Enterprise-grade Report Server

ComponentOne® a division of GrapeCity®

DOWNLOAD YOUR FREE TRIAL

▸ componentone.com

HOBBYIST
LICENSE
OFFER

ONLY
$1

JavaScript

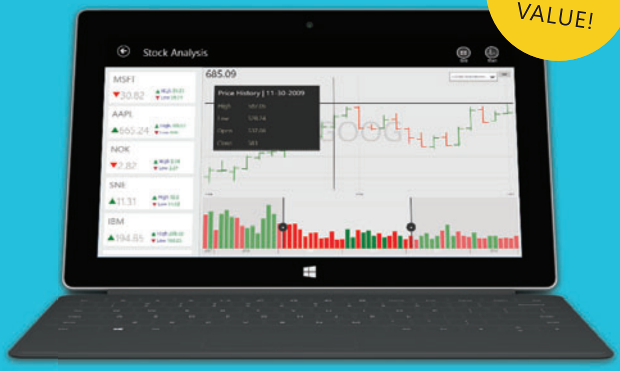
syncfusion.com/jshobbyist

WinRT

syncfusion.com/winrthobbyist

$599
VALUE!

$399
VALUE!



✓ 40+ controls: charts, grids, gauges, and more

✓ One year of support and updates

✓ Individual developers qualify

Buy online and get your license today!

Syncfusion®

FROM THE ORGANISERS OF **SOFTWARE ARCHITECT**

# DEVWEEK

17TH SUCCESSFUL YEAR!

31ST MARCH – 4TH APRIL 2014 | CENTRAL HALL, WESTMINSTER, LONDON

**THE DEFINITIVE DEVELOPER CONFERENCE**

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

**Database | UX | Architecture | Agile | MS Technologies | Mobile | Web**
**Programming Languages / Techniques | DevOps | Security | Cloud | Test | MQ | BI**

www.devweek.com

SPONSORS: ASPOSE Your File Format Experts | Logi Analytics | Oxford Instruments The Business of Science | sage

GOLD SPONSOR: K2

# WELCOME

**DevWeek is back! The UK's leading developer conference returns to London for its 17th successful year from Monday, 31st March, to Friday, 4th April 2014.**

Keep your skills up to date with the latest technologies, best practices and frameworks from industry-leading experts. For 2014, we have a brand-new agenda, including new topics, new speakers and more content than ever before:

- ◆ **20 x full-day workshops**
- ◆ **102 x 90min breakout sessions**
- ◆ **Two keynote presentations**
- ◆ **49 expert speakers (over half of them new for 2014!)**

**We look forward to seeing you there!**
**The DevWeek Team**

## AGENDA GUIDE

To help you navigate the wealth of content at DevWeek 2014, all breakout sessions and workshops within this brochure have been colour-coded as a reference to the top-level topic they are covering. Please refer to the agenda guide below.

A quick, at-a-glance agenda of the main conference days is available on pages 4-5. The full agenda for all five days, with presentation abstracts, can be found on pages 6-21.

- ◆ **Database**
- ◆ **UX**
- ◆ **Architecture**
- ◆ **Agile**
- ◆ **MS Technologies**
- ◆ **Mobile**
- ◆ **Web**
- ◆ **Programming Languages / Techniques**
- ◆ **DevOps**
- ◆ **Security**
- ◆ **Cloud**
- ◆ **Test**
- ◆ **MQ**
- ◆ **BI**

**DID YOU KNOW?**
**You can highlight your favourite sessions and save your own personalised agenda by using the interactive online agenda at devweek.com**

*These colour symbols appear under the descriptions of each presentation throughout the guide.*

**VENUE**
**CENTRAL HALL WESTMINSTER LONDON, SW1H 9NH**

## Venue

**For 2014, DevWeek has moved to brand-new venue: Central Hall in Westminster, London. The venue, right in the heart of London, is a fantastic facility that seems almost purpose-built for DevWeek!**
The iconic outline of Central Hall, built in 1912, proudly sits opposite Westminster Abbey. Offering easy access to public transport and the West End, the venue provides a visual impact and gives 'meeting in London' a new dimension. Constructed as one of the oldest 'purpose-built' conference centres in the UK, Central Hall uniquely fuses historical architecture 'built for events' with state-of-the-art event technology.

**Nearest Tube Stations:**
Westminster (District, Circle & Jubilee)
St James's Park (District & Circle)
For more information on Central Hall, please visit www.c-h-w.com

## Sponsorship & Exhibition

**Interested in becoming a sponsor or an exhibitor?**
DevWeek is the UK's leading event for software developers, DBAs and IT architects. If you provide services or technologies relevant to the DevWeek audience, get in touch today to discuss the wide range of sponsorship and exhibition opportunities. Please contact:
Chris Handsley
E: chris.handsley@publicis-blueprint.co.uk
T: +44 (0)7534 707063

## DevWeek 2014 is bigger and better than ever before

Here are just four reasons why you won't want to miss this year's conference:

### 01 SHARE YOUR TICKET!

Only have the time to attend one day? Need to share your ticket with others in your team?

There's nothing quite like experiencing the full conference yourself. But if you can't attend the whole event, you can share your ticket with others in your team and make the most out of the full five days of sessions and workshops. The online registration page allows you to fill in your colleagues' details so that we know who from your team will be joining us each day.

For more info, please visit devweek.com/booknow

### 02 DON'T MISS A SESSION!

Can't be in two places at once? Catch up with any vital breakout sessions you missed, online, after the event.

With up to 10 concurrent tracks, there will inevitably be times when you have to choose between two equally important breakout sessions. This year, you won't miss out! All breakout sessions (subject to speaker approval) will be filmed and provided online for all registered delegates.

### 03 WE HAVE A REVAMPED LINE-UP OF 49 INDUSTRY-LEADING SPEAKERS

For 2014, we're bringing you the best line-up of speakers DevWeek has ever seen. Of course, we still have lots of DevWeek favourites, but this year more than half of our speakers are joining us for the first time. Among the newcomers are industry-leading thinkers, innovators, pioneers and educators; bringing new expertise, new ideas and new perspectives to this year's talks and workshops. Our speakers have the inside track on everything you need to stay ahead of the competition.

For the full list of speakers, please refer to pages 22-23 or visit devweek.com/speakers

### 04 OUR BRAND-NEW AGENDA INCLUDES MORE SESSIONS AND TOPICS THAN EVER BEFORE

Our aim for DevWeek 2014 is to provide an agenda that covers the topics that are most relevant to software developers, DBAs and IT architects right now. That's why there are more presentations and workshops to choose from than ever before: 20 full-day workshops, 102 breakout sessions and two keynote presentations, to be precise!

The conference has been planned so that your whole team can benefit from attending, with everything from introductory sessions for junior staff to full-day workshops for comprehensive topic overviews. With the widest-possible variety of in-depth sessions, you can learn everything you and your team need to know to stay ahead of the game at DevWeek 2014.

For the complete agenda, please refer to pages 6-21 or visit devweek.com

**Iconic:** The exterior of Central Hall, Westminster, Central London's largest conference venue

## AT-A-GLANCE AGENDA

**THIS AT-A-GLANCE AGENDA** HIGHLIGHTS ALL OF THE MAIN CONFERENCE BREAKOUT SESSIONS, TAKING PLACE BETWEEN 1ST AND 3RD APRIL, ALLOWING YOU TO QUICKLY HIGHLIGHT THE KEY SESSIONS THAT YOU WANT TO ATTEND. *For pre- and post-conference workshops, please refer to pages 6 and 20*

| DAY 2: MAIN CONFERENCE | | | | DAY 3: MAIN CONFERENCE | |
|---|---|---|---|---|---|
| 9.30 | 11.30 | 14.00 | 16.00 | 9.30 | 11.30 |
| Sander Hoogendoorn<br>Roy Osherove<br>WELCOME ADDRESS AND KEYNOTE PRESENTATIONS:<br>DEATH BY DOGMA VERSUS ASSEMBLING AGILE<br>INTEGRITY-DRIVEN DEVELOPMENT | Robert Boedigheimer<br>CRYPTOGRAPHY 101 USING THE .NET FRAMEWORK ◆ | Allen Holub<br>OAUTH 2.0 ◆◆ | Bob Beauchemin<br>HOW TO SET UP SECURITY FOR YOUR DATABASE APPLICATION ◆ | Roy Osherove<br>REFACTORING SKILLS FOR TDD ◆<br>ALL-DAY WORKSHOP | |
| | Mark Murphy<br>OPTIONS FOR GETTING INTO ANDROID ◆ | Mark Smith<br>DEVELOPING CROSS-PLATFORM MOBILE APPLICATIONS WITH C# AND XAMARIN ◆ | Mike Taulty<br>WINDOWS 8 AND WINDOWS PHONE 8: BUILDING PORTABLE .NET CODE FOR BOTH PLATFORMS ◆◆ | Mark Murphy<br>TOP 10 ANDROID APP SECURITY STEPS ◆◆ | Kevlin Henney<br>TEST SMELLS AND FRAGRANCES ◆ |
| | Ido Flatow<br>IAAS VS. PAAS: WINDOWS AZURE COMPUTE STRATEGIES ◆◆ | Giles Davies<br>WELL, I LIKE TEAM FOUNDATION SERVER AND I LIKE GIT, BUT WHICH IS BETTER? ◆ | Austin Bingham<br>GERRIT & JENKINS: A DEV-OPS DUO FOR PRODUCTIVITY, EXPERIMENTATION & BETTER SOFTWARE ◆ | Mike Taulty<br>A LAP AROUND WINDOWS AZURE MOBILE SERVICES ◆ | Andrew Clymer<br>BRAND NEW WORLD OF ASYNCHRONOUS PROGRAMMING ◆◆ |
| | Brian Randell<br>HEAVEN OR HOLLYWOOD: IRON MAN NUI – IS IT GOOD UX OR JUST FUI? ◆ | Paul Ardeleanu<br>PROTOTYPING SAVES YOUR BACON ◆ | Jules May<br>'IF' AND 'GOTO' – THE EVIL TWINS: HOW TO ERADICATE 95% OF ALL YOUR BUGS IN ONE SIMPLE: STEP ◆ | Nuno Filipe Mendes Godinho<br>WHAT'S NEW IN WINDOWS AZURE ◆ | Shay Friedman<br>THE WONDERFUL WORLD THAT IS TWITTER BOOTSTRAP ◆ |
| | Dino Esposito<br>WHAT'S NEW IN ASP.NET MVC 5 ◆◆ | Kevlin Henney<br>INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS ◆ | Michael Kennedy<br>18 WAYS YOUR BRAND-NEW ASP.NET MVC PROJECT CAN BE BETTER ◆ | Brian Randell<br>HEAVEN OR HOLLYWOOD: BUILDING GREAT UX WITH NUI ◆ | Gil Fink<br>INTRODUCTION TO HTML5 ◆ |
| | Sahil Malik<br>I WANT TO DEVELOP FOR SHAREPOINT, BUT I DON'T KNOW WHERE TO START ◆ | Shay Friedman<br>ASP.NET MVC TIPS, TRICKS AND HIDDEN GEMS ◆ | Pearl Chen<br>ANGULAR JS AIN'T JUST ANOTHER MVC FRAMEWORK ◆◆ | Robert Boedigheimer<br>WEB PERFORMANCE – LIVE SITE REVIEWS! ◆ | Sebastien Lambla<br>HTTP CACHING ON .NET ◆ |
| | Allan Kelly<br>WHAT DO YOU GET IF YOU COMBINE XP AND KANPAN? GET READY FOR XANPAN! ◆◆ | Andrew Clymer<br>SIMPLIFYING THREAD-SAFE CODE WITH CONCURRENT DATA STRUCTURES ◆ | Ben Lambert<br>PRACTICE PROGRAMMING – WHAT'S THE POINT? ◆ | Sander Hoogendoorn<br>HOW FRAMEWORKS CAN KILL YOUR PROJECTS AND PATTERNS TO AVOID BEING KILLED ◆ | Yaniv Rodenski<br>Ido Flatow<br>BATTLE OF THE FRAMEWORKS: ASP.NET VS. NODE.JS ◆ |
| | Dan Clark<br>CREATING HADOOP MAP-REDUCE JOBS IN C# ◆◆ | Dejan Sarka<br>COLUMNSTORE INDEXES AND BATCH PROCESSING IN SQL SERVER 2012 AND 2014 ◆ | Jim Webber<br>A LITTLE GRAPH FOR THE BUSY DEVELOPER ◆ | Sasha Goldshtein<br>WHAT'S NEW IN C++ 11? ◆ | Giles Davies<br>WHAT'S NEW IN VISUAL STUDIO 2013 ◆◆ |
| | Bob Beauchemin<br>MEMORY-OPTIMISED TABLES AND COMPILED STORED PROCEDURES (AKA HEKATON) ◆ | Matt Milner<br>WHAT'S NEW IN BIZTALK SERVER 2013 ◆ | Klaus Aschenbrenner<br>THE DANGEROUS BEAUTY OF BOOKMARK LOOKUPS ◆ | Allen Holub<br>MESSAGING WITH RABBITMQ ◆ | Dan Clark<br>USING HIVE TO QUERY AND PROCESS BIG DATA ◆ |
| | Richard Blewett<br>AN INTRODUCTION TO THE REACTIVE FRAMEWORK ◆ | Neal Ford<br>FUNCTIONAL THINKING ◆ | Iordanis Giannakakis<br>DEVICE FRAGMENTATION VS. CLEAN CODE ◆ | Dejan Sarka<br>EXCEL 2013 AND OFFICE 365 POWER BI ◆◆◆ | Bob Beauchemin<br>NEW CARDINALITY ESTIMATING FOR QUERY PLANS ◆ |

**BOOK NOW**
BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

*Please refer to page 2 for guide to colour symbols*

## AT-A-GLANCE AGENDA

ALSO EACH DAY... 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

|  |  | DAY 4: MAIN CONFERENCE |  |  |  |
|---|---|---|---|---|---|
| 14.00 | 16.00 | 9.30 | 11.30 | 14.00 | 16.00 |
|  |  | Neal Ford<br>CONTINUOUS DELIVERY WORKSHOP ◆<br>ALL-DAY WORKSHOP |  |  |  |
| Ralf Westphal<br>MAKE REFACTORING UNAVOIDABLE – TDD AS IF YOU MEANT IT ◆ | Dino Esposito<br>A FIRST LOOK AT ASP.NET IDENTITY ◆◆ | Sahil Malik<br>CUSTOM SECURITY TRIMMING FOR SEARCH IN SHAREPOINT 2013 ◆◆ | Seb Rose<br>SO LONG, AND THANKS FOR ALL THE TESTS ◆ | Sander Hoogendoorn<br>CROSSING THE CHASM FROM WEB TO WINDOWS 8 DEVELOPMENT (AND SOME ANDROID TOO) ◆◆ | Seb Rose<br>MUTATION TESTING – WHY YOU SHOULD CARE ◆ |
| Paul Ardeleanu<br>IOS DEVELOPER OVERVIEW ◆ | Brian Randell<br>CROSS-PLATFORM DEV (IOS, ANDROID AND JAVA) WITH TFS AND TEAM EXPLORER EVERYWHERE ◆ | Ralf Westphal<br>INCREASE CHANGEABILITY – OOP AS IF YOU MEANT IT ◆ | Roy Osherove<br>MOCKING AND ISOLATION FRAMEWORKS DEEP DIVE IN .NET ◆ | Sasha Goldshtein<br>FIRST STEPS IN IOS APPLICATION DEVELOPMENT ◆ | Simon Robinson<br>INSIDE WORD, EXCEL AND POWERPOINT DOCUMENTS: A DEVELOPER'S PERSPECTIVE ◆ |
| Allen Holub<br>LIVING ON CLOUD 1001: A COMPARATIVE LOOK AT AMAZON AND GOOGLE CLOUD PLATFORMS ◆ | Bob Beauchemin<br>SQL SERVER CLOUD CHOICES – USE WINDOWS AZURE SQL DATABASE OR SQL SERVER IN A VM? ◆◆ | Nuno Filipe Mendes Godinho<br>LESSONS LEARNED ON HOW TO SELECT YOUR CLOUD COMPUTING VENDOR ◆ | Austin Bingham<br>A CULTURE OF REVIEW ◆ | Nuno Filipe Mendes Godinho<br>REAL-TIME DATA MANAGEMENT IN THE CLOUD ◆ | Shay Friedman<br>MIGRATING APPLICATIONS TO WINDOWS AZURE ◆ |
| Pavel Skribtsov<br>COMPUTING LIKE THE BRAIN: INTRODUCTORY GUIDE TO AI ◆ | David Britch<br>BUILDING NATIVE WINDOWS STORE APPS FOR C# DEVELOPERS ◆◆ | Richard Blewett<br>INTRODUCING UNIT TESTING IN LEGACY CODE ◆ | Sahil Malik<br>SHAREPOINT APPS: AN OVERVIEW ◆ | Simon Robinson<br>.NET COLLECTIONS ◆ | David Britch<br>ACCELERATING WINDOWS STORE APP DEVELOPMENT USING PRISM FOR THE WINDOWS RUNTIME ◆ |
| Robert Boedigheimer<br>MOBILE WEB SITES ◆◆ | Ido Flatow<br>IIS FOR DEVELOPERS ◆◆ | Gil Fink<br>CREATING DATA-DRIVEN HTML5 APPLICATIONS ◆ | Ido Flatow<br>ALREADY FAMILIAR WITH ASP.NET WEB API? YOU SURE? ◆ | Sebastien Lambla<br>REST ON .NET ◆◆ | Yaniv Rodenski<br>SIGNALRITY ◆ |
| Udi Dahan<br>PROGRAMMING IN THE FOURTH DIMENSION ◆ | Pearl Chen<br>BETA.TELUS.COM: RESPONSIVE, ITERATIVE, COLLABORATIVE ◆◆ | Allen Holub<br>KNOCKOUT: AN INTRODUCTION ◆ | Brian Randell<br>MAKING THE MOST OF THE TFS SERVICE ◆◆ | Sahil Malik<br>SHAREPOINT CSOM AND REST IN THE REAL WORLD ◆◆ | Andrew Clymer<br>PATTERNS IN PARALLEL PROGRAMMING WITH TPL ◆◆ |
| Sasha Goldshtein<br>TASK AND DATA PARALLELISM: REAL-WORLD EXAMPLES ◆ | Ed Courtenay<br>WHAT ARE AUTO-MOCKING CONTAINERS, AND WHY SHOULD YOU USE THEM? ◆ | Mike Taulty<br>WINDOWS 8.1 APPS: WHAT'S NEW FOR APP DEVELOPERS? ◆ | Kevlin Henney<br>SEVEN INEFFECTIVE CODING HABITS OF MANY PROGRAMMERS ◆ | Ed Courtenay<br>GROWING CONFIGURABLE SOFTWARE WITH EXPRESSION<T> ◆◆ | Robert Smallshire<br>DELIVER DOMAIN-DRIVEN DESIGNS – DYNAMICALLY! ◆ |
| Michael Kennedy<br>APPLIED NOSQL IN .NET ◆ | Dejan Sarka<br>SQL SERVER TRANSACTION ISOLATION LEVELS ◆ | Dejan Sarka<br>THE ART (AND ARIMA) OF FORECASTING ◆ | Michael Kennedy<br>HIGH-PERFORMANCE NOSQL TECHNIQUES ◆ | Dejan Sarka<br>OPTIMISING TEMPORAL QUERIES (PART 1): AN INTRODUCTION ◆ | Dejan Sarka<br>OPTIMISING TEMPORAL QUERIES (PART 2): THE OPTIMISATION ◆ |
| Klaus Aschenbrenner<br>UNIQUEIDENTIFIERS AS CLUSTERED KEYS IN SQL SERVER ◆ | Matt Milner<br>DID YOU GET MY MESSAGE? BROKERED MESSAGING ON WINDOWS AND BEYOND ◆ | Bob Beauchemin<br>WORK WITH SPATIAL DATA IN SQL SERVER ◆ | Udi Dahan<br>LOOSELY-COUPLED ORCHESTRATION WITH MESSAGING ◆ | David Evans<br>SPECIFICATION BY EXAMPLE ◆ | Bob Beauchemin<br>WHAT'S IN MICROSOFT'S HADOOP OFFERINGS? ◆ |

## DAY 1 AGENDA: ALL-DAY PRE-CONFERENCE WORKSHOPS

Monday 31st March

THE FOLLOWING WORKSHOPS RUN FOR A **FULL DAY, FROM 09.30 TO 17.30** WITH A SHORT BREAK IN THE MORNING AND AFTERNOON, AND A LUNCH BREAK AT 13.00.

UNLESS OTHERWISE NOTED IN THE DESCRIPTION, THEY ARE PRESENTATION-BASED IN STYLE RATHER THAN 'HANDS-ON' LABS.

*Please refer to page 2 for guide to colour symbols.*

### WHAT'S NEW IN SQL SERVER 2014 FOR DEVELOPERS

**Bob Beauchemin**

**WORKSHOP REF: M1**

**SQL Server 2014 is just around the corner and, in this workshop, Beauchemin will discuss in detail the most important and compelling new features.** This includes memory-optimised tables and compiled storage procedures (aka Hekaton), a new in-memory storage engine for greater speed in processing. The more interesting and useful implementation detail is that this is all integrated with native SQL Server – no new DDL and DML to learn, and integration between memory-optimisation and "traditional" tables is built-in.

He will also cover clustered columnstore indexes and other columnstore enhancements, which don't require extra copies of your data to use the in-memory, column-based xVelocity engine, and new cardinality estimating for query plans, which have been vastly overhauled based on a decade or more of real-world experience with the most difficult query patterns.

While these three changes alone are enough to fill the whole day's workshop, there are many more changes coming in CTP2 and beyond. Depending on public availability of the relevant information, Beauchemin will also discuss and demonstrate these.

🔶

### HOW TO DESIGN STUFF THAT MATTERS, FAST!

**Eewei Chen**

**WORKSHOP REF: M2**

Too much time is wasted creating that big design upfront, only to find that users don't like what you have built once it has been released. Today, we are in danger of not only over-designing, but also designing solutions to the wrong problems.

**In this workshop, Chen will demonstrate how to experiment with rapid design techniques to ensure design solutions for the right business problems are delivered to the right target audiences rapidly and continuously.**

He will show how to create design solutions fast, as a team, and how to work with a client to get products that really matter out into market early. Good design involves elegantly solving problems despite the constraints. More often than not, time is one of those constraints.

In this workshop, you will be forced to think and act quickly, exploring how to quickly work as a team to address a real-world problem; rapidly analysing customer, industry and business trends, behaviours and needs to validate ideas; and learning how to apply the latest design thinking, Lean Startup, Lean UX and agile methodologies, to bring your prototype to life and ensure it is as useful as it can possibly be.

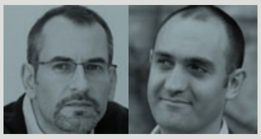
🔶

### BUILDING GRAPH DATABASE APPLICATIONS

**Ian Robinson**

**Jim Webber**

**WORKSHOP REF: M3**



Neo4j is a JVM-based graph database. For highly connected, semi-structured data, graph databases are thousands of times faster than relational databases, making them ideal for managing complex data across many domains, from finance to social, telecoms to geospatial.

**In this workshop, Robinson and Webber will cover the core functionality of the Neo4j graph database. With a mixture of architecture and hands-on coding sessions, you'll quickly learn how to develop a Neo4j-backed application.** Each session comprises a set of practical exercises designed to introduce and reinforce an aspect of the Neo4j stack.

Attendees won't need any previous experience of graph databases to participate. They will, however, need some experience of Java, and a laptop with a Java IDE of their choice. Attendees will leave with a good grounding in developing a graph database solution, a copy of O'Reilly's Graph Databases book, and a heap of additional exercises to help them hone their skills further.

🔶

### PROBLEM SPACE ANALYSIS: A new analytical technique to deliver robust, easily implementable and change-tolerant architectures

**Jules May**

**WORKSHOP REF: M4**

How do you design a large system? We know Waterfall doesn't work very well, but also that Agile scales poorly. Various proposals have been made (BDUF, domain-driven design, prototyping) but none has yet achieved much traction.

The key to managing a large system is – precisely – managing change. No specification ever survives its own implementation: as a system takes shape, everyone –developers, architects and stakeholders – change their minds continually. In any non-trivial project, goalposts are constantly in motion. A robust architecture is one that can anticipate those changes, and a good design is one that can accommodate them cheaply and efficiently.

Problem space analysis is a technique that simply and clearly anticipates, documents and defines the changes that can affect a project. It informs the architectural design so that it can accommodate those changes, and it delivers a change-tolerant ubiquitous language to unify and coordinate the development effort.

**In this workshop, May will introduce the principles of problem space analysis, and will show how those principles can be translated into architectures and thence into working systems, even while the goalposts are moving. The technique will be actualised using a real-life design problem.**

◆

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

## AGILE SOFTWARE ARHICTECTURE SKETCHES AND NoUML

**Simon Brown**

**WORKSHOP REF: M5**

Agility is about moving fast and this requires good communication. A consistent, shared vision is essential in order for teams to push in the same direction, but it's surprising that many agile teams struggle to effectively communicate the architecture of the software they are building.

As an industry, we do have the Unified Modeling Language (UML), yet many people favour informal "boxes and lines" sketches instead. The problem is that such diagrams rarely make any sense, usually need a narrative to accompany them and ultimately slow the team down.

Although we can argue whether UML offers an effective way to communicate software architecture, that's often irrelevant because many teams have already thrown out UML or simply don't know it. Abandoning UML is one thing but, in the race for agility, many software development teams have lost the ability to communicate visually.

**This hands-on workshop is aimed at everybody involved in the software development process and is about improving architectural communication. Brown will demonstrate some patterns and anti-patterns related to NoUML diagrams, and you'll learn some simple techniques for communicating software architecture using informal sketches.**

## SHAREPOINT 2013 DEVELOPERS' OVERVIEW

**Sahil Malik**

**WORKSHOP REF: M6**

SharePoint 2013 – I bet you've heard of it. Yeah, it's that big, complex platform that seems to be quite in demand. **This workshop includes everything you need to get rolling in the SharePoint world, making it the perfect introduction to SharePoint for a .NET developer.**

In this workshop, Malik covers a basic introduction, and touches almost every important topic in SharePoint. And don't worry, you need no prior SharePoint experience or knowledge to attend. The intended audience for this session is .NET developers who are interested in learning more about SharePoint development.

## IOS CRASH COURSE

**Paul Ardeleanu**

**WORKSHOP REF: M7**

It is without doubt that iOS is one of the most popular mobile operating systems out there. And with version 7, it moves on to a whole new level.

**In this workshop, join Ardeleanu for a full day of iOS geekness, where you'll learn how to plan, build, debug and release iOS 7 apps. This workshop is mainly for programmers who want to dabble into iOS and learn enough to "become dangerous".**

Topics will include: a gentle introduction to Objective-C; exploring Xcode 5 (IDE, Simulator, Instruments); iOS 7's new UI; how to migrate from iOS 6 to iOS 7; using storyboards to build user interfaces fast; design patterns (Model View Controller, Singleton, Delegation, Observer, Decorator); data persistence using plists, sqlite and Core Data; web services (inc. RESTful APIs); in-app purchases using the Store Kit framework; new frameworks (Sprite Kit, JavaScript Core); storing user data security with the iOS keychain; debugging and optimisation; and App Store and Ad-hoc distribution.

## NODE.JS FOR DEVELOPERS

**Yaniv Rodenski**

**WORKSHOP REF: M8**

It seems as if the world has gone mad. For years, we have been straggling JavaScript on the client side, and now, all of a sudden, people can't get enough of it on the server side. They claim it is "fun".

This is not a fringe phenomenon; Node.js is being used by giants such as eBay, LinkedIn and Microsoft.

**In this workshop, Rodenski will show how to embrace the zen of Node.js.**

## EXCEL 2013 – SELF-SERVICE BI AND DATA QUALITY

**Dejan Sarka**

**WORKSHOP REF: M9**

You have probably already heard for the term "self-service business intelligence (BI)". In Microsoft products, Excel is the tool for doing your own advanced analysis. It is powerful out of the box; however, the real power comes from free add-ons, providing help with data preparation and data quality.

**In this workshop, Sarka will show how to use Excel efficiently for analysis. You will learn how to use it as a client tool for Analysis Services; and how to retrieve data from a relational source, such as SQL Server, or from a big data source, such as HDInsight server.**

Sarka will introduce the full Power BI suite: Power Pivot, Power View, Power Map and Power Query. In addition, he will show how to use data-mining add-ons for the most advanced analysis.

The Master Data Services (MDS) add-on converts Excel into a powerful MDS client. And you can also use the Data Quality Services (DQS) matching algorithms with this add-on. What's more, with the Fuzzy Lookup add-on, you can do approximate matching of huge amounts of data, with much better accuracy and performance than with the DQS matching.

Today, we are in danger of not only over-designing but also of designing solutions to the wrong problems.

EEWEI CHEN
HOW TO DESIGN STUFF THAT MATTERS, FAST!

**DAY 2 AGENDA: MAIN CONFERENCE STREAMED SESSIONS**

THE FOLLOWING PAGES PROVIDE THE FULL ABSTRACTS FOR ALL THE MAIN BREAKOUT SESSIONS BETWEEN 1ST AND 3RD APRIL. THE MAIN CONFERENCE WILL BEGIN ON TUESDAY, 1ST APRIL, WITH A WELCOME ADDRESS AND TWO KEYNOTE PRESENTATIONS FOR ALL ATTENDEES.

*Please refer to page 2 for guide to colour symbols.*

**Sander Hoogendoorn**

**DEATH BY DOGMA VERSUS ASSEMBLING AGILE**

**Roy Osherove**

**INTEGRITY-DRIVEN DEVELOPMENT**

## WELCOME ADDRESS & KEYNOTE PRESENTATIONS

### 9.30am: Death by dogma versus assembling agile

Almost all organisations, large and small, are turning towards agile to escape failing traditional software development projects. Due to the strong increase in popularity of agile approaches and techniques, many newcomers enter the field of agile coaching; a lot of them without the necessary real-life experience but proudly waving their agile certificates, proving they at least had two days of training.

During this challenging talk, appreciated international speaker Sander Hoogendoorn, global agile thought-leader at Capgemini, shows what happens to organisations and projects that are coached by well-meaning consultants with little experience. Often this leads to very dogmatic applications of the more popular agile approaches, mostly Scrum and Kanban. This dogmatic thinking currently blocks the use of more elaborate techniques, tools and technology in agile projects, even when

**Robert Boedigheimer**

**CRYPTOGRAPHY 101 USING THE .NET FRAMEWORK**

In the world of PCI compliance and other government and corporate security regulations, how does a web developer deal with the very real security threats to their web site? In this session, Boedigheimer teaches the "black art" of cryptography, including public/private and symmetric encryption, hashing, digital signatures and a dash of salt. He will review the basics of cryptography and what techniques are appropriate for various situations. Discover practical techniques for securing content received on public web sites. Review .NET classes to use for cryptography, how ASP.NET uses cryptography, and how to protect sections of the web.config file.

◆

**Mark Murphy**

**OPTIONS FOR GETTING INTO ANDROID**

If you want to hop on the Android bandwagon, this session will help you get started! Here, you will learn about the basics of traditional Android application development using Java, how most Android apps are created. In addition, you will be exposed to a plethora – nay, a veritable cornucopia! – of alternative ways of creating Android apps, from HTML5 and hybrid apps, to 3D gaming engines and app generators. You will leave with a good idea of the technologies that best fit your skill set, as a starting point for further exploration.

◆

**Ido Flatow**

**IAAS VS. PAAS: WINDOWS AZURE COMPUTE STRATEGIES**

A few years ago, working with Windows Azure was simple. You want a background service? Use a "Worker" role. You want a web application? Use a "Web" role. Today, with the addition of web sites and virtual machines, the decision of what to deploy and how to deploy it just got harder. In this session, Flatow will explore the various hosting options offered by the Windows Azure platform, examine the steps required to deploy to each environment, and discuss the advantages and disadvantages of each solution.

◆ ◆

**Brian Randell**

**HEAVEN OR HOLLYWOOD: IRON MAN NUI – IS IT GOOD UX OR JUST FUI?**

Beautiful, rich, and even sexy: the user interfaces and experiences shown on cinema screens over the years keep getting better and better. From the original tricorders and PADDs in Star Trek, to the hand-waving, holographic computer interface used by Iron Man in his lab, Hollywood has shown us what could be. The question is: are these dream user experiences really heaven for users or are they just Hollywood props? In this session, Randell will take you through a variety of Hollywood user experiences and break them down into the primitives you should and should not use in your own application.

◆

**Dino Esposito**

**WHAT'S NEW IN ASP.NET MVC 5**

Freshly released, ASP.NET MVC 5 is the latest version of the popular Microsoft framework for more sustainable web development. This session assumes some familiarity with the framework and focuses on the delta from previous versions. This includes such areas as authentication, membership, routing, HTML templates, action filters and Web API.

◆ ◆

08.30: REGISTRATION AND COFFEE

9.30

these would really improve the work. Due to this lack of experience and the growing dogmatism in the agile beliefs, more and more agile projects will fail.

But perhaps even more importantly, during this talk, Hoogendoorn will also show that there is no such thing as one-size-fits-all agile. Different organisations and different projects require different agile approaches. Sometimes lightweight agile, user stories, simple planning and estimation are just fine. But in many projects, the way of working used should rather be built up from slightly more enterprise-ready approaches.

During this talk, Hoogendoorn will demonstrate how to assemble an agile approach that is specifically suited to your project, with many examples from real-life agile implementations.

**10.15am: Integrity-driven development**
What does it mean to develop with integrity? Is it just a wishy-washy subject or can we translate it into actual day-to-day practices? The way we promise things; the way we act when we know we can't promise something; how we give out estimates. Do we face reality or do we tell people what they want to hear? Does code review improve integrity?

In this session, Osherove will explain why integrity can provide a welcome change to developers who feel helpless in their jobs, doing the wrong thing, and knowing what the right thing would have been.

He will use examples from day-to-day work life, where you can choose to use integrity or avoid it, and what are the possible consequences.

11.00: COFFEE BREAK

11.30

**Sahil Malik**

**I WANT TO DEVELOP FOR SHAREPOINT, BUT I DON'T KNOW WHERE TO START**
If you find SharePoint is interesting, but don't know where to start, then this session is for you. Malik will walk you through the fundamental building blocks of how to move from zero to hero. Specific topics will include core programming skills, investing in the right areas to optimise your learning curve, free resources and tools to help you get started, tips and tricks to get you on your way, and plenty more besides. If you're looking to get started, but need some help on your journey then don't miss this session!
◆

**Allan Kelly**

**WHAT DO YOU GET IF YOU COMBINE XP AND KANPAN? GET READY FOR XANPAN!**
The world doesn't need another software development method, there are plenty already! But each team needs to learn to create its own. So what do you get if you cross Kanban with Extreme Programming? Xanpan! In this session, Kelly will describe what you get if you mix these two elements, throw in a bit more Lean, season with Economics and stir. The resulting Xanpan focuses on teams not projects, and allows for planned and unplanned work within iterations and levels flow. One team using this approach claims to be able to deliver "to the day".
◆◆

**Dan Clark**

**CREATING HADOOP MAP-REDUCE JOBS IN C#**
All data processing in Hadoop essentially boils down to a map-reduce process. The mapping consists of retrieving the data and performing operations, such as filtering and sorting. The reduce part of the process involves a summary operation, such as grouping and counting. Hadoop map-reduce jobs are often written in Java, which can present a steep learning curve to non-Java programmers. However, the Microsoft SDK for Hadoop provides a convenient API wrapper around the Hadoop Streaming framework. Using this API, you can create custom map-reduce jobs in C#. In this session, Clark will explore how to do this, and provide several concrete examples for learning the process.
◆◆

**Bob Beauchemin**

**MEMORY-OPTIMISED TABLES AND COMPILED STORED PROCEDURES (AKA HEKATON)**
SQL Server 2014 introduces a new in-memory storage engine for greater speed in processing. In addition, rather than reproduce the page-based structures of the original storage engine, this engine operates on linked lists of hash buckets – all structures are lock and latch-free. Multi-version storage and optimistic concurrency are used to get the most out of the new engine. The most interesting and useful implementation detail is that this is all integrated with native SQL Server – no new DDL and DML to learn, and integration between memory-optimised and "traditional" tables is built-in.
◆

**Richard Blewett**

**AN INTRODUCTION TO THE REACTIVE FRAMEWORK**
The Reactive Framework is a library that is based around the IObservable interface and LINQ. It introduces a compelling programming model that allows you to build "event"-based code with declarative LINQ statements. In this session, Blewett will introduce the Reactive Framework and show how it can greatly simplify your code.
◆

13.00: LUNCH BREAK

## DAY 2 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

**Allen Holub**

### OAUTH 2.0

The OAuth 2.0 authentication protocol is used by many web services (eg Google Apps) to allow third-party access without exposing your password. For example, OAuth can permit a mobile phone app to access your Google calendar without giving your Google password to that app. In this session, Holub will look at how the OAuth 2 protocol works, at how it grants security, and at its vulnerabilities. He will look in depth at code examples (in Java) that use Google's OAuth high-level APIs, and also examine a low-level implementation of the protocol that uses the underlying REST API (and so can interface with any vendor).

🟥 🟦

**Mark Smith**

### DEVELOPING CROSS-PLATFORM MOBILE APPLICATIONS WITH C# AND XAMARIN

In this session, Smith will demonstrate how to build applications that target iOS, Android and Windows Phone, while reducing the coding required to build native apps! He will show you how to leverage C# and the Mono/Xamarin platform to share code across the most common mobile platforms through case study and real-world examples. Learn how to save time and money by writing the majority of the application once and then re-using that logic for each of your target platforms.

🟨

**Giles Davies**

### WELL, I LIKE TEAM FOUNDATION SERVER AND I LIKE GIT, BUT WHICH IS BETTER?

Learn how Team Foundation Server now allows you to choose between Team Foundation Version Control and Git, while retaining integration into the agile tools, work items, builds, test case management and more. In this session, Davies will demonstrate setting up and using Git within Team Foundation Server and Visual Studio, showing how you can use the built-in Git integration alongside your favourite command line and other tools. No prior knowledge of either Team Foundation Server or Git is required for this session.

◆

**Paul Ardeleanu**

### PROTOTYPING SAVES YOUR BACON

Prototyping is often a misunderstood subject, especially when it comes to mobile apps. It is often mistaken for wire-framing or detailed project specifications. In this session, Ardeleanu will explore the tools and techniques available to create an agile environment where the client can participate in the process. He will take an app from the idea stage and progress it through the list of features, writing the Application Definition Statement (ADS), sketching, paper prototyping and eventually on to something that can run on the actual device. Clients love that! And it could save your bacon.

🟨

**Kevlin Henney**

### INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

Although it is a simple agile value, the idea that individuals and interactions are more significant than processes and tools is often overlooked. Of course, processes and tools make a difference – sometimes a very big difference – but what determines whether a process or tool is effective is related to the individuals and interactions. To best achieve agility you need to start with the current context and understand how people actually behave in response to their environment, their beliefs and one another. Does making "business value" the centrepiece of what they do motivate those who produce it? Or is it more about the individuals and interactions?

🟩

**Bob Beauchemin**

### HOW TO SET UP SECURITY FOR YOUR DATABASE APPLICATION

In this session, Beauchemin will cover best practices for SQL Server security from an application perspective, including how to most effectively and easily set up logins, users and group users. Principal of least privilege is the model used here, with special attention paid to auditing and contained databases.

🟥

**Mike Taulty**

### WINDOWS 8 AND WINDOWS PHONE 8: BUILDING PORTABLE .NET CODE FOR BOTH PLATFORMS

For a .NET developer building device apps on Windows and Windows Phone, there's a simple dream – write code once and run it on all Windows devices. That day hasn't quite arrived, but there's still a lot you can do with design patterns such as MVVM and techniques such as Portable Class Libraries to share code across Windows/Phone. In this session, Taulty will take a look at these approaches by taking an app from "no shared code" through to "lots of shared code", and illustrating the possibilities for sharing along the way.

🟥 🟨

**Austin Bingham**

### GERRIT & JENKINS: A DEV-OPS DUO FOR PRODUCTIVITY, EXPERIMENTATION & BETTER SOFTWARE

The Gerrit code review system and the Jenkins continuous integration server form a powerful open-source combination, supporting rapid development while maintaining high quality through peer review and automated testing. Gerrit's code reviews help ensure that your quality standards are maintained for all commits, while Jenkins' flexible automation framework adds testing to your commit cycle. In this session, Bingham will briefly cover installation of the tools before quickly moving on to explaining and demonstrating the features and workflows that make the Gerrit & Jenkins combination so attractive.

◆

**Jules May**

### 'IF' AND 'GOTO' – THE EVIL TWINS: HOW TO ERADICATE 95% OF ALL YOUR BUGS IN ONE SIMPLE STEP

In 1968, CACM published a letter from Edgar Dijkstra, called "Go To statement considered harmful". In it, he explained exactly why most bugs in programs were caused by Gotos, and he appealed for Goto to be expunged from programming languages. But, Goto has a twin brother, which is responsible for nearly every bug that appears in our programs today. That twin is "If". In this session, May will revisit Dijkstra's original explanation, and show why If and Goto have the same pathology. He will then go on to explain how to avoid this pathology altogether, and construct programs that are orders of magnitude more reliable than what we have come to expect.

🟫

**Michael Kennedy**

### 18 WAYS YOUR BRAND-NEW ASP.NET MVC PROJECT CAN BE BETTER

So you're ready to start that new and ambitious ASP.NET MVC project. Maybe you're kicking off a new start-up or just finally moving that old-and-crusty webforms project into the modern development world. Either way, this session will give you some easy things you can do immediately after creating your new MVC project that you will thank yourself for as your project grows in complexity.

🟦

## DAY 2 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

13.00: LUNCH BREAK

14.00

**Shay Friedman**

### ASP.NET MVC TIPS, TRICKS AND HIDDEN GEMS

The ASP.NET MVC framework has been around for quite a while now, and it has been constantly gaining popularity ever since it appeared on the scene. However, despite that fact, a lot of MVC developers are not aware of various hidden gems that can make their development experience much easier and nicer. In this session, Friedman will go through some of those secrets, helping to ease and quicken your everyday work with the ASP.NET MVC framework.

◆

**Andrew Clymer**

### SIMPLIFYING THREAD-SAFE CODE WITH CONCURRENT DATA STRUCTURES

When developing multi-threaded applications that share state, we need to consider thread safety when sharing state across multiple threads. These techniques require the developer to understand the possible race conditions and select the cheapest synchronisation technique to satisfy thread safety. But, while essential, they can often become tedious and make the simplest of algorithms seemingly overly complicated and hard to maintain. In this session, Clymer will explore the use of built-in concurrent data structures shipped with TPL that will simplify multi-threaded code while maximising concurrency and efficiency.

◆

**Dejan Sarka**

### COLUMNSTORE INDEXES AND BATCH PROCESSING IN SQL SERVER 2012 AND 2014

With large data warehouses (DW), it is essential to have fast queries. Before columnstore indexes and batch processing it was nearly impossible to achieve reasonable performance of DW queries on large tables without investing huge amounts of money in parallelism, such as the Parallel Data Warehouse. With SQL Server 2012 and 2014, you can achieve satisfactory performance even with a traditional single-box database server. In this session, Sarka goes beyond a simple introduction of the columnstore indexes and batch processing, to include an in-depth explanation and examples on how to get the most out of these new features.

◆

**Matt Milner**

### WHAT'S NEW IN BIZTALK SERVER 2013

Learn about the new features in BizTalk Server 2013 including new adapters, Windows Azure support, improvements to runtime and administrative features, as well as ESB Toolkit integration and what that really means. BizTalk Server 2013 is an evolution of Microsoft's popular integration messaging product that embraces the cloud and the latest technologies such as SFTP, REST and Service Bus. In this session, Milner will demonstrate how to use the new REST adapter to consume a public service, connecting to a Service Bus Queue using the adapter to send messages and another to receive messages, provisioning a BizTalk 2013 server in Windows Azure IaaS, and much more.

◆

**Neal Ford**

### FUNCTIONAL THINKING

Learning the syntax of a new language is easy, but learning to think under a different paradigm is hard. In this session, Ford will help you transition from a Java-writing imperative programmer to a functional programmer, using Java, Clojure and Scala as examples. He will take common topics from imperative and OOP languages and look at alternative ways of solving those problems in functional languages. Expect your mind to be bent, but you'll leave with a much better understanding of both the syntax and semantics of functional languages.

◆

15.30: COFFEE BREAK

16.00

**Pearl Chen**

### ANGULAR JS AIN'T JUST ANOTHER MVC FRAMEWORK

You've probably heard of Backbone.js as a JavaScript MVC (Model-View-Controller) framework for building web apps, but it's worth exploring other options when you want to get more out of your framework. AngularJS is highly opinionated, which makes it easier for beginners to grasp core MVC concepts, while veteran web developers will enjoy its integration with automated testing suites. And for anyone who wants to write less boilerplate code? You will love Angular JS's two-way data binding abilities. Attend this session and Angular JS just might turn into your favourite MVC framework.

◆◆

**Ben Lambert**

### PRACTICE PROGRAMMING – WHAT'S THE POINT?

Practice Programming is a simple concept that can help any developer stay at the top of their game, learning better ways to do things. It can be as simple as the old favourite "Fizz-Buzz" program, to writing games in downtime. In this session, Lambert will explore the idea of Practice Programming, and how it can help you and the people around you in a relaxed, and probably irreverent, discussion. He will look at the ideas behind Practice Programming, why too much practice can be a bad idea (practice makes imperfect) and why practicing "bad programming" can actually be helpful too. There will also be examples that you can use for your own Practice Programming (both for good and bad).

◆

**Jim Webber**

### A LITTLE GRAPH FOR THE BUSY DEVELOPER

In this session, Webber will explore powerful analytic techniques for graph data, beginning with some of the innate properties of (social) graphs from fields such as anthropology and sociology. By understanding the forces and tensions within the graph structure and applying some graph theory, it is possible to predict how the graph will evolve over time. To test just how powerful and accurate graph theory is, he'll (retrospectively) predict World War 1 based on a social graph and a simple algorithm. These powerful techniques can also be applied to modelling domains in Neo4j (a graph database). Don't worry, there won't be much maths.

◆

**Klaus Aschenbrenner**

### THE DANGEROUS BEAUTY OF BOOKMARK LOOKUPS

SQL Server Bookmark Lookups can be very powerful, when used correctly. However, there are a lot of side effects that you should be aware of when your indexing strategy relies on them. In this session, Aschenbrenner will dig into the various problems that Bookmark Lookups can cause, how they occur and how you can resolve them to get better performance from your SQL Server. Specific topics will include the tipping point, parameter sniffing problems, bad statistics, auto/forced parameterisation and Bookmark Lookup deadlocks.

◆

**Iordanis Giannakakis**

### DEVICE FRAGMENTATION VS. CLEAN CODE

Trying to implement the same functionality on different devices, OS versions, manufacturers and so on can be more trouble than you'd expect. In this session, Giannakakis will show you how you can get the job done without compromising code quality and readability. Specifically, he will demonstrate how to use dependency injection, model-view-presenter pattern and other techniques to achieve the task at hand.

◆

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

17.30: DRINKS RECEPTION

## DAY 3 AGENDA: MAIN CONFERENCE STREAMED SESSIONS

**Mark Murphy**

### TOP 10 ANDROID APP SECURITY STEPS

This session will focus on defending your users against the forces of evil, no matter what you define as "the forces of evil". Here, you will learn about 10 security measures that Android developers can readily incorporate into their apps to protect their users' data at rest (files and databases) and in motion (transmission over the Internet). Attendees will depart with a checklist of what needs to be done to their apps, and instructions on how to do it (including copious sample code), to show the [insert evildoer here] who's boss.

◆◆

**Mike Taulty**

### A LAP AROUND WINDOWS AZURE MOBILE SERVICES

Mobile app developers often need a cloud back-end to power their apps. At a minimum, that back-end would need to provide data services but it's typical for a mobile app to need other services, such as authentication, authorisation and services to deliver push notifications down to devices. Azure Mobile Services provides all of this and more in a really slick and scalable way, all running on node.js in Azure's cloud. In this session, Taulty will demo Azure Mobile Services and talk about its support for Windows, Windows Phone, iOS, Android and Web and you'll be surprised how much we can get done in a session like this.

◆

**Nuno Filipe Mendes Godinho**

### WHAT'S NEW IN WINDOWS AZURE

In this session, Godinho will look at how different problems can require different storage solutions. He will demonstrate when and how SQL Azure and Windows Azure Storage can be used, in order to scale and control costs. This is important since we need to choose a real and functional way to split our data and do horizontal, vertical and hybrid partitioning.

◆

**Brian Randell**

### HEAVEN OR HOLLYWOOD: BUILDING GREAT UX WITH NUI

In this session, Randell will show you how to build great modern experiences including touch, voice and motion, using C# and .NET. He will dig into design patterns around NUI and show you how to build rich experiences that support more than just the traditional keyboard and mouse. He will go through the entire process, from wire framing and development, to the testing of an app that supports touch, as well as voice and motion via Kinect.

◆

**Robert Boedigheimer**

### WEB PERFORMANCE – LIVE SITE REVIEWS!

Do you have questions about your web site's performance? Would you like to have your site reviewed live? Don't worry, Boedigheimer isn't selling anything, but you'll be selling a lot more when your site is so much faster! There are many techniques, such as HTTP compression, caching with expirations, bundling and minification, image optimisation, CDNs and more, that he will review in the context of attendees' real web sites. He will also demonstrate tools such as Fiddler and Page Speed, which you can use to diagnose performance on your own. Ever wanted a free performance review of your web site with tips on how to improve it? Now's your chance!

◆

**Kevlin Henney**

### TEST SMELLS AND FRAGRANCES

Unit testing is now considered a mainstream practice, but that does not mean it is as common, pervasive or as well understood as it could or should be. Many programmers struggle with the quality of their tests and with the focus of their code. For some programmers, products and projects, tests can be more of a challenge than any other aspect of the system, which is why tests are often inadequate or left to one side. In this session, Henney takes a look at which characteristics make for unit tests that smell, and which make for more fragrant unit tests.

◆

**Andrew Clymer**

### BRAND NEW WORLD OF ASYNCHRONOUS PROGRAMMING

TPL Dataflow is a downloadable addition to the code TPL (Task Parallel Library) that ships with the .NET framework. It provides an alternative approach to define concurrency. Instead of simply throwing threads at synchronously structured programming and having to deal with all the thread safe and race conditions that introduces, we have the concept of many autonomous objects each with its own thread of execution. These autonomous objects co-operate with other such objects through asynchronous message passing. In this session, Clymer will show how TPL Dataflow can greatly reduce the complexity normally associated with asynchronous programming.

◆◆

**Shay Friedman**

### THE WONDERFUL WORLD THAT IS TWITTER BOOTSTRAP

So you're building for the web, right? And you're having fun, right? Inventing the wheel each and every time when you create this "float-right" class, right? Feeling the warmth of a thousand suns when you design yet another form, right? That's not so much fun anymore, eh? Feel sad no more! Twitter Bootstrap is a comprehensive CSS framework that makes all of the usual web design stuff much easier. It's so good, you might enjoy CSS once again! Come to the session to see what Twitter Bootstrap is, why it is so awesome, and how you create smooth-looking web sites in a matter of minutes!

◆◆

**Gil Fink**

### INTRODUCTION TO HTML5

HTML is the mark-up language that every web developer uses to structure and present content in the Internet. HTML5 is the standard that is being shaped and developed currently. It extends and improves the last HTML4 standard and takes it to the next level with multimedia, communication support and more. In this session you will get to know what HTML5 is and how you can use it even now in your web applications/sites.

◆

**Sebastien Lambla**

### HTTP CACHING ON .NET

HTTP caching is full of wonderful hidden gems that few developers know about. In this session, Lambla will explore how the combined power of OWIN and OpenRasta can leverage the main features of HTTP, and allow your infrastructure to cache, expand and scale, whatever your web framework of choice is.

◆◆

## DAY 3 AGENDA: MAIN CONFERENCE STREAMED SESSIONS

08.30: REGISTRATION AND COFFEE

9.30

**Sander Hoogendoorn**

### HOW FRAMEWORKS CAN KILL YOUR PROJECTS AND PATTERNS TO AVOID BEING KILLED

When it comes to writing code, a seemingly endless stream of new frameworks hits the streets every year to help you. Or even every month. And, yes, frameworks can help you write better code faster. But also, once you apply one or more frameworks to a project, trouble begins. What if you require features that aren't implemented? What if it contains bugs or omissions? And what if a new version is released that is implemented differently? These problems can bring your project a halt. In this session, Hoogendoorn demonstrates pragmatic architectures and patterns that will help your projects avoid framework issues and to keep code independent of framework choices.

◆

**Sasha Goldshtein**

### WHAT'S NEW IN C++ 11?

With the introduction of version 11, C++ is enjoying a renaissance. But the changes are so substantial that it's now almost an entirely new language. The new C++ style is based on lambda functions, rvalue references, automatic type inference, variadic templates, new standard library collections, smart pointers and many other features. In this session, Goldshtein will show how these features fit into existing C++ programs and how modern C++ development is up to par in productivity and performance with any other language.

◆

**Allen Holub**

### MESSAGING WITH RABBITMQ

Messaging is one of the most effective ways to pass non-time-critical information between servers. It's ideal for use with remote databases, logging, monitoring and so forth. RabbitMQ is one of the most flexible of the messaging systems. It's robust, open source, easy to use and supports pretty much all of the developer platforms. In this session, Holub will discuss messaging in general and how best to apply it in your application. He then goes on to discuss Rabbit's architecture and programming.

◆

**Dejan Sarka**

### EXCEL 2013 AND OFFICE 365 POWER BI

Excel is *the* analytical tool in the Microsoft Office suite for analysing both relational and unstructured big data. In this session, Sarka explores some of Excel's technologies: Power Pivot, through which Excel becomes a single-user analysis service in tabular mode; Power View and Power Map, for creating ad-hoc reports on the tabular model with minimal effort; and Power Query, giving you a data search engine, so you can query data from within your business and from external data sources. Sarka will also discuss the Office 365 BI sites, which provide the infrastructure for publishing your reports, if you do not use an internal infrastructure.

◆◆◆

**Roy Osherove**

### REFACTORING SKILLS FOR TDD

You have been trying to learn TDD, and it's going pretty well, but your code looks horrible. Your design skills are lacking, and trying to let TDD drive the design without having design skills can lead to a problematic, although testable, design of your code. This workshop will specifically deal with design skills, and assumes that you already know how to write unit tests. In this session, Osherove, the author of "The Art of Unit Testing", walks through important techniques for refactoring and design of code that will be either test driven or refactored for testability as part of a test-driven legacy effort. He will give real examples in .NET and Java. And most of the workshop will be hands-on pairing and refactoring on real code. Osherove will cover the following topics: clean code *[continued below]*

11.00: COFFEE BREAK

11.30

**Yaniv Rodenski**
**Ido Flatow**

### BATTLE OF THE FRAMEWORKS: ASP.NET VS. NODE.JS

It seems that Node.js is the coolest kid on the block. More and more companies, from start-ups to giants such as Microsoft and Linkedin, are using Node.js in their applications. But can this framework compete with a veteran such as ASP.NET? There are many considerations for using either but, in this session, Rodenski and Flatow will just let the frameworks fight it out.

◆

**Giles Davies**

### WHAT'S NEW IN VISUAL STUDIO 2013

Discover what's new in Visual Studio 2013. In this session, you can hear from Microsoft about new features around developer productivity, web development, Windows 8.1 app development, .NET 4.5.1 and Team Foundation Server. Davies will outline what's new in 2013, and provide numerous demonstrations in each of these areas. Knowledge of earlier versions is assumed.

◆◆

**Dan Clark**

### USING HIVE TO QUERY AND PROCESS BIG DATA

Writing map-reduce jobs to process data is not a trivial experience. This is a time-consuming task, even for a seasoned Java or C# programmer. Hive is a useful tool for creating and running map-reduce jobs in Hadoop. HiveQL is a declarative language, modelled to provide a similar experience to writing SQL: you construct and run the statement, submitting it to a query engine; the query engine transforms the query into a series of map-reduce jobs. By not having to worry about the low-level coding you become more productive and can concentrate on data analysis. This session will get you up to speed with Hive, and includes several practical examples.

◆

**Bob Beauchemin**

### NEW CARDINALITY ESTIMATING FOR QUERY PLANS

Having accurate row estimates is crucial to obtain the best query plans. In SQL Server 2014, the cardinality estimating calculators have been vastly overhauled based on a decade or more of real-world experience with the most difficult query patterns. Among the patterns that give improved estimates are ever-increasing keys and columns with correlation. One of the most interesting features is the instrumentation of the cardinality calculators through extended events; you can see how the estimate is produced.

◆

and SOLID design principles; design for testability; refactoring patterns on hard-to-test legacy code; writing tests against refactored code; when it does and does not make sense to refactor; and open-source projects and how we would refactor them.

Full-day workshop

◆

13.00: LUNCH BREAK

## DAY 3 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

**Ralf Westphal**

### MAKE REFACTORING UNAVOIDABLE – TDD AS IF YOU MEANT IT

TDD as usually taught means well, but often falls short of delivering on good design. That's because good design is optional, even though it's supposed to be a natural result of TDD's third step: refactor. In practice, many developers skip refactoring in favour of more red+green progress. Fortunately, there is a way out of this atrophied TDD application. It's called "TDD as if you meant it" (TDDaiymi), and makes it impossible to skip refactoring. In this session, you can see how TDDaiymi is done. Be surprised by how the code for a common code kata looks different when implemented by doing TDD as if you meant it.

◆

**Paul Ardeleanu**

### IOS DEVELOPER OVERVIEW

So, you're a developer, but you have never worked with iOS? In this session, Ardeleanu will provide a head start on how to plan, build, debug and release iOS 7 apps. The talk will cover a wide range of topics, including: the App Store ecosystem; the nature of an app; iOS 7's new UI and how to migrate from iOS 6; interface design; dev tools (Xcode – IDE, Simulator, Instruments – docs, etc); design patterns; debugging and optimisation; App Store and ad-hoc distribution; and what makes an app successfully.

◆

**Allen Holub**

### LIVING ON CLOUD 1001: A COMPARATIVE LOOK AT AMAZON AND GOOGLE CLOUD PLATFORMS

In this session, Holub provides a comparative overview of the major "Cloud" deployment environments from a programmer's perspective. He will define what "Cloud" actually means, and then look at the two main cloud architectures and ecosystems: Amazon's virtual-machine model (EC2 and related services) and Google's application-server model (App Engine).

◆

**Pavel Skribtsov**

### COMPUTING LIKE THE BRAIN: INTRODUCTORY GUIDE TO AI

Every now and again, every professional developer faces a program that he or she has trouble writing. Try to imagine an algorithm that has to differentiate a dog from a cat. They come in different shapes and sizes, and there is no single feature that could discriminate between the two. Any attempt to code that algorithm manually using deep-nested "if/else" branches is doomed. On the other hand, people have no trouble with this task. In this session, Skribtsov will introduce the basics of an artificial intelligence-based approach to solving these problems.

◆

**Robert Boedigheimer**

### MOBILE WEB SITES

Are you tired of hearing how great native apps are? Boedigheimer certainly is. In this session, he will explain the differences between native apps, hybrid apps and mobile web sites; review "responsive web design" and other approaches to create mobile web sites for various form factors (smartphone, tablets, desktops, etc); demonstrate how to provide your visitors with a great experience while using your existing web skills (or new ones with HTML 5 and CSS 3); and show how to avoid re-writing the same solution for each major mobile platform.

◆ ◆

**Dino Esposito**

### A FIRST LOOK AT ASP.NET IDENTITY

ASP.NET Identity is the new and comprehensive membership system for the whole ASP.NET platform, including Web API and SignalR. Similar in many ways to the popular simple membership provider, the new ASP.NET Identity goes well beyond that in a number of aspects: replaceable storage, flexible representation of user profiles, external logins, claims-based authentication, and role providers. In this session, Esposito will go through a list of examples showcasing the most interesting parts of the framework.

◆ ◆

**Brian Randell**

### CROSS-PLATFORM DEV (IOS, ANDROID AND JAVA) WITH TFS AND TEAM EXPLORER EVERYWHERE

You build web sites. You build for iOS. You use Java. But you work in an environment where they want you to use Team Foundation Server (or the cloud-based Team Foundation Service) for version control and work item tracking. Well, great. That's not a problem. It's the modern era and Microsoft is a friend not a foe. In this session, Randell will show you how you can work your way while still storing your bits in TFS. Oh, and guess what? You can even use Git with it. You'll learn about builds and getting your product out the door.

◆

**Bob Beauchemin**

### SQL SERVER CLOUD CHOICES – USE WINDOWS AZURE SQL DATABASE OR SQL SERVER IN A VM?

In this session, Beauchemin covers the trade-offs and choices to consider when moving your SQL Server application to the cloud. He will also explain the differences between the two implementations, as well as integration between both on-premises/Windows Azure SQL and on-premises SQL Server in VM.

◆ ◆

**David Britch**

### BUILDING NATIVE WINDOWS STORE APPS FOR C# DEVELOPERS

Learn how to build native Windows Store apps, in order to create fast and fluid user experiences. In this session, Britch will demonstrate how to build Windows Store apps using XAML and C++/CX, and natively implement development patterns and practices that will be familiar to C# developers. This will be done using a simple, two-page, photo-viewer app. The first page displaying a thumbnail gallery, selecting one of which navigates the user to the second page, where the photo is displayed full screen, and photo effects can be performed. C++ is the language for power and performance, and the combination of C++ 11 and C++/CX makes C++/CX read a lot like C#, while giving you the benefits of native code.

◆ ◆

**Ido Flatow**

### IIS FOR DEVELOPERS

Developers rely on IIS to host their application, but for years it was considered IT's domain and we weren't allowed near it. But ever since IIS 7 introduced new architectural changes, more control has been delegated to the developer. In this session, Flatow will demonstrate how IIS 8 works, how to deploy applications to IIS, configure it for better performance, and how to use it to debug your applications.

◆ ◆

## DAY 3 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

13.00: LUNCH BREAK

14.00

**Udi Dahan**

### PROGRAMMING IN THE FOURTH DIMENSION

There are certain types of requirements that necessitate developers thinking in four dimensions. Specifically, dealing with the passage of time in various business processes. In this session, Dahan will take you through some common scenarios that will show the kind of business problems that are created by traditional programming techniques, as well as solutions from the field of Event-Stream Processing (also known as Complex Event Processing). Although there is foundational support for these patterns in most queuing technology, we'll see why more supportive debugging and visualisation tools are needed. You, too, can be a Time Lord.

◆

**Sasha Goldshtein**

### TASK AND DATA PARALLELISM: REAL-WORLD EXAMPLES

Many developers have seen the Task Parallel Library APIs for concurrent applications, but have only played around with toy examples. In this session, Goldshtein will demonstrate how to extract concurrency and parallelism from seemingly impossible situations, how to gain scalability from lock-free code, and how to analyse real-world parallel applications with profilers to see the precise benefits gained from parallelisation.

◆

**Michael Kennedy**

### APPLIED NOSQL IN .NET

Perhaps you've heard about the next generation of databases roughly classified as NoSQL databases? These databases are generally much better than RDBMS at scaling, performance and ease-of-development (for instance, in NoSQL the object-relational impedance mismatch usually disappears). Unfortunately, many talks on NoSQL are very academic and general. Not this one. In this session, Kennedy will explore the NoSQL landscape and look at the various options out there. Then he'll demonstrate how to leverage MongoDB (a popular NoSQL DB) to build .NET applications using LINQ as the data access language. From there he will build a .NET application using LINQ and MongoDB in a series of interactive demos using Visual Studio 2012 and C#.

◆

**Klaus Aschenbrenner**

### UNIQUEIDENTIFIERS AS CLUSTERED KEYS IN SQL SERVER

Everybody is doing it; nobody wants to talk about it in public – uniqueidentifiers that are used as clustered keys in SQL Server. They have a lot of pros for devs, but DBAs just cry when they see them used in this manner. In this session, Aschenbrenner will cover the basics about uniqueidentifiers, why they are good and bad, and how you can find out if they affect the performance of your database. If they are affecting your database negatively, you will also learn some best practices how you can resolve those performance limitations without changing your underlying application.

◆

**Roy Osherove**

### REFACTORING SKILLS FOR TDD

Full-day workshop

◆

15.30: COFFEE BREAK

16.00

**Pearl Chen**

### BETA.TELUS.COM: RESPONSIVE, ITERATIVE, COLLABORATIVE

In early 2013, a skunkworks team of designers, developers and strategists came together to rethink the website of one of Canada's largest telcos. Beta.telus.com was the first project to come out of the TELUS Digital Labs, and there were many lessons learned along the way, such as how to implement Lean methodology in an Enterprise-loving company, how to get designers and developers rapid prototyping together, and more. What worked and what needed re-work in this large mobile-first, responsive redesign? Get a behind the scenes look into some of the processes, tools and technologies used to get beta.telus.com off the ground in only four months, with updates almost every week.

◆◆

**Ed Courtenay**

### WHAT ARE AUTO-MOCKING CONTAINERS, AND WHY SHOULD YOU USE THEM?

Explore how to use auto-mocking containers to improve and streamline your unit tests, and to exercise your IoC container in a testable fashion. Using Ninject and the Ninject Mocking Kernel package as an example, this demo-led session will take you through the problem space that auto-mocking containers address, covering lessons learned from a large-scale development. Starting from a simple project, Courtenay will demonstrate the fragility of existing unit tests and how developers can be discouraged from making comprehensive tests because they can be an inhibitor to refactoring processes. He will then introduce the idea of auto-mocking containers by demonstrating the use of the Ninject Mocking Kernel.

◆

**Dejan Sarka**

### SQL SERVER TRANSACTION ISOLATION LEVELS

SQL Server has a complete set of transaction isolation levels, providing any of the four pessimistic and two optimistic locking levels. In order to achieve a good compromise between data integrity and performance suitable for business needs, it is essential to choose the right one. A developer needs to thoroughly understand the possible problems that can be caused by selecting an inappropriate isolation level. So in this session, Sarka will demonstrate how they all work, explaining the infrastructure behind each one, and leave you with an understanding of which levels are best suited to which problems.

◆

**Matt Milner**

### DID YOU GET MY MESSAGE? BROKERED MESSAGING ON WINDOWS AND BEYOND

In this session, Milner will explain the importance of brokered messaging and the various tools you can use to implement it. He will discuss MSMQ, Service Bus for Windows and Windows Azure, and RabbitMQ. You will learn about core brokered messaging concepts and how each of these tools supports them. If you are building distributed systems and aren't using a message broker in some fashion, you need to come to this session to learn why you should. Demonstrations will cover simple queued messaging with all frameworks, scale out with multiple receivers, integrating RabbitMQ with Service Bus using AMQP, and much, much more.

◆

**Roy Osherove**

### REFACTORING SKILLS FOR TDD

Full-day workshop

◆

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

## DAY 4 AGENDA: MAIN CONFERENCE STREAMED SESSIONS

**Sahil Malik**

### CUSTOM SECURITY TRIMMING FOR SEARCH IN SHAREPOINT 2013

Learn about security trimming in SharePoint 2013 search and how claims-based authentication is securing your content. In this session, Malik will describe and demonstrate how security trimming works end to end, from content processing to search front end. This session is quite hands-on and will also demonstrate how custom security trimmers can be used. You need to know basic SharePoint to attend.

◆◆

**Ralf Westphal**

### INCREASE CHANGEABILITY – OOP AS IF YOU MEANT IT

OOP means well, but often falls short when it comes to delivering changeability. The larger OO software gets, the more it becomes a monolith – that's the experience of many teams, despite SOLID investments in good design. How come? Maybe it´s because one of the fundamentals principles of OO is not adhered to: today´s objects just don´t communicate by messaging. Fortunately, that can change quite simply. Messaging is not so much a matter of syntax, than a matter of thinking. In this session, Westphal will present a simple definition of messaging, plus two rules of how to organise your code for more changeability. It´s simple, and it´s even more OO than before.

◆

**Nuno Filipe Mendes Godinho**

### LESSONS LEARNED ON HOW TO SELECT YOU CLOUD COMPUTING VENDOR

Cloud computing is here to stay, and one of the important things to understand is how to choose your cloud vendor, independently of the "flavour" of the cloud they are providing, be it SaaS, PaaS or IaaS, public, private, hybrid or community. All of this is important, but it's also important to know what to ask them, and what responses to expect. In this session, Godinho will share his experiences of choosing a cloud vendor and present a complete set of questions everyone should ask before they commit.

◆

**Richard Blewett**

### INTRODUCING UNIT TESTING IN LEGACY CODE

Legacy code presents a problem for introducing unit tests: you general can't test without changing the code but if you change the code you have no tests to verify it still works. In this session, Blewett will demonstrate techniques for introducing tests into your legacy code base using coding patterns and tools such as Microsoft Fakes shim support.

◆

**Gil Fink**

### CREATING DATA-DRIVEN HTML5 APPLICATIONS

HTML5 is the web standard that is being shaped and developed currently. It extends and improves the previous HTML4 standard and takes it to the next level with support for multimedia, communication, semantics and more. In this session, Fink will take a close look at the new storage options that HTML5 brings and how to use them. He will build a grocery-store-list app that will use in-memory storage, and then refactor it to use web storage. He will then add AppCache to take the app offline. Finally, he will replace the storage mechanism to use IndexedDB instead of web storage.

◆

**Seb Rose**

### SO LONG, AND THANKS FOR ALL THE TESTS

TDD has long been promoted by agile practitioners, but the community still argues about how to go about it. Inside-out or outside-in? Mockist or classical? Through a component's public API or for every class? And then there's Kent Beck's famous quote: "I get paid for code that works, not for tests, so my philosophy is to test as little as possible to reach a given level of confidence." This introduces a further level of subjectivity, especially since developers are frequently overconfident. In this session, Rose will explore the choices that agile teams need to make when considering which development practices to adopt. He'll look at some of the different approaches and urge teams to practice until you're happy with the way you code.

◆

**Roy Osherove**

### MOCKING AND ISOLATION FRAMEWORKS DEEP DIVE IN .NET

Which isolation framework should you choose? What is the difference between constrained and unconstrained frameworks? Why do some frameworks support more features than others? How does that affect design, if at all? What are the values of a good isolation framework? In this session, Osherove will explore these questions and more.

◆

**Austin Bingham**

### A CULTURE OF REVIEW

Code and design reviews can be an effective and powerful tool for maintaining software quality and spreading knowledge. Done poorly, however, they can slow down development, alienate developers and lower morale. In this session, Bingham will look at how to build a "culture of review", where reviews – even critical ones – are seen as positive; where they improve quality; and where they're an integral part of the goal of building an effective software team. He'll look at what we know (and don't know!) about reviews, and he'll examine the various beneficial ways to use reviews in your work.

◆

**Sahil Malik**

### SHAREPOINT APPS: AN OVERVIEW

SharePoint developer or not, apps allow you to author functionality for SharePoint, without having to deal with SharePoint. In this session, Malik will give you a real-world overview of how to write apps for SharePoint: what not to do, and what is just not going to work out of the box, but you'll probably have to do.

◆

**Ido Flatow**

### ALREADY FAMILIAR WITH ASP.NET WEB API? YOU SURE?

As with any framework, there are those who know how to use ASP.NET Web API, and there are those who know how to use ASP.NET Web API. Want to be the second type of person? In this session, Flatow will skip the basic introduction to ASP.NET Web API and move on to the interesting stuff: pipeline architecture, extensibility, asynchronous actions, security and implementing HTTP concepts such as streaming and caching.

◆

## DAY 4 AGENDA: MAIN CONFERENCE STREAMED SESSIONS

08.30: REGISTRATION AND COFFEE

9.30

**Allen Holub**

### KNOCKOUT: AN INTRODUCTION

The Knockout framework is a standalone implementation of the MVVM (Model View ViewModel) pattern, which is one of the best user-interface architectures for web applications. It implements "declarative bindings" and automatic UI refresh, which in practice, automatically updates the user interface as the underlying data model's state changes. When used with a "server push" package, such as LightStreamer, you can modify a value on the server and that modification will be immediately visible on the web client (and vice versa). In this session, Holub will look at Knockout's architecture and how to leverage that architecture to build highly interactive web-application user interfaces. There will be several JavaScript examples.

◆

**Mike Taulty**

### WINDOWS 8.1 APPS: WHAT'S NEW FOR APP DEVELOPERS?

Windows 8 was a dramatic shift: a new version of the operating system with new input mechanisms, new devices, new apps on a new platform, and a new Store. Only a year later, Windows 8.1 came along with thousands of refinements and enhancements to both the operating system and the app platform. In this session, Taulty will pick out some highlights and walk you through code to show the main areas that have changed for a .NET developer who's already building Windows 8 apps or who's coming to Windows 8.1 new for the first time.

◆

**Dejan Sarka**

### THE ART (AND ARIMA) OF FORECASTING

Forecasting is not easy, especially if it is about the future. Using the Time Series algorithm in SQL Server Analysis Services is a quite straightforward process. However, many times you do not get valuable results. In this session, Sarka will talk about Auto-Regressive Trees (ART) and Auto-Regressive Integrated Moving Averages (ARIMA) algorithms in depth, explaining why you get what you get, and how can you overcome typical forecasting problems. He will also show how you can measure the quality of your forecasts, and discuss the data preparation for forecasting.

◆

**Bob Beachemin**

### WORK WITH SPATIAL DATA IN SQL SERVER

Every phone, tablet and Twitter message contains spatial data. In this session, Beauchemin will demonstrate how to import your spatial data into SQL Server and use it there. He will also include a discussion/demo of using SQL Server's spatial library with other data platforms for analysis.

◆

**Neal Ford**

### CONTINUOUS DELIVERY WORKSHOP

Getting software released to users is often a painful, risky and time-consuming process. In this workshop, Ford sets out the principles and technical practices that enable rapid, incremental delivery of high-quality, valuable new functionality to users. Through automation of the build, deployment and testing process, and improved collaboration between developers, testers and operations, delivery teams can get changes released in a matter of hours – sometimes even minutes – no matter what the size of a project or the complexity of its code base. To begin with, Ford will move from release, back through testing, to development practices, analysing at each stage how to improve collaboration and increase

*[continued below]*

11.00: COFFEE BREAK

11.30

**Brian Randell**

### MAKING THE MOST OF THE TFS SERVICE

Do you have a distributed team? Are you looking for someone to take care of your team's source code repository and bug list so you don't have to? If so, take a look at Team Foundation Service. Located at tfs.visualstudio.com, it brings together Microsoft Visual Studio, Team Foundation Server and Windows Azure. In this session, Randell will show you how to get started with the service, load up code, track your work and run builds. More importantly, he will help you understand the pros and cons of moving to the cloud. He'll also discuss how Team Foundation Service differs from Team Foundation Server, and what that means to you and your team.

◆◆

**Kevlin Henney**

### SEVEN INEFFECTIVE CODING HABITS OF MANY PROGRAMMERS

Habits help you manage the complexity of code. You apply existing skills and knowledge automatically to the detail, while focusing on the bigger picture. But because you acquire habits largely by imitation, and rarely question them, how do you know your habits are effective? Many of the habits that programmers have for naming, formatting, commenting and unit testing do not stand up as rational and practical on closer inspection. In this session, Henney examines seven coding habits that are not as effective as programmers believe, and suggests alternatives.

◆

**Michael Kennedy**

### HIGH-PERFORMANCE NOSQL TECHNIQUES

You're one of the brave ones who has jumped into the NoSQL pool and found it a refreshing change. That's awesome. But there is so much more to being successful with NoSQL databases than simply getting started. In this session, Kennedy will explore some of the issues, techniques and best practices for being successful with NoSQL, in general (and MongoDB, in particular). This includes exploring correct document/entity design, indexes and deployment – to name just a few of the topics. If you're getting started with NoSQL, this session should help you take things to the next level.

◆

**Udi Dahan**

### LOOSELY-COUPLED ORCHESTRATION WITH MESSAGING

Event-driven architecture (EDA) is one of the most powerful, yet least utilised, architectural approaches to building distributed systems. Ever since callbacks made their way into programming languages, developers have been arguing whether the benefits of looser-coupling outweigh the drawback seeing the control flow in one place. These arguments continue as developers look to leverage technologies such as RabbitMQ on-premise, or Azure Queues and Amazon SQS in the cloud. In this session, Dahan will demonstrate which development practices can bring the global flow visualisation of orchestration to your publish/subscribe messaging architectures to get the best of both worlds.

◆

feedback so as to make the delivery process as fast and efficient as possible. Later, Ford will introduce agile infrastructure, including the use of Puppet to automate the management of testing and production environments. He will discuss automating data management, including migrations. Development practices that enable incremental development and delivery will be covered at length, including a discussion of why branching is inimical to continuous delivery.

Full-day workshop

◆

13.00: LUNCH BREAK

## DAY 4 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

**Sander Hoogendoorn**

### CROSSING THE CHASM FROM WEB TO WINDOWS 8 DEVELOPMENT (AND SOME ANDROID TOO)

With the introduction of the new Windows 8 / Windows Store platform, Microsoft has delivered a new software development paradigm. Building applications for this platform is different from ASP.NET web development, and even differs from building WPF and Silverlight applications. Or is it? In this session, Hoogendoorn will demonstrate the software architecture, frameworks and patterns that his team is using to migrate from ASP.NET and traditional desktop development to Windows 8 / Windows Store development. Using these advanced techniques, developers with knowledge of those platforms are capable of crossing the chasm to the new paradigm.

◆◆

**Sasha Goldshtein**

### FIRST STEPS IN IOS APPLICATION DEVELOPMENT

There are iPhones and iPads everywhere, and you want to get your app through the gates of the App Store? This session will set you up for iOS application development. Goldshtein will cover the basics of Xcode and Objective C, create an iOS application using storyboards, and explore some additional iOS APIs through a live coding demo.

◆

**Nuno Filipe Mendes Godinho**

### REAL-TIME DATA MANAGEMENT IN THE CLOUD

Everyone is talking about Big Data problems and thinking about technologies such as Hadoop, Elastic MR and the like. But these are used for data analytics, not in real time. When we think about real-time systems, such as CCTV analysis for disturbances, financial markets, manufacturing with real-time production adjustments, everything changes. In this session, Godinho will discuss the nature of real-time and scenarios where it is relevant, event-driven architectures, the products and frameworks that can help, NServiceBus, NEsper, MessageHandler and reactive extensions.

◆

**Simon Robinson**

### .NET COLLECTIONS

Are you certain that of the many available .NET collections, you always use the most appropriate one? If you are using dictionaries, does your code implement hash codes correctly? Do you worry about thread safety issues when references to collections are passed around? In this session, Robinson will go under the bonnet to look at how different collections work, what they are appropriate for, best practices for using them, and how to deal with thread safety. You'll also learn about the new immutable collections.

◆

**Sebastien Lambla**

### REST ON .NET

Resources are at the core of the web, as intended. In this session, building on seven years of developing resource-oriented frameworks, Lambla will demonstrate how you can build an efficient, feature-rich framework to develop the web applications of tomorrow, based on the power of .NET and OWIN.

◆◆

**Seb Rose**

### MUTATION TESTING – WHY YOU SHOULD CARE

Do you know how good your tests are? Mutation testing can tell you. Unlike test coverage metrics (which only tell us how much of your application was executed, not whether the tests were any use), mutation testing lets us say something concrete about the quality of your test suite. Mutation testing has been around for years, but it's only recently that performance tools (such as PI Test for Java) have become available. In this session, Rose will look at the motivation and technology behind mutation testing and see some examples in action.

◆

**Simon Robinson**

### INSIDE WORD, EXCEL AND POWER-POINT DOCUMENTS: A DEVELOPER'S PERSPECTIVE

Most of us use Word, Excel or PowerPoint every day, but how well do you really understand what these apps are capable of? What features are supported in their file formats? Their user interfaces are designed for end users, not for programmers, and therefore tend to hide their internal document structures. In this session, Robinson will use Open XML to explore what lurks inside Office document files. You'll gain a better understanding of their document object models, which will help you both when you're coding with Office and when you're trying to use Office apps as an end user.

◆

**Shay Friedman**

### MIGRATING APPLICATIONS TO WINDOWS AZURE

Windows Azure is the next big thing for server-side applications and one of its major-use cases is hosting existing .NET applications. However, Window Azure is not your regular playground and some preparations are necessary. In this session, Friedman will take you through the migration path and the different ways to make sure your application is ready to move to the cloud. In addition, he will explain how you can estimate the cost of running your web application in the cloud.

◆

**David Britch**

### ACCELERATING WINDOWS STORE APP DEVELOPMENT USING PRISM FOR THE WINDOWS RUNTIME

Learn how to use Prism for the Windows Runtime to accelerate managed Windows Store app development. This is done by using its support for MVVM and commonly required core services. In this session, Britch – one of the Prism for the Windows Runtime team members – will demonstrate how to accelerate app development by using Prism's support for bootstrapping MVVM apps, data binding/commands, navigation, state management, displaying a Flyout, validation of user input and more. This will be done using a simple two-page photo-viewer app.

◆

**Yaniv Rodenski**

### SIGNALRITY

Building real-time web applications was a web developer's "Holy Grail" for too long. Even with the arrival of the long-awaited WebSockets protocol, developers still need to support a wide range of legacy browsers using a variety of techniques. Scaling out real-time web applications is also not trivial – data integrity must be maintained across servers. SignalR allows .NET developers to overcome these challenges using a clean and coherent API. It also comes with a set of client-side libraries supporting a variety of client environments, including JavaScript, .NET, WinRT and Windows Phone 8. In this session, Rodenski will demonstrate how to create real-time web applications using SignalR and how to scale these applications in a real-world environment.

◆

## DAY 4 AGENDA (CONTINUED): MAIN CONFERENCE STREAMED SESSIONS

13.00: LUNCH BREAK

14.00

**Sahil Malik**

**SHAREPOINT CSOM AND REST IN THE REAL WORLD**

CSOM and REST: the single most important SharePoint topic you need to be on top of. In this session, Malik will cover the architecture of REST and CSOM, what tools to use when developing, how to discover the API, advanced scenarios such as debugging and concurrency, best practices, and less-obvious things that will bite you. Ouch! Don't miss.

◆◆

**Ed Courtenay**

**GROWING CONFIGURABLE SOFTWARE WITH EXPRESSION<T>**

Learn how to use expression trees to enable configurable and testable business rules, without sacrificing performance. In this session, Courtenay will start by demonstrating an example project where business rules are controlled by configuration using a composite strategy pattern. The pattern will be refactored into a provider that supplies an expression tree, which will then be compiled. Simple benchmarking will be used to demonstrate the performance benefits of this approach. Another demo will show creating a simple expression tree and compiling it to a standalone DLL, which will then be executed by a test project.

◆◆

**Dejan Sarka**

**OPTIMISING TEMPORAL QUERIES (PART 1): AN INTRODUCTION**

Although temporal data is part of many business applications, most RDBMS, including SQL Server, do not support it out of the box. However, before solving the problem, you need to understand it. In this session, Sarka will provide an introduction to temporal problems, then go on to develop Interval CLR data type. He will then discuss what kind of constraints pertain to temporal data, and how we query temporal data. Please note, this is the theoretical introduction needed for the following Optimising temporal queries (Part 2): the optimisation session, intended to provide the groundwork for those who do not yet have a deep understanding of temporal problems.

◆

**David Evans**

**SPECIFICATION BY EXAMPLE**

Get practical, hands-on experience of Specification by example (SBE), one of the most important concepts in agile and iterative development. SBE is a set of process patterns that help agile software teams deliver value more reliably, by using the power of concrete examples as a means of collaborating on requirements and design, enabling a shared understanding between business and technical people. In this session, Evans will cover some of the key patterns of SBE, through a set of hands-on exercises, to ensure you get beyond the theory and into the practice, where deep learning takes place.

◆

**Neal Ford**

**CONINUOS DELIVERY WORKSHOP**

Full-day workshop

◆

15.30: COFFEE BREAK

16.00

**Andrew Clymer**

**PATTERNS IN PARALLEL PROGRAMMING WITH TPL**

The free lunch is over: applications no longer get a massive performance boost whenever a new CPU arrives. In order to take advantage of the increased power, applications need to be designed to take advantage of the multiple cores. This means a shift in the algorithms and techniques we employ. In the same way that OO has design patterns, parallel programming also has its own catalogue of patterns to solve re-occurring problems. In this session, Clymer will introduce a variety of parallel processing patterns and give examples of implementations using Microsoft's TPL from simple fork/join, divide and conquer, geometric decomposition, pipelines to the slightly bizarre Monte Carlo.

◆◆

**Robert Smallshire**

**DELIVER DOMAIN-DRIVEN DESIGNS – DYNAMICALLY!**

How will you model your ever-changing world? Domain-driven design (DDD) codifies perspectives on reality into a domain model, which is realised as a software system around which valuable services can be constructed. As business domains change, our requirements and models must follow. Dynamic languages, such as Python and Ruby, coupled with document databases, can produce flexible domain models that can gracefully accommodate new information that wasn't explicitly modelled by the original designers. Discover when and how dynamic language solutions are most appropriate for domain models, and understand the trade-offs.

◆

**Dejan Sarka**

**OPTIMISING TEMPORAL QUERIES (PART 2): THE OPTIMISATION**

Having a SQL Server solution for a problem does not mean the job is done. Of course, the next immediate issue is the performance. Temporal queries that involve intervals are typically very IO and CPU intensive. For example, a test for overlapping intervals was solved with inefficient queries for years. However, a handful of solutions with fast queries were developed lately. In this high-level technical session, Sarka introduces five different methods to get efficient queries that search for overlapping intervals, one of the most complex temporal problems. Of course, these solutions can be implemented on other temporal problems as well.

◆

**Bob Beauchemin**

**WHAT'S IN MICROSOFT'S HADOOP OFFERINGS?**

In this session, Beauchemin looks at Microsoft's implementation of Hadoop in conjunction/comparison with some of the more well-known offerings. See what Microsoft has added for .NET data programming and management. This session covers both the cloud implementation and "traditional" Hadoop running under Windows.

◆

**Neal Ford**

**CONINUOS DELIVERY WORKSHOP**

Full-day workshop

◆

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

## DAY 5 AGENDA: ALL-DAY POST-CONFERENCE WORKSHOPS

Friday 4th April

THE FOLLOWING WORKSHOPS RUN FOR A **FULL DAY, FROM 09.30 TO 17.30** WITH A SHORT BREAK IN THE MORNING AND AFTERNOON, AND A LUNCH BREAK AT 13.00.

UNLESS OTHERWISE NOTED IN THE DESCRIPTION, THEY ARE PRESENTATION-BASED IN STYLE RATHER THAN 'HANDS-ON' LABS.

*Please refer to page 2 for guide to colour symbols.*

### BUILDING SCALABLE JAVASCRIPT APPS

**Gil Fink**

**WORKSHOP REF: F1**

Building and maintaining large and scalable JavaScript web apps is not at all easy. So how you build such things without being driven into madness? Using and combining proven JavaScript patterns will do the trick.

**In this one-day workshop, Fink will talk about the patterns behind some of the largest JavaScript apps, such as Gmail and Twitter, and how to apply them in your own apps. He will start from object patterns and then focus in on module patterns, promises, timers and more.**

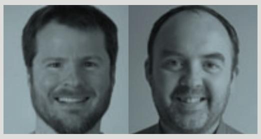
◆

### DELIVER DOMAIN-DRIVEN DESIGNS – DYNAMICALLY!

**Austin Bingham**

**Robert Smallshire**

**WORKSHOP REF: F2**



How do you statically model an ever-changing world? The approach of domain-driven design (DDD) describes the codification of perspectives of reality into a domain model, which is in turn realised as a software system around which valuable services can be constructed. As business and physical domains evolve, our requirements, models and implementations must follow if they are to remain relevant. Dynamic languages, such as Python are a great match for the dynamism of the real world. It is perhaps surprising then, that for much of the decade since its inception, DDD has manifested its results in rigid relational-database schemas, object relational mappers pushed beyond reasonable limits and inflexible object models in statically typed languages such as Java or C#.

**In this hands-on workshop, participants will work together to implement a domain model in Python using nothing more that plain old Python objects.**

Bingham and Smallshire will show how core DDD concepts, such as entities, immutable value objects, aggregates and repositories, can be implemented in Python. They will build declarative tools from scratch to facilitate model implementation, and they'll evaluate persistence solutions, including object databases, graph databases, "traditional" RDBMSs and document stores.

◆

### SHAREPOINT 2013 APP DEVELOPMENT

**Sahil Malik**

**WORKSHOP REF: F3**

Apps enable any developer to deliver SharePoint functionality. They allow you to use things such as MVC, TDD and everything else you like with SharePoint. Apps are the single biggest change between SharePoint 2010 and 2013. Is it surprising, then, to discover that the apps platform is full of gaping holes – missing functionality and pitfalls – things you need to watch out for?

**In this session, Malik will cover all those topics that you won't read on MSDN, but that you will discover in your projects. No stone will be left unturned.**

Those who will get the most out of this session are those who are either .NET developers who don't want to learn SharePoint – or are clinging on to best practices as they are being forced into delivering SharePoint functionality – or seasoned SharePoint developers who want to know the real deal on apps.

◆

### AGILE/OO-DESIGN, FROM START TO FINISH

**Allen Holub**

**WORKSHOP REF: F4**

Many people who think they're doing OO aren't. For example, the dynamic model (that shows how run-time objects interact) should drive the design process; the class diagram is an artefact you build while doing dynamic modelling. Fixating on the class diagram renders your program, at best unwieldy, at worst non-functional. Similarly, basic OO architectural goals (such as eliminating getter/setter functions) seem impossible to do unless you understand how the design process actually works. It turns out that the process you use influences both the quality and the basic structure of the design.

**In this workshop, Holub will cover an agile version of the OO-design process, with an emphasis on how to arrive at an optimal design.**

He'll provide a quick overview of the process, then spend much of the class working through one (or more if we have time) real-world examples that show you the entire process, from front to back: requirements gathering and problem-statement definition, use-case analysis (story development), and the simultaneous construction of the dynamic and static models using UML.

◆

**BOOK NOW**

BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200

DAY 5 AGENDA: ALL-DAY POST-CONFERENCE WORKSHOPS

8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

## FROM ZERO TO APP WITH NOSQL, MONGODB AND .NET

Michael Kennedy

WORKSHOP REF: F5

**Do you want leave this conference ready to build the next generation of maintainable and high performance applications in .NET? Then this is the workshop for you.** We assume you are a competent .NET developer but are otherwise basically new to NoSQL. We will build out and end-to-end data driven web application in ASP.NET MVC and MongoDB (the most popular NoSQL database around these days).

Here are just some of the topics we'll cover:

1. Why NoSQL and why document databases
2. Installation quick start
3. Your first app in MongoDB (.NET)
4. Understanding the shell and native js query language.
5. Designing entities and models
6. High performance techniques: Indexes, profiling and plan detection
7. Atomic operations, concurrency, and durability.
8. Management tools may fold into 1 or 2.
9. Distributed file system: Grid.FS
10. Performance / scaling mongodb (deployment techniquest: sharding, replica sets, etc)
11. Map / Reduce

◆

## EFFECTIVE USER STORIES

David Evans

WORKSHOP REF: F6

This practical tutorial addresses a number of challenges that agile teams face when working with user stories. The deceptively simple style of stories makes them initially appealing, but potentially dangerous if not handled well. Many teams easily fall into bad habits and story dysfunctions, from having oversized epics that live for multiple iterations, to swarms of sticky post-it notes that leave external stakeholders baffled as to what is actually going to be delivered and when.

**In this session, Evans will explore a range of practical tips and techniques that will help you regain control of your backlog and allow you to create stories that are expressive, meaningful, concise and valuable.**

This tutorial will be beneficial for anyone responsible for creating, implementing, accepting and collaborating on the development of stories. Topics covered include: stories as better requirements; naming, structure and writing style; acceptance criteria, examples and testing; backlog management: cards and systems; splitting and slicing large stories; communicating priorities and milestones: story maps; and collaboration, conversation and documentation. There will also be a number of practical exercises to support this content.

◆

## HOW DO THE COOL KIDS CREATE CHATS TODAY?

Shay Friedman

WORKSHOP REF: F7

In the last couple of years, the world of web development has grown tremendously, both in popularity and in its maturity. HTML has improved with HTML5 and its client side APIs, JavaScript has improved with end-to-end frameworks such as Angular JS, and CSS has improved with frameworks such as Twitter Bootstrap and LESS. At the same time, server-side technologies have improved and been adjusted to better fit the web, with frameworks such as SignalR.

**In this workshop, Friedman will build a chat web site from scratch, using all the technologies above, and see how web applications are being constructed today. Do you want to get a glimpse of what web development looks like today? This is the workshop for you!**

◆

## ANDROID FOR .NET DEVELOPERS

Sasha Goldshtein

WORKSHOP REF: F8

Android is the most popular smartphone platform, with hundreds of thousands of apps and millions of new devices activated every day.

**In this workshop, Goldshtein will explore the fundamentals of developing Android applications, including: setting up the Eclipse development environment, running apps on emulators and physical devices, building UI and connecting it to code, navigating across multiple activities, and storing data in files and SQLite.**

At the end of the day, you will be equipped to develop your own Android applications.

◆

## BUILDING DOMAIN-DRIVEN APPS WITH APACHE ISIS

Dan Haywood

WORKSHOP REF: F9

Domain-driven design is a great approach for building line-of-business enterprise applications, with the emphasis on the bit that matters: the domain. But maintaining all the artefacts of a custom-coded n-layer architecture (views, controllers, commands, bindings etc) can massively impede your ability to learn, explore and experiment as you go looking for those deeper domain insights.

What if you could build the system just by writing the core domain objects, and leave concerns such as the GUI (which as we know, often accounts for a substantial portion of the development effort) until later? In any case, GUIs follow fashions and trends, so we ought to consider the domain object model independently of the UI that sits in front of it.

**There's a pattern for this type of approach: naked objects. In this workshop, Haywood will show how you can use an open-source framework that implements this pattern – Apache Isis – to rapidly build domain-driven applications. You can use the framework for prototyping, or take your application through to production.**

◆

Do you want to get a glimpse of what web development looks like today? This is the workshop for you!

SHAY FRIEDMAN
HOW DO THE COOL KIDS CREATE CHATS TODAY?

## MEET YOUR SPEAKERS

DEVWEEK ATTRACTS A WORLD-CLASS SPEAKER FACULTY. FIND OUT MORE ABOUT THEIR EXPERIENCE AND EXPERTISE HERE.

### Paul Ardeleanu

**Paul** is an experienced iOS and web tinker, trainer and mentor, currently teaching iOS at SkillsMatter in London, UK. He started programming back when Fortran was cool and graduated with a PhD in Computational Physics from UCLan.

### Klaus Aschenbrenner

**Klaus** provides independent SQL Server consulting services across Europe and the US. He has worked with SQL Server 2005/2008/2012/2014 from the very beginning, and in 2004/2005 was honoured with the MVP award from Microsoft.

### Bob Beauchemin

**Bob** is a database-centric application practitioner and architect, writer, instructor, course author and Developer Skills Partner for SQL skills. Over the past few years he's been writing and teaching his SQL Server 2005-2012 courses worldwide.

### Austin Bingham

**Austin** is a founding director of Sixty North, a software consulting, training and application development company. He is also an experienced presenter and teacher, having spoken at a number of conferences and software groups.

### Richard Blewett

**Richard** has worked in the software industry for more than 20 years, starting with mainframes through the early years of client/server to today's service-oriented world. He now focuses on WCF, BizTalk, Workflow and Azure.

### Robert Boedigheimer

**Robert** works for Schwans Shared Services, LLC providing business solutions with web technologies and leads Robert Boedigheimer Consulting, LLC. He has been designing and developing web sites for 19 years, from the early days of ASP and ASP.NET.

### David Britch

**David** is a principal technologist at Content Master and, for the past 18 months, has been working on a number of projects with the patterns & practices group at Microsoft. David has a PhD in Computation, specialising in image and signal processing.

### Simon Brown

**Simon** works as an independent consultant, specialising in software architecture, technical leadership and the balance with agility. He is the founder of Coding the Architecture, a website about pragmatic, hands-on software architecture.

### Eewei Chen

**Eewei** has worked in the new media creative industry since 1993, and founded HaaYaa.com to collaborate with the media industry. He also teaches Lean Startup and Lean UX techniques at the Kingston Business School and Adaptive Marketing Institute in India.

### Pearl Chen

**Pearl** was most recently the Senior Manager Research & Technology at the Canadian Film Centre's Media Lab and she is currently a lead front-end web developer on the beta.telus.com redesign project at TELUS.

### Andrew Clymer

**Andy** cut his teeth working in various start-ups, before being bought up by Cisco in 1997. After a few years at Cisco, the pull of the start-up world was too much to resist, and Andy became a co-founder of Rock Solid Knowledge.

### Ed Courtenay

**Ed** is an experienced software developer and evangelist, who has been programming professionally for more than 25 years. He is now working for a major manufacturer and retailer in the UK, leading the development team responsible for the ecommerce systems.

### Dan Clark

**Dan** is a senior BI consultant for Pragmatic Works. He focuses on learning new BI technologies and training others how to best implement the technology. He is also a regular speaker at developer/database conferences and user group meetings.

### Udi Dahan

**Udi** is the Software Simplist, an internationally renowned expert on software architecture and design, and one of the world's thought-leaders in service-oriented architecture and domain-driven design. He is also the creator of NServiceBus.

### Giles Davies

**Giles** works in the Developer and Platform Evangelism Group in Microsoft UK as a technical specialist covering development tools. He specialises in Application Lifecycle Management tooling of Team Foundation Server and Visual Studio.

### Dino Esposito

**Dino** is a trainer and software consultant based in Rome. A member of the IDesign team, he specialises in Microsoft .NET technologies, and spends most of his time teaching and consulting across Europe and the USA.

### David Evans

**David** is an experienced agile consultant, coach and trainer with more than 25 years of IT experience. A thought-leader in the field of agile testing and Specification by example, he is a regular speaker at events and conferences across Europe.

### Gil Fink

**Gil** is an expert in web development and Microsoft web and data platforms. He works as a senior consultant and architect at Sela Group. And he is currently consulting for various enterprises and companies, helping to develop web and RIA-based solutions.

### Ido Flatow

**Ido** is a senior architect and trainer at Sela Group, and an expert on Windows Azure and web technologies, such as WCF, ASP.NET, IIS and Silverlight. He is also a Microsoft Integrated Systems MVP and a Microsoft certified trainer (MCT).

### Neal Ford

**Neal** is director, software architect and meme wrangler at ThoughtWorks, a global IT consultancy focused on end-to-end software development and delivery. He is also an internationally acclaimed speaker, speaking at more than 300 conferences worldwide.

### Shay Friedman

**Shay** is a Microsoft C# MVP and the author of IronRuby Unleashed. With more than 10 years of experience in the software industry, Friedman co-founded CodeValue, where he creates products for developers, consults and runs courses around the world.

### Iordanis Giannakakis

**Iordanis** is a software engineer with eight years of experience on the full stack. Based in London, he is currently the Android team lead at Shazam, where he is trying to promote BDD and software craftsmanship to developers.

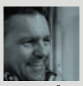
### Nuno Filipe Mendes Godinho

**Nuno** is the Director of Cloud Services, Europe, at Aditi Technologies, and has more than 14 years of experience in the industry. His main focus is helping customers identify, plan, manage and develop software products and solutions in the cloud.

### Sasha Goldshtein

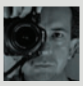
**Sasha** is the CTO of Sela Group, a Microsoft C# MVP, and an international consultant and trainer. He is also the author of numerous training courses, covering parallel programming, Android and iOS application development, .NET debugging and .NET performance.

**Dan Haywood**



**Dan** is a freelance consultant, author and trainer, specialising in domain-driven design, agile development and enterprise architecture on Java and .NET. He currently advises the Irish government on a major NO app, and builds applications on Apache Isis.

**Kevlin Henney**



**Kevlin** is a UK-based independent consultant and trainer, with interests in patterns, practice, process and programming. He has also been a columnist for web sites and magazines, including Better Software, The Register, and the C/C++ Users Journal.

**Allen Holub**

**Allen** is an internationally acknowledged expert in OO-design, agile process, Java and cloud-based web application development. He provides training and consulting services in those areas, and even slings code on occasion.

**Sander Hoogendoorn**

**Sander** is the Principal Technology Officer and Global Agile Thought-leader at Capgemini, where he is involved in the innovation of software development processes, requirements, architectures, patterns, frameworks and technologies.

**Allan Kelly**

**Allan** has held just about every job in the software world, from system admin to development manager, by way of programmer and product manager. Today, he works helping teams adopt and deepen agile practices, and writing far too much.

**Michael Kennedy**

As a full-time instructor for DevelopMentor, **Michael** specialises in core .NET, web, agile development, and Test-Driven Development (TDD) technologies. He has been building commercial applications with .NET since its initial public beta in 2001.

**Ben Lambert**

**Ben** is a developer and architect with more than 20 years of professional experience. He is currently the head of the development team at Active Documents, specialising in document management, system design, C# and Java development.

**Sebastien Lambla**

**Sebastien** runs Caffeine IT, a consultancy helping clients from all over Europe implement first-grade and innovative solutions, from brushing up software design skills to implementing enterprise-ready resource-oriented architectures.

**Sahil Malik**

**Sahil**, the founder and principal of Winsmarts.com, has been a Microsoft MVP and INETA speaker for the past 10 years. He has trained in the Microsoft technology space, and has architected and delivered SharePoint-based solutions for major clients.

**Jules May**

**Jules** is a software architect, currently working with Senergy, an energy services company in Scotland. He has been teaching and speaking for 25 years and conducts frequent lectures and workshops.

**Matt Milner**

**Matt** is an independent consultant and trainer who, from the early days of the Internet, was building websites and databases. Currently, Matt builds applications and teaches other developers about the latest Microsoft technologies.

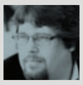
**Mark Murphy**

**Mark** is the founder of CommonsWare and a three-time entrepreneur. His experience ranges from consulting on open-source and collaborative development for the Fortune 500 to application development on just about anything smaller than a mainframe.

**Roy Osherove**

**Roy** is the chief scientist at Bouvet.no, and one of the original ALT.NET organisers. He consults and trains teams worldwide on the gentle art of unit testing, test-driven development and how to lead software teams, including many videos.

**Brian Randall**



**Brian** is a partner with MCW Technologies, an Endjin Associate and a Microsoft ALM MVP. He spends his time teaching Microsoft technologies to developers, working with new and emerging technologies, and consulting worldwide.

**Ian Robinson**

**Ian** is Neo Technology's Director of Customer Success, working with customers to design and develop graph database solutions. He is a co-author of Graph Databases (O'Reilly) and REST in Practice (O'Reilly).

**Simon Robinson**

**Simon** is a developer and Pluralsight instructor. His programming career has spanned industries ranging from academic research to telecoms to finance, and he has worked with both front-end and back-end code for Windows and web.

**Yaniv Rodenski**

**Yaniv** is a Senior Architect at Sela Group, with more than 16 years of experience as a developer, team leader, R&D manager and architect in various environments. He is also the founder and co-manager of the Windows Azure User Group in Israel.

**Seb Rose**

**Seb** wrote his first commercial software in the early 80s on an Apple II. He went on to graduate from the University of Edinburgh with a 1st Class Joint Honours in Computer Science and Electronics. He now focuses on helping teams adopt and refine agile practices.

**Dejan Sarka**

**Dejan** is an independent consultant, trainer and developer, focusing on database and business intelligence applications. His specialties include data modelling, data mining and data quality. He is also the founder of the Slovenian SQL Server and .NET Users Group.

**Pavel Skribtsov**

**Pavel** is a graduate from the Moscow Institute of Physics and Technologies (MIPT), with a PhD in "Neurocomputers application for representation of static and dynamic 3D data". He is also the founder, CEO and ideological leader of Pawlin Technologies Ltd.

**Robert Smallshire**

**Robert** is a founding director of Sixty North, a software product and consulting business in Norway. He has worked in senior architecture and technical management roles, is the organiser of the Oslo Python group, and holds a PhD in Natural Science.

**Mark Smith**

**Mark** has been involved with software development for more than 20 years, starting on mainframes (IBM VM/SP) and moving on to OS/2, and finally settling on Windows. He has extensive experience training and consulting for large companies.

**Mike Taulty**

**Mike** works in the Developer and Platform Group at Microsoft in the UK, where he helps developers get the best from the platform. Prior to this, he spent three years with Microsoft Consulting Services as a consultant on developer technologies.

**Jim Webber**

**Jim** is Chief Scientist with Neo Technology, the company behind the popular open-source graph database Neo4j. There, he works on R&D for highly scalable graph databases and writes open-source software.

**Ralf Westphal**

**Ralf** is a freelance consultant, project coach and trainer on software architectural topics and team organisation. He is the co-founder of the "Clean Code Developer" initiative to increase software quality.

## REGISTER AND PRICING OPTIONS

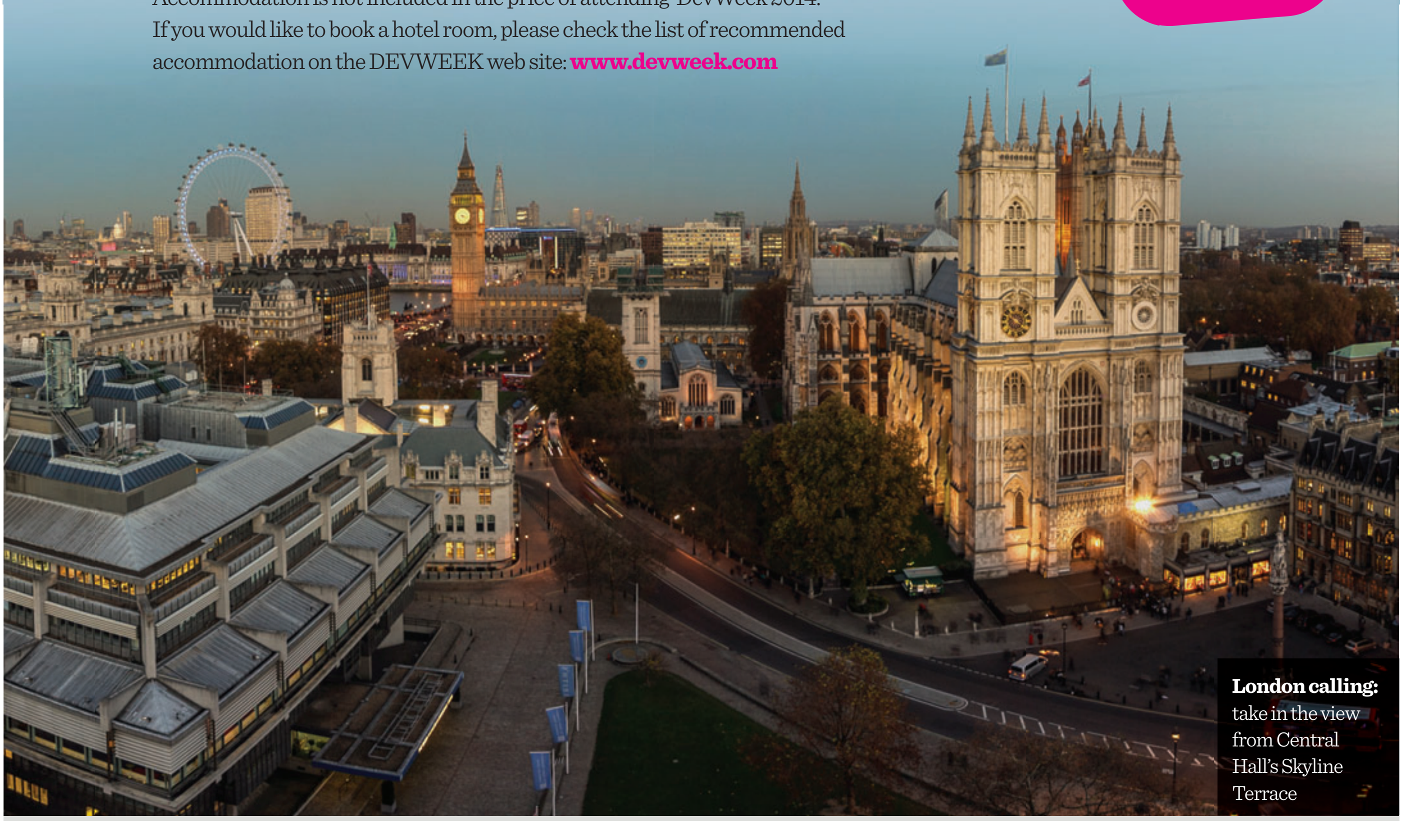
### DEVWEEK PRICING STRUCTURE

| All prices include refreshments, buffet lunch and session notes, but exclude travel and accommodation. | BOOKING BEFORE FRIDAY 20 DECEMBER 2013 SAVE UP TO £300 | BOOKING BEFORE FRIDAY 31 JANUARY 2014 SAVE UP TO £200 | BOOKING BEFORE FRIDAY 7 MARCH 2014 SAVE UP TO £100 | BOOKING AFTER FRIDAY 7 MARCH 2014 |
|---|---|---|---|---|
| UNIVERSAL PASS: All 5 days | £1495 +VAT | £1595 +VAT | £1695 +VAT | £1795 +VAT |
| 4-DAY PASS: Main conference + pre/post workshop | £1195 +VAT | £1295 +VAT | £1395 +VAT | £1495 +VAT |
| 3-DAY PASS: Main conference only | £895 +VAT | £995 +VAT | £1095 +VAT | £1195 +VAT |
| 1-DAY WORKSHOP-ONLY PASS | £345 +VAT | £395 +VAT | £445 +VAT | £495 +VAT |

**TO REGISTER, PLEASE VISIT DEVWEEK.COM**

**For further information telephone: +44 (0)20 7407 9964**
**or email: devweek@bsi.co.uk**
Accommodation is not included in the price of attending DevWeek 2014.
If you would like to book a hotel room, please check the list of recommended accommodation on the DEVWEEK web site: **www.devweek.com**

**BOOK NOW**
BOOK YOUR PLACE BY 31 JANUARY AND SAVE UP TO £200


**London calling:** take in the view from Central Hall's Skyline Terrace

Please note that submission of a completed registration form constitutes a firm booking, subject to the following terms and conditions. Any cancellations received after 17th January 2014 will incur an administration fee of 30%. Cancellations must be made in writing at least 60 days before the start of the conference, or the full fee will be charged. We are happy to accept substitutions if they are submitted in writing before the conference begins. The organisers reserve the right to make changes to the programme and speakers without notice if this is unavoidable. If delegates are unable to attend for any reason that is beyond the control of the organisers, such as transport problems, personal illness, bereavement, inclement weather, terrorism or act of God, it will not be possible to make any refunds of conference or workshop fees.