

# msdn<sup>®</sup> magazine

## SOA TIPS

Address Scalability Bottlenecks with Distributed Caching

Iqbal Khan ..... 24

## THREAD PERFORMANCE

Resource Contention Concurrency Profiling in Visual Studio 2010

Maxim Goldin ..... 38

## CLOUD DIAGNOSTICS

Take Control of Logging and Tracing in Windows Azure

Mike Kelly ..... 50

## EXPRESS YOURSELF

Encoding Videos Using Microsoft Expression Encoder 3 SDK

Adam Miller ..... 62

## INPUT VALIDATION

Enforcing Complex Business Data Rules with WPF

Brian Noyes ..... 74

## PRACTICAL ODATA

Building Rich Internet Apps with the Open Data Protocol

Shayne Burgess ..... 82

## COLUMNS

### CUTTING EDGE

C# 4.0, the Dynamic Keyword and COM

Dino Esposito page 6

### CLR INSIDE OUT

F# Fundamentals

Luke Hoban page 16

### TEST RUN

Generating Graphs with WPF

James McCaffrey page 92

### BASIC INSTINCTS

Multi-Targeting Visual Basic Applications in Visual Studio 2010

Spotty Bowles page 98

### THE WORKING PROGRAMMER

Going NoSQL with MongoDB, Part 2

Ted Neward page 104

### UI FRONTIERS

The Ins and Outs of ItemsControl

Charles Petzold page 109

### DON'T GET ME STARTED

We're All in This Together

David Platt page 112



This month at  
[msdn.microsoft.com/magazine](http://msdn.microsoft.com/magazine):

### SILVERLIGHT ONLINE

Silverlight in an Occasionally Connected World

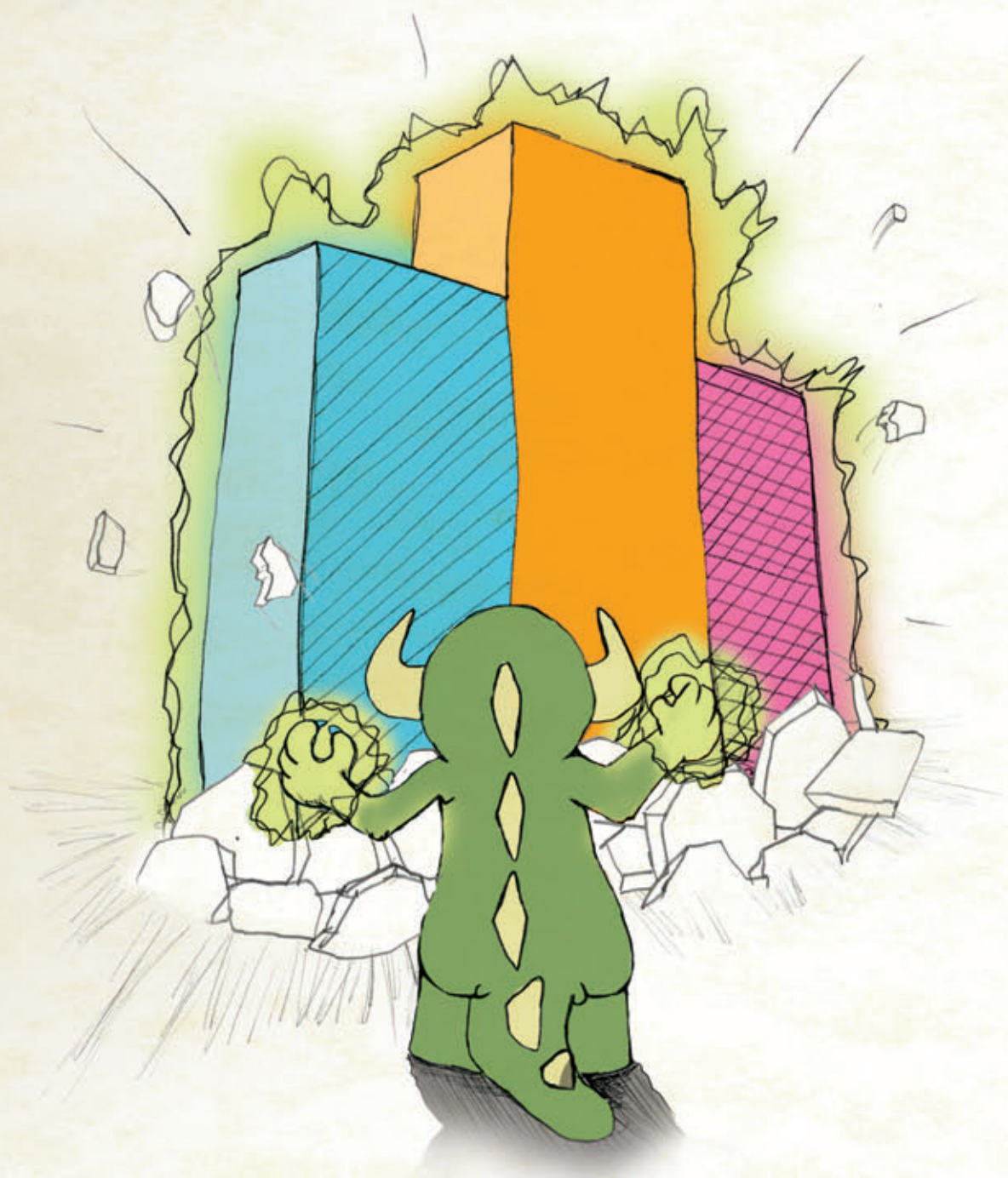
Mark Bloodworth and Dave Brown

### APPFABRIC CACHE

Real-World Usage and Integration

Andrea Colaci

**Microsoft<sup>®</sup>**



# BREAK OUT OF THE BOX

---

Sure, Visual Studio 2010 has a lot of great functionality—we're excited that it's only making our User Interface components even better! We're here to help you go beyond what Visual Studio 2010 gives you so you can create Killer Apps quickly, easily and without breaking a sweat! Go to [infragistics.com/beyondthebox](http://infragistics.com/beyondthebox) today to expand your toolbox with the fastest, best-performing and most powerful UI controls available. You'll be surprised by your own strength!

Infragistics Sales 800 231 8588  
Infragistics Europe Sales +44 (0) 800 298 9055  
Infragistics India +91-80-6785-1111  
[twitter.com/infragistics](https://twitter.com/infragistics)

---





# dtSearch®

## Instantly Search Terabytes of Text



- ◆ 25+ full-text and fielded data search options

- ◆ Built-in file parsers and converters **highlight hits** in popular file types

- ◆ Spider supports static and dynamic web data; **highlights hits** with links, formatting and images intact

- ◆ API supports C++, .NET, Java, SQL, etc. .NET Spider API. Includes 64-bit (Win/Linux)

- ◆ Fully-functional evaluations available

### Content extraction only licenses also available

"Bottom line: dtSearch manages a terabyte of text in a single index and returns results in less than a second" — **InfoWorld**

dtSearch "covers all data sources ... powerful Web-based engines" — **eWEEK**

"Lightning fast ... performance was unmatched by any other product" — **Redmond Magazine**

For hundreds more reviews, and hundreds of developer case studies, see [www.dtSearch.com](http://www.dtSearch.com)

**1-800-IT-FINDS • [www.dtSearch.com](http://www.dtSearch.com)**

**The Smart Choice for Text Retrieval® since 1991**

**msdn®**  
magazine  
JUNE 2010 VOLUME 25 NUMBER 6

**LUCINDA ROWLEY** Director

**DIEGO DAGUM** Editorial Director/[mmeditor@microsoft.com](mailto:mmeditor@microsoft.com)

**KERI GRASSL** Site Manager

**KEITH WARD** Editor in Chief/[mmeditor@microsoft.com](mailto:mmeditor@microsoft.com)

**TERRENCE DORSEY** Technical Editor

**DAVID RAMEL** Features Editor

**WENDY GONCHAR** Managing Editor

**MARTI LONGWORTH** Associate Managing Editor

**SCOTT SHULTZ** Creative Director

**JOSHUA GOULD** Art Director

**ALAN TAO** Senior Graphic Designer

**CONTRIBUTING EDITORS** K. Scott Allen, Dino Esposito, Julie Lerman, Juval Lowy, Dr. James McCaffrey, Ted Neward, Charles Petzold, David S. Platt

### Redmond Media Group

**Henry Allain** President, Redmond Media Group

**Matt Morollo** Vice President, Publishing

**Doug Barney** Vice President, Editorial Director

**Michele Imgrund** Director, Marketing

**Tracy Cook** Online Marketing Director

ADVERTISING SALES: 508-532-1418/[mmorollo@1105media.com](mailto:mmorollo@1105media.com)

**Matt Morollo** VP Publishing

**Chris Kourtoglou** Regional Sales Manager

**William Smith** National Accounts Director

**Danna Vedder** Microsoft Account Manager

**Jenny Hernandez-Asandas** Director Print Production

**Serena Barnes** Production Coordinator/[msdnadproduction@1105media.com](mailto:msdnadproduction@1105media.com)

### 1105 MEDIA

**Neal Vitale** President & Chief Executive Officer

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

**Michael J. Valenti** Executive Vice President

**Abraham M. Langer** Senior Vice President, Audience Development & Digital Media

**Christopher M. Coates** Vice President, Finance & Administration

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

**Carmel McDonagh** Vice President, Attendee Marketing

**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 U.S. funds: U.S. \$35; Canada \$45; International \$60. 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, e-mail [MSDNmag@1105service.com](mailto: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 IMS/NJ. Attn: Returns, 310 Paterson Plank Road, Carlstadt, NJ 07072.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o MSDN Magazine, 16261 Laguna Canyon Road, Ste. 130, Irvine, CA 92618.

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

**Corporate Address:** 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, [www.1105media.com](http://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](mailto: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](mailto:1105reprints@parsintl.com), [www.magreprints.com/QuickQuote.asp](http://www.magreprints.com/QuickQuote.asp)

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

All customer service inquiries should be sent to [MSDNmag@1105service.com](mailto:MSDNmag@1105service.com) or call 847-763-9560.

**Microsoft**

**BPA**  
WORLDWIDE

Printed in the USA



/\*IT'S EVERYWHERE.\*/\*

Code. It's used to create the things that are all around us. Almost everywhere you look, it's there. Just like the possibilities you see as a developer. With Visual Studio 2010, you can realize your vision with new tools that will dramatically impact the way you work, from design to development to deployment.

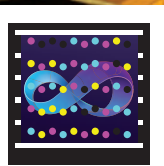
**LIFE RUNS ON CODE.**

WHAT WILL YOU DO WITH VISUAL STUDIO 2010?

Get started at **VisualStudio.com**



SNAP  
IT



[Snap this tag to get the latest news on Visual Studio 2010 or Text VS2010 to 21710\*]

Get the free app for your phone at <http://gettag.mobi>

\*Standard messaging and data charges apply.



# We're All Ears

One of the most important parts of my job as editor in chief is listening to you. As in, You the Readers. This magazine's value is in presenting the kind of information that helps you better do your primary job; for most, that means sharpening your software development skills.

Over the past half-year or so, I've been talking to readers, and, especially, reading your e-mail comments about ways to improve the magazine (keep those comments coming to [mmeditor@microsoft.com](mailto:mmeditor@microsoft.com)). I've gathered enough information from various sources now to get an idea of what many of you are looking for. The next step is to ask you which of these things you'd most like to see in the magazine.

**More coverage of non-C# languages.** Our core coverage, at least for the foreseeable future, will feature C#. Of course, we do have articles that use other languages like F#, where those languages are the most appropriate to use for a given task. And we have regular coverage of Visual Basic through our Basic Instincts column.

A number of readers, however, have stated their desire to have more regular coverage of C# alternatives, particularly C++.

What do you think: Would you like to see a regular column covering C++ programming, or are you fine with the occasional C++, like the one we ran in April on new C++ features in Visual Studio 2010 ([msdn.microsoft.com/magazine/ee336130](http://msdn.microsoft.com/magazine/ee336130))? Or do you feel we have the right mix?

**Articles targeted toward beginning or inexperienced developers.** Numerous readers have expressed their frustration that too much of our content is over their heads, and they'd like to see more coverage of basic development or coding practices.

This is a tricky problem. The reason is that people who are dissatisfied with something are much more likely to speak up than those who aren't. So, even though our coverage is properly targeted toward more-experienced developers, we're hearing more

often from the minority, making that group seem larger than it is. The other possibility, of course, is that the response properly reflects our readers, and we do need more articles dealing with lower-level topics. Again, this is something we cover in the magazine; the question is how often we should dip into that well.

**Existing products/technologies vs. new/future products and technologies.** Do you prefer more coverage of products and technologies you're using day-to-day, or more looking-ahead coverage so you can get an idea of what else is out there that you'd like to try out?

We won't be totally upending the ship if we change a little rigging.

Also, remember that we won't be totally upending the ship if we change a little rigging. What I mean is that any changes we make in our coverage will be gradual and incremental. If we add some more C++ articles, we won't be doing drastically fewer C# stories, for example. This is a fine-tuning process, to make sure this is your must-read magazine each month. I encourage you to take a little time when you can and let us know your feelings on these topics. To paraphrase the inimitable Frasier Crane: We're listening.

One final note: I'll be at Tech-Ed this June, and would love to chat with you in person. Look for me in the 1105 Media booth, and other places. If you see me, please grab me (gently) so we can talk a bit. I hope I get a chance to speak with many of you there. If you're unable to make it, be sure to drop me a line at [mmeditor@microsoft.com](mailto:mmeditor@microsoft.com).

*Keith Ward*

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

© 2010 Microsoft Corporation. All rights reserved.

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

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

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



# Introducing **OnTime 2010** **NEW!** Scrum Planning Board that Changes Everything!



"Best Project Management Software"  
Readers' Choice - 4 years in a row

## Project Management • Bug Tracking • Scrum

Sign up for *OnTime Now* (in the Cloud) or install locally!



**Track Everything**  
Track everything from bugs to software requirements, team member tasks, and more. 360-degree visibility.



**Planning Board**  
The **NEW** planning board makes this both the most powerful & easiest to use version of OnTime **EVER!**



**Tortoise SVN**  
Perfect for Subversion Teams! The new TortoiseSVN plug-in for OnTime 2010 provides tight integration.



**Scrum**  
Harness the power of scrum within the OnTime client. Use Sprints, Burndowns, Trending Charts and more.



**Workflow**  
Define a hyper-fast, lightweight, agile workflow or a robust process-enforcement system.



**Help Desk**  
Use OnTime for tracking Help Desk incidents - even monitor email accounts to generate help tickets.



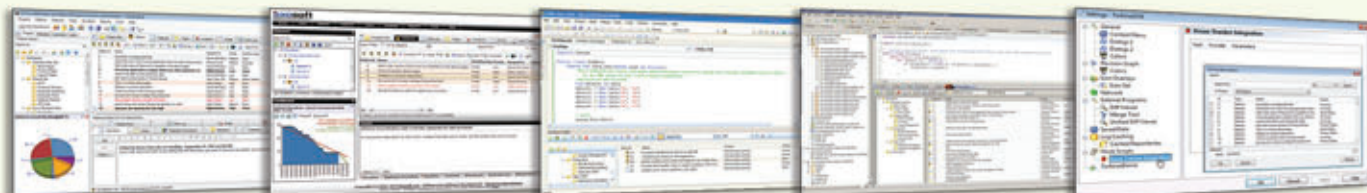
**Notifications**  
Ensure team members are in sync with automatic email notifications and alerts.



**Customize**  
Unlimited custom fields, customize all field names, and decide when fields are visible, editable or required.



**OnTime Now!**  
Cloud project management done right! During signup, speed-test multiple data centers & pick the quickest.



Windows

Web

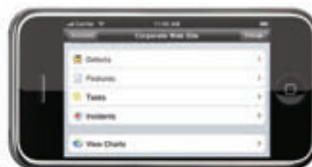
Visual Studio

Eclipse

Sub-Version

**axosoft**

**FREE** 1-User License • **FREE** 30-day Team Trials



800.653.0024  
[www.axosoft.com](http://www.axosoft.com)



# C# 4.0, the Dynamic Keyword and COM

I grew up as a C/C++ developer and, especially before the advent of the Microsoft .NET Framework, I often chided my colleagues who programmed in Visual Basic for using such a weakly typed language.

There was a time when static typing and strongly typed programming were the obvious way to software happiness. But things change, and today the community of C# developers—to which it seems nearly all former C/C++ developers have migrated—often feel the distinct need for a much more dynamic programming model. Last month, I introduced some features of dynamic programming that Microsoft makes available through C# 4.0 and Visual Studio 2010. This month, I'll delve deeper into some related scenarios, starting with one of the most compelling reasons for using C# 4.0—easy programming with COM objects within the .NET Framework.

## Easy Access to COM Objects

An object is said to be dynamic when its structure and behavior aren't fully described by a statically defined type that the compiler knows thoroughly. Admittedly, the word dynamic sounds a bit generic in this context, so let's look at a simple example. In a scripting language such as VBScript, the following code runs successfully:

```
Set word = CreateObject("Word.Application")
```

The `CreateObject` function assumes that the string it gets as an argument is the progID of a registered COM object. It creates an instance of the component and returns its `IDispatch` automation interface. The details of the `IDispatch` interface are never visible at the level of the scripting language. What matters is that you can write code such as:

```
Set word = CreateObject("Word.Application")
word.Visible = True
Set doc = word.Documents.Add()
Set selection = word.Selection
selection.TypeText "Hello, world"
selection.TypeParagraph()
doc.SaveAs(fileName)
```

In this code, you first create a reference to a component that automates the behavior of the underlying Microsoft Office Word application. Next, you make the Word main window visible, add a new document, write some text into it and then save the document somewhere. The code is clear, reads well and, more importantly, works just fine.

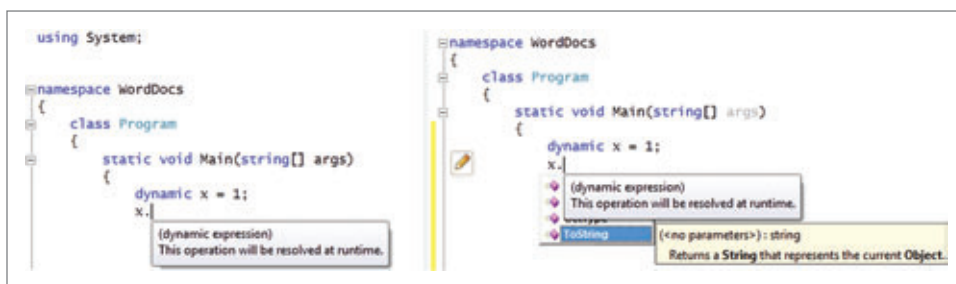


Figure 1 IntelliSense for a Dynamic Object in Visual Studio 2010, with and Without ReSharper

The reason this works, however, is due to a particular capability offered by VBScript—late binding. Late binding means that the type of a given object isn't known until the execution flow hits the object. When this happens, the runtime environment first ensures that the member invoked on the object really exists and then invokes it. No preliminary check whatsoever is made before the code is actually executed.

I often chided my colleagues  
who programmed in Visual Basic  
for using such a weakly  
typed language.

As you may know, a scripting language such as VBScript doesn't have a compiler. However, Visual Basic (including the CLR version) for years had a similar feature. I confess I frequently envied my Visual Basic colleagues for their ability to more easily use COM objects—often valuable building blocks of an application you need to interop with, such as Office. In some cases, in fact, my team ended up writing some portions of our interop code in Visual Basic, even when the entire application was in C#. Should this be surprising? Isn't polyglot programming a new frontier to reach?

In Visual Basic, the `CreateObject` function exists for (strong) compatibility reasons. The point is that .NET Framework-based languages were designed with early binding in mind. COM interoperability is a scenario addressed by the .NET Framework

Code download available at [code.msdn.microsoft.com/mag201006CutEdge](http://code.msdn.microsoft.com/mag201006CutEdge).



# Your best source for software development tools!

# programmer's paradise®



**NEW RELEASE!**

Paradise #  
105 26301A01

**\$3,214.99**

## LEADTOOLS Recognition SDK

by LEAD Technologies

Develop robust 32/64 bit document imaging and recognition functionality into your applications with accurate and high-speed multi-threaded Forms, OCR, OMR, and 1D/2D barcode engines.

- Supports text, OMR, image, and barcode fields
- Auto-registration and clean-up to improve recognition results
- Provided as both high and low level interface
- Includes comprehensive confidence reports to assess performance

[programmers.com/LEAD](http://programmers.com/LEAD)

## Multi-Edit<sup>x</sup>

by Multi Edit Software

Multi-Edit<sup>x</sup> is "The Solution" for your editing needs with support for over 50 languages. Edit plain text, ANY Unicode, hex, XML, HTML, PHP, Java, Javascript, Perl and more! No more file size limitations, unlimited line length, any file, any size Multi-Edit<sup>x</sup> is "The Solution"!

**Pre-Order Your Copy and Save!**

**NEW RELEASE!**



1-49 Users  
Paradise #  
A30Z10101A01

**\$223.20**

[programmers.com/multiedit](http://programmers.com/multiedit)

## VMware vSphere

Put time back into your day.

Your business depends on how you spend your time. You need to manage IT costs without losing time or performance. With proven cost-effective virtualization solutions from VMware, you can:

- Increase the productivity of your existing staff three times over
  - Control downtime—whether planned or not
  - Save more than 50% on the cost of managing, powering and cooling servers
- Make your time (and money) count for more with virtualization from VMware.



VMware  
Advanced  
Acceleration Kit  
for 6 processors  
Paradise #  
V55 78101A01

**\$9,234.99**

[programmers.com/vSphere](http://programmers.com/vSphere)



**Certified for Windows 7/2008R2**

Paradise #  
P35 04201A01

**\$550.99**

## Pragma Fortress SSH—SSH Server & Client for Windows

by Pragma Systems

Contains SSH, SFTP, SCP servers and clients for Windows.

- Certified for Windows Server 2008R2
- Compatible with Windows 7
- High-performance servers with centralized management
- Active Directory & GSSAPI authentication
- Supports over 1000 sessions
- Offers FIPS mode
- Hyper-V and PowerShell support
- Runs in Windows 2008R2/2008/2003/7/Vista/XP/2000

[programmers.com/pragma](http://programmers.com/pragma)

## AdminStudio & Application Virtualization Pack

by Flexera Software

One Application Software Deployment Tool for Reliable MSI Packaging, Application Virtualization, and Windows 7 Migration. Top choice of Microsoft®, Novell®, LANDesk® and other software management solutions. Cut MSI packaging time by up to 70%, Deploy software to desktops with 99% success or better. AdminStudio is the only MSI packaging solution to support multiple virtualization formats, including Microsoft® App-V™, VMware® ThinApp™ and Citrix® XenApp™.

[programmers.com/flexera](http://programmers.com/flexera)



Professional  
Upgrade from  
any Active AS  
Pro + Silver Mtn  
Paradise #  
I21 09401S05

**\$4,228.99**

## BUILD ON VMWARE ESXi AND VSPHERE

for Centralized Management, Continuous Application Availability, and Maximum Operational Efficiency in Your Virtualized Datacenter.

Programmer's Paradise invites you to take advantage of this webinar series sponsored by our TechXtend solutions division.

**FREE VIRTUALIZATION WEBINAR SERIES: REGISTER TODAY! TechXtend.com/Webinars**



**NEW VERSION 6!**

Professional Ed.  
Paradise #  
D03 04301A01

**\$1,310.99**

## ActiveReports 6

by GrapeCity

Integrate Business Intelligence/Reporting/Data Analysis into your .NET applications using the NEW ActiveReports 6.

- Fast and Flexible reporting engine
- Data Visualization and Layout Controls such as Chart, Barcode and Table Cross Section Controls
- Wide range of Export and Preview formats including Windows Forms Viewer, Web Viewer, Adobe Flash and PDF
- Royalty-Free Licensing for Web and Windows applications

[programmers.com/grapecity](http://programmers.com/grapecity)

## CA ERwin® Data Modeler r7.3 – Product Plus 1 Year Enterprise Maintenance

by CA

CA ERwin Data Modeler is a data modeling solution that enables you to create and maintain databases, data warehouses and enterprise data resource models. These models help you visualize data structures so that you can effectively organize, manage and moderate data complexities, database technologies and the deployment environment.



Paradise #  
P26 04201E01

**\$3,919.99**

[programmers.com/ca](http://programmers.com/ca)

## TX Text Control 15.1

Word Processing Components

TX Text Control is royalty-free, robust and powerful word processing software in reusable component form.

- .NET WinForms control for VB.NET and C#
- ActiveX for VB6, Delphi, VBScript/HTML, ASP
- File formats DOCX, DOC, RTF, HTML, XML, TXT
- PDF and PDF/A export, PDF text import
- Tables, headers & footers, text frames, bullets, structured numbered lists, multiple undo/redo, sections, merge fields, columns
- Ready-to-use toolbars and dialog boxes

**NEW RELEASE!**



Professional Edition  
Paradise #  
T79 02101A02

**\$848.99**

**Download a demo today.**

[programmers.com/theimagingsource](http://programmers.com/theimagingsource)

## SAVE 25% ON Toad® Extension for Visual Studio 2010\*



**Save 25%!**

### EXCLUSIVE OFFER FOR MSDN SUBSCRIBERS!

#### Extend Your Visual Studio Development to Oracle

Toad Extension for Visual Studio is an Oracle database schema provider that goes beyond online development to offer you:

- Easier Oracle development in Visual Studio 2010
- Assure application code is synchronized with the database
- Oracle change management for application teams
- Team collaboration and Microsoft's ALM methodology

Paradise #  
Q13ZM6101A01

List Price:  
**\$799.00**

Your Price:  
**\$599.25**

[programmers.com/toad\\_VS2010](http://programmers.com/toad_VS2010)

## SPECIAL PROMOTION SAVE 20%



### ComponentOne Studio® Enterprise

The world's most complete VS2010-ready component suite for developing all layers of Windows, Web, and Mobile applications.

Studio Enterprise  
Paradise #  
C18 01101A07  
**\$1,040.00**

Studio Enterprise  
w/Platinum Support  
Paradise #  
C18 01101E02  
**\$1,280.00**

[programmers.com/componentone](http://programmers.com/componentone)



\* Exclusive to MSDN subscribers — from now until 31st July, 2010

# 866-719-1528

# programmersparadise.com

Prices subject to change. Not responsible for typographical errors.

Figure 2 The Real Implementation of a Dynamic Variable

```
internal class Program
{
    private static void Main(string[] args)
    {
        object x = 1;

        if (MainSiteContainer.site1 == null)
        {
            MainSiteContainer.site1 = CallSite<
                Action<CallSite, Type, object>>
                .Create(Binder.InvokeMember(
                    "WriteLine",
                    null,
                    typeof(Program),
                    new CSharpArgumentInfo[] {
                        CSharpArgumentInfo.Create(...)
                    }));
        }
        MainSiteContainer.site1.Target.Invoke(
            site1, typeof(Console), x);
    }

    private static class MainSiteContainer
    {
        public static CallSite<Action<CallSite, Type, object>> site1;
    }
}
```

but never specifically supported by languages with keywords and facilities—not until C# 4.0.

C# 4.0 (and Visual Basic) has dynamic lookup capabilities that indicate late binding is now an approved practice for .NET Framework developers. With dynamic lookup, you can code access to methods, properties, indexer properties and fields in a way that bypasses static type checking to be resolved at run time.

C# 4.0 also enables optional parameters by recognizing default value in a member declaration. This means that when a member with optional parameters is invoked, optional arguments can be omitted. Furthermore, arguments can be passed by name as well as by position. At the end of the day, improved COM binding in C# 4.0

simply means that some common features of scripting languages are now supported by an otherwise static and strongly typed language. Before we look at how you can leverage the new dynamic keyword to operate seamlessly with COM objects, let's delve a bit deeper into the internal mechanics of dynamic type lookup.

## Dynamic Language Runtime

When you declare a variable as dynamic in Visual Studio 2010, you have no IntelliSense at all in the default configuration. Interestingly, if you install an additional tool such as ReSharper 5.0 ([jetbrains.com/resharper](http://jetbrains.com/resharper)), you can get some partial information through IntelliSense about the dynamic object. **Figure 1** shows the code editor with and without ReSharper. The tool just lists the members that appear to be defined on the dynamic type. At the very minimum, the dynamic object is an instance of `System.Object`.

Let's see what happens when the compiler encounters the following code (the code is deliberately trivial to simplify understanding the implementation details):

```
class Program
{
    static void Main(string[] args)
    {
        dynamic x = 1;
        Console.WriteLine(x);
    }
}
```

In the second line, the compiler doesn't attempt to resolve the symbol `WriteLine`, and no warning or error is thrown as would happen with a classic static type checker. As far as the dynamic keyword is concerned, C# is like an interpreted language here. Consequently, the compiler emits some ad hoc code that interprets the expression where a dynamic variable or argument is involved. The interpreter is based on the Dynamic Language Runtime (DLR), a brand-new component of the .NET Framework machinery. To use more specific terminology, the compiler has to generate an expression tree using the abstract syntax supported by the

DLR and pass it to the DLR libraries for processing. Within the DLR, the compiler-provided expression is encapsulated in a dynamically updated site object. A site object is responsible for binding methods to objects on the fly. **Figure 2** shows a largely sanitized version of the real code emitted for the trivial program shown earlier.

The code in **Figure 2** has been edited and simplified for readability, but it shows the gist of what's going on. The dynamic variable is mapped to a `System.Object` instance and then a site is created for the program in the DLR. The site manages a binding between the `WriteLine` method with its parameters and the target object. The binding holds within the context of the type `Program`. To invoke the method `Console.WriteLine` on a dynamic variable, you invoke the site and pass the

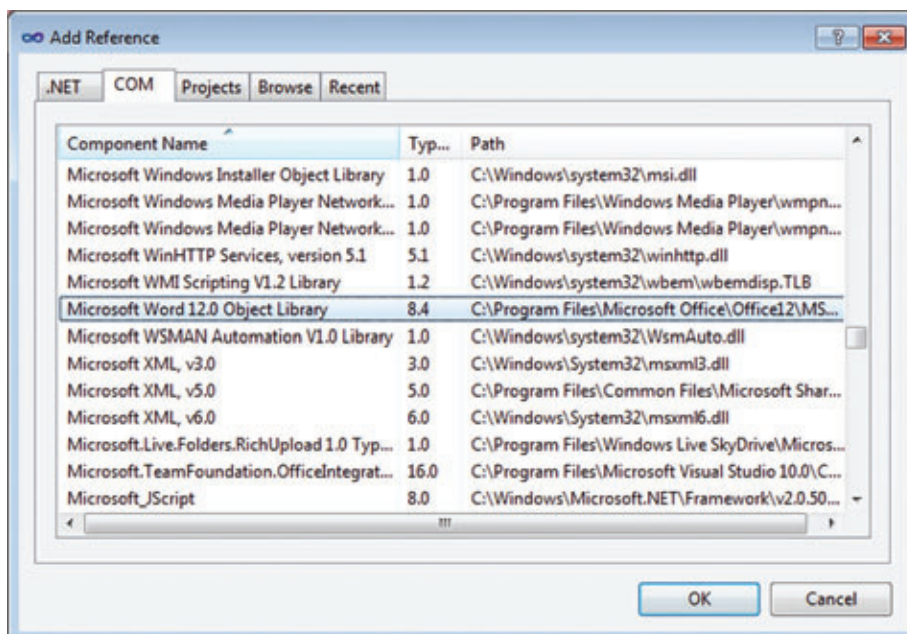


Figure 3 Referencing the Word Object Library





# The XtraReports Suite

for Winforms and ASP.NET

## The Report Viewer

for WPF and Silverlight

Award-Winning Development Tools by DevExpress

Learn more and download your free evaluation copy today

Visit [DevExpress.com/XtraReports](http://DevExpress.com/XtraReports)



Figure 4 Creating a New Word Document in C# 3.0

```
public static class WordDocument
{
    public const String TemplateName = @"Sample.dotx";
    public const String CurrentDateBookmark = "CurrentDate";
    public const String SignatureBookmark = "Signature";

    public static void Create(String file, DateTime now, String author)
    {
        // Must be an Object because it is passed as a ref
        Object missingValue = Missing.Value;

        // Run Word and make it visible for demo purposes
        var wordApp = new Application { Visible = true };

        // Create a new document
        Object template = TemplateName;
        var doc = wordApp.Documents.Add(ref template,
            ref missingValue, ref missingValue, ref missingValue);
        doc.Activate();

        // Fill up placeholders in the document
        Object bookmark_CurrentDate = CurrentDateBookmark;
        Object bookmark_Signature = SignatureBookmark;
        doc.Bookmarks.get_Item(ref bookmark_CurrentDate).Range.Select();
        wordApp.Selection.TypeText(current.ToString());
        doc.Bookmarks.get_Item(ref bookmark_Signature).Range.Select();
        wordApp.Selection.TypeText(author);

        // Save the document
        Object documentName = file;
        doc.SaveAs(ref documentName,
            ref missingValue, ref missingValue, ref missingValue,
            ref missingValue, ref missingValue, ref missingValue,
            ref missingValue, ref missingValue, ref missingValue,
            ref missingValue, ref missingValue, ref missingValue);

        doc.Close(ref missingValue,
            ref missingValue, ref missingValue);
        wordApp.Quit(ref missingValue,
            ref missingValue, ref missingValue);
    }
}
```

target object (in this case the Console type) and its parameters (in this case the dynamic variable). Internally, the site will check whether the target object really has a member WriteLine that can accept a parameter like the object currently stored in the variable x. If something goes wrong, the C# runtime just throws RuntimeBinderException.

My team also ended up writing some portions of our interop code in Visual Basic even when the entire application was in C#.

## Working with COM Objects

New C# 4.0 features working with COM objects from within .NET Framework-based applications considerably easier today. Let's see how to create a Word document in C# and compare the code you need in .NET 3.5 and .NET 4. The sample application creates a new Word document based on a given template, fills it up and saves it to a fixed location. The template contains a couple of bookmarks

for common pieces of information. Whether you target the .NET Framework 3.5 or the .NET Framework 4, the very first step on the way to programmatically creating a Word document is adding the Microsoft Word Object Library (see Figure 3).

Before Visual Studio 2010 and the .NET Framework 4, to accomplish this you needed code such as that in Figure 4.

.NET Framework-based languages were designed with early binding in mind.

To interact with a COM automation interface, you often need Variant types. When you interact with a COM automation object from within a .NET Framework-based application, you represent Variants as plain objects. The net effect is that you can't use a string to indicate, say, the name of the template file you intend to base your Word document on, because the Variant parameter must be passed by reference. You have to resort to an Object instead, as shown here:

```
Object template = TemplateName;
var doc = wordApp.Documents.Add(ref template,
    ref missingValue, ref missingValue, ref missingValue);
```

A second aspect to consider is that Visual Basic and scripting languages are much more forgiving than C# 3.0. So, for example, they don't force you to specify all parameters that a method on a COM object declares. The Add method on the Documents collection requires four arguments, and you can't ignore them unless your language supports optional parameters.

As mentioned earlier, C# 4.0 does support optional parameters. This means that while simply recompiling the code in Figure 4 with

Figure 5 Creating a New Word Document in C# 4.0

```
public static class WordDocument
{
    public const String TemplateName = @"Sample.dotx";
    public const String CurrentDateBookmark = "CurrentDate";
    public const String SignatureBookmark = "Signature";

    public static void Create(string file, DateTime now, String author)
    {
        // Run Word and make it visible for demo purposes
        dynamic wordApp = new Application { Visible = true };

        // Create a new document
        var doc = wordApp.Documents.Add(TemplateName);
        templatedDocument.Activate();

        // Fill the bookmarks in the document
        doc.Bookmarks[CurrentDateBookmark].Range.Select();
        wordApp.Selection.TypeText(current.ToString());
        doc.Bookmarks[SignatureBookmark].Range.Select();
        wordApp.Selection.TypeText(author);

        // Save the document
        doc.SaveAs(fileName);

        // Clean up
        templatedDocument.Close();
        wordApp.Quit();
    }
}
```





C# 4.0 works, you could even rewrite it and drop all ref parameters that carry only a missing value, as shown here:

```
Object template = TemplateName;  
var doc = wordApp.Documents.Add(template);
```

With the new C# 4.0 “Omit ref” support, the code in **Figure 4** becomes even simpler and, more importantly, it becomes easier to read and syntactically similar to scripting code. **Figure 5** contains the

edited version that compiles well with C# 4.0 and produces the same effect as the code in **Figure 4**.

The code in **Figure 5** allows you to use plain .NET Framework types to make the call to the COM object. Plus, optional parameters make it even simpler.

The dynamic keyword and other COM interop features introduced in C# 4.0 don't make a piece of code necessarily faster, but it enables you to write C# code as if it were script. For COM objects, this achievement is probably as important as an increment of performance.

To interact with a COM  
automation interface, you  
need Variant types.

### No PIA Deployment

Since the beginning of the .NET Framework, you could wrap a COM object into a managed class and use it from a .NET-based application. For this to happen, you need to use a primary interop assembly (PIA) provided by the vendor of the COM object. PIAs are necessary and must be deployed along with client applications. However, more often than not, PIAs are too big and wrap up an entire COM API, so packing them with the setup may not be a pleasant experience.

Visual Studio 2010 offers the no-PIA option. No-PIA refers to the compiler's ability to embed required definitions you'd get from a PIA in the current assembly. As a result, only definitions that are really needed are found in the final assembly and there's no need for you to pack vendor's PIAs in your setup. **Figure 6** shows the option in the Properties box that enables no-PIA in Visual Studio 2010.

No-PIA is based on a feature of C# 4.0 known as type equivalence. In brief, type equivalence means that two distinct types can be considered equivalent at run time and used interchangeably. The typical example of type equivalence is two interfaces with the same name defined in different assemblies. They're different types, but they can be used interchangeably as long as the same methods exist.

In summary, working with COM objects can still be expensive, but the COM interop support in C# 4.0 makes the code you write far simpler. Dealing with COM objects from .NET Framework-based applications connects you to legacy applications and critical business scenarios over which you'd otherwise have little control. COM is a necessary evil in the .NET Framework, but dynamic makes it a bit less so. ■

**DINO ESPOSITO** is the author of “Programming ASP.NET MVC” from Microsoft Press and has coauthored “Microsoft .NET: Architecting Applications for the Enterprise” (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. You can join his blog at [weblogs.asp.net/despos](http://weblogs.asp.net/despos).

**THANKS** to the following technical expert for reviewing this article:  
Alex Turner

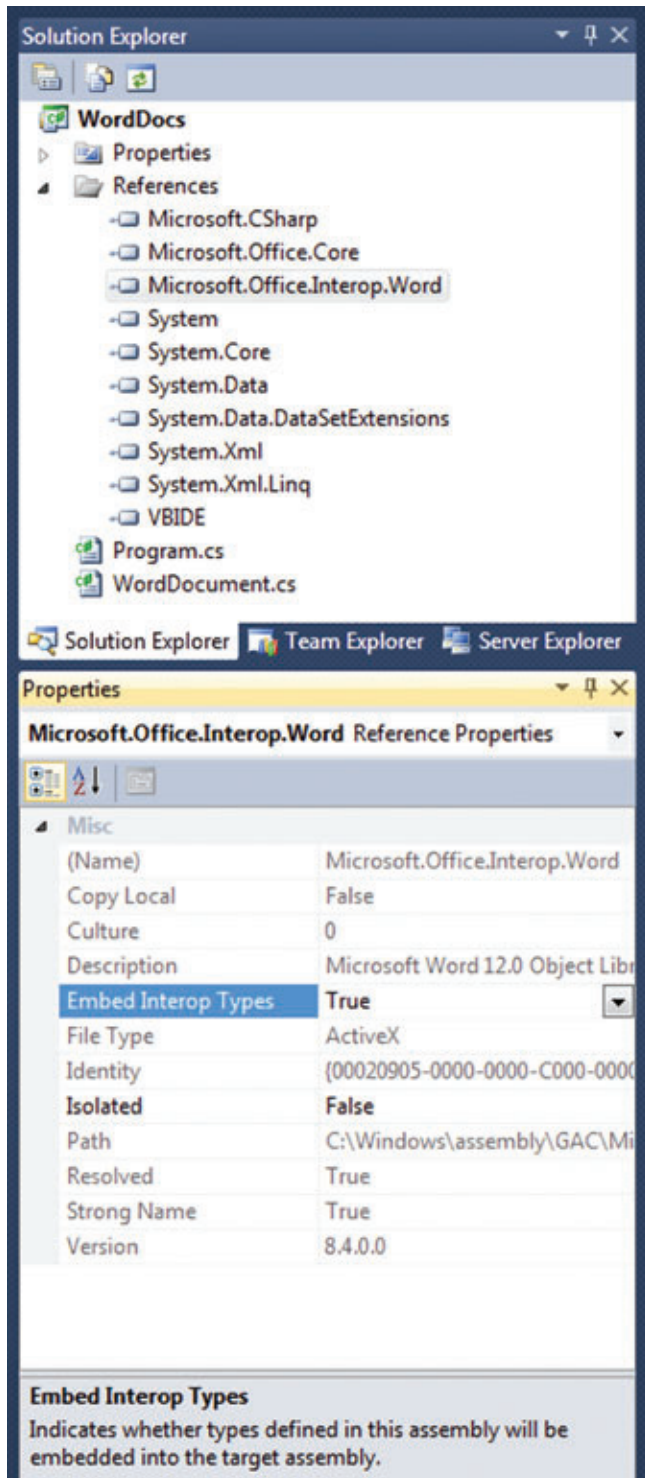


Figure 6 Enabling the No-PIA Option in Visual Studio 2010





**The DXPivotGrid, XtraPivotGrid,  
and ASPxPivotGrid Suites**  
for WPF, Winforms and ASP.NET  
Award-Winning Development Tools by DevExpress  
Learn more and download your free evaluation copy today  
Visit **DevExpress.com/XtraGrid**



# RadControls *for* Silverlight

Presenting the industry leading UI components for Silverlight with unmatched performance and pioneering support for Silverlight 4.

## **Pioneering Support for Microsoft Silverlight 4**

Telerik is the first component vendor to provide native controls built on Silverlight 4. RadControls for Silverlight 4 fully match the feature set of their Silverlight 3 counterparts, and closely follow Microsoft's latest advancements in the Silverlight 4 framework, staying up to date with the latest beta releases and the fast-paced Microsoft release schedule for this technology.

## **Engineered for Great Performance**

Telerik Silverlight controls are engineered for outstanding performance by utilizing various techniques that help reduce page loading time and speed up data operations such as streamlined themes and templates, virtualized scrolling, innovative LINQ-based data engine, built-in support for asynchronous databinding, RadCompression module and more.

## **Support for Visual Studio 2010 and Expression Blend**

RadControls for Silverlight provide support for Visual Studio 2010 Beta 2, offering toolbox support, property browsing and WYSIWYG preview for all controls. Telerik is working closely with Microsoft to ensure best practices are followed and provide the most complete design experience, allowing for easy development in Visual Studio 2010 and styling in Expression Blend.

## **A Comprehensive Silverlight Toolset from the Masters of Web UI**

An established leader in web interface technologies, Telerik offers a comprehensive suite of 40+ controls that bring style and interactivity to your line-of-business applications. Featuring a lightning fast DataGrid, rich data visualization controls, and a powerful Outlook-like Scheduling control, RadControls provides all the building blocks for developing next generation Rich Internet Applications (RIAs).

## **Code Re-Use with RadControls for WPF**

RadControls for Silverlight and RadControls for WPF are derived from the same codebase and share the same API. They represent two almost mirror toolsets for building rich line-of-business web and desktop applications, allowing for substantial code and skills reuse between Silverlight and WPF development and shortening your learning curve.





 **telerik**  
deliver more than expected

## F# Fundamentals

F# is a new, functional and object-oriented programming language for the Microsoft .NET Framework, and it's integrated into this year's release of Microsoft Visual Studio 2010. F# combines simple, succinct syntax with strong static typing, and it scales from lightweight explorative programming in the F# Interactive up to large-scale .NET Framework-based component development with Visual Studio.

F# is designed from the ground up to run on the CLR. As a .NET Framework-based language, F# leverages the rich libraries available on the .NET Framework platform, and can be used to build .NET libraries or implement .NET interfaces. F# also takes advantage of many of the CLR core features, including generics, garbage collection, tail call instructions and the fundamental Common Language Infrastructure (CLI) type system.

This article takes a look at some of the core concepts of the F# language and its implementation on top of the CLR.

### A Quick Look at F#

Let's start with a brief look at a number of the core language features in F#. For more details on any of these features and the many other interesting concepts in the F# language, see the documentation available via the F# Developer Center at [fsharp.net](http://fsharp.net).

The most fundamental feature of F# is the `let` keyword, which binds a value to a name. `let` can be used to bind both data and function values, and for both top-level and local bindings:

```
let data = 12

let f x =
    let sum = x + 1
    let g y = sum + y*y
    g x
```

F# provides a few core datatypes and a language syntax for working with structured data, including lists, typed optional values and tuples:

```
let list1 = ["Bob"; "Jom"]

let option1 = Some("Bob")
let option2 = None

let tuple1 = (1, "one", '1')
```

These pieces of structured data, and others, can be matched against by using F# pattern matching expressions. Pattern matching is similar to using `switch` statements in C-like languages, but provides a richer way to both match and extract parts out of matched expressions, somewhat akin to the way regular expressions are used for pattern-matching strings:

```
let person = Some ("Bob", 32)

match person with
| Some(name,age) -> printfn "We got %s, age %d" name age
| None -> printfn "Nope, got nobody"
```

F# leverages the .NET Framework libraries for many tasks, such as accessing data from a rich variety of data sources. .NET libraries can be used from F# in the same way they are used in other .NET languages:

```
let http url =
    let req = WebRequest.Create(new Uri(url))
    let resp = req.GetResponse()
    let stream = resp.GetResponseStream()
    let reader = new StreamReader(stream)
    reader.ReadToEnd()
```

F# is also an object-oriented language and can define any .NET class or struct, similar to C# or Visual Basic:

```
type Point2D(x,y) =
    member this.X = x
    member this.Y = y
    member this.Magnitude =
        x*x + y*y
    member this.Translate(dx, dy) =
        new Point2D(x + dx, y + dy)
```

F# is designed from the ground up to run on the CLR.

In addition, F# supports two special kinds of types: records and discriminated unions. Records provide a simple representation of data values with named fields, and discriminated unions are an expressive way to represent types that can have a number of different kinds of values, with different associated data in each kind:

```
type Person =
    { Name : string;
      HomeTown : string;
      BirthDate : System.DateTime }

type Tree =
    | Branch of Tree * Tree
    | Leaf of int
```

### F# on the CLR

F# is in many ways a higher-level language than C#, with its typesystem, syntax and language constructs being further away from the metadata and intermediate language (IL) of the CLR. This has a few interesting implications. Most importantly, it means F# developers can often solve problems and think about their programs at a higher level,

Post your questions and comments on the CLR Team blog at [blogs.msdn.com/clrteam](http://blogs.msdn.com/clrteam).



# ESRI® Developer Network

Integrate Mapping and GIS into Your Applications

Visit us at  
Microsoft TechEd booth #2044.



Give your users an effective way to visualize and analyze their data so they can make more informed decisions and solve business problems.

By subscribing to the ESRI® Developer Network (EDN™), you have access to the complete ESRI geographic information system (GIS) software suite for developing and testing applications on every platform. Whether you're a desktop, mobile, server, or Web developer, EDN provides the tools you need to quickly and cost-effectively integrate mapping and GIS into your applications.

Subscribe to EDN and leverage the power of GIS to get more from your data. Visit [www.esri.com/edn](http://www.esri.com/edn).



ESRI®

closer to the domain of the problem at hand. But it also means the F# compiler does more work in mapping F# code onto the CLR, and that the mapping is less direct.

The C# 1.0 compiler and the CLR were developed at the same time, and the features of both were closely aligned. Almost all C# 1.0 language constructs have a very direct representation in the CLR type system and in CIL. This has become less true in later C# releases as the C# language evolved faster than the CLR itself. Iterators and anonymous methods were fundamental C# 2.0 language features that didn't have direct CLR equivalents. In C# 3.0, query expressions and anonymous types followed this trend.

## F# supports two special kinds of types: records and discriminated unions.

F# takes this a step further. Many of the language constructs don't have direct IL equivalents, so features like pattern matching expressions get compiled into a rich set of IL instructions used to accomplish the pattern matching efficiently. F# types such as records and unions automatically generate many of the members needed.

Note, however, that I'm discussing the compilation techniques used by the current F# compiler. Many of these implementation details are not directly visible to the F# developer and could be modified in future versions of the F# compiler for performance optimizations or to enable new features.

### Immutable By Default

The basic `let` binding in F# is similar to `var` in C#, except for one very important difference: you can't change the value of a `let`-bound name later. That is, values are immutable by default in F#:

```
let x = 5
x <- 6 // error: This value is not mutable
```

Immutability has big benefits for concurrency because there is no need to worry about locking when using immutable state—it can be safely accessed from multiple threads. Immutability also tends to decrease coupling between components. The only way for one component to influence another is to make an explicit call to the components.

Mutability can be opted into in F#, and is often used when calling other .NET libraries, or to optimize particular code paths:

```
let mutable y = 5
y <- 6
```

Similarly, types in F# are immutable by default:

```
let bob = { Name = "Bob";
            HomeTown = "Seattle" }
// error: This field is not mutable
bob.HomeTown <- "New York"
```

```
let bobJr = { bob with HomeTown = "Seattle" }
```

In this example, when mutation is not available, it's common to instead use copy-and-update to make a new copy from an old one while changing one or more fields. Although a new object is created, it shares many pieces with the original. In this example,

only a single string—"Bob"—is needed. This sharing is an important part of the performance of immutability.

Sharing can also be seen in F# collections. For example, the F# list type is a linked-list data structure that can share a tail with other lists.

```
let list1 = [1;2;3]
let list2 = 0 :: list1
let list3 = List.tail list1
```

Because of the copy-and-update and sharing inherent in programming with immutable objects, the performance profile of these programs is often quite different from typical imperative programs.

The CLR plays a big role here. Immutable programming tends to create more short-lived objects as a result of transforming data rather than changing it in place. The CLR garbage collector (GC) deals well with these. Short-lived small objects are relatively very cheap due to the generational mark-and-sweep used by the CLR GC.

### Functions

F# is a functional language and, not surprisingly, functions play an important role throughout the language. Functions are a first-class part of the F# type system. For example, the type `char -> int` represents F# functions that take a `char` and return an `int`.

Although similar to .NET delegates, F# functions have two important differences. First, they're not nominal. Any function that takes a `char` and returns an `int` is of type `char -> int`, whereas multiple differently named delegates may be used to represent functions of this signature, and are not interchangeable.

Second, F# functions are designed to efficiently support either partial or full application. Partial application is when a function with multiple parameters is given only a subset of the parameters, thus resulting in a new function that takes the remaining parameters.

```
let add x y = x + y
```

```
let add3a = add 3
let add3b y = add 3 y
let add3c = fun y -> add 3 y
```

All first-class F# function values are instances of a type `FSharpFunc<T, 'Result>`, as defined in the F# runtime library, `FSharp.Core.dll`. When using an F# library from C#, this is the type that all F# function values taken as parameters or returned from methods will have. This class looks roughly like the following (if you were defining it in C#):

```
public abstract class FSharpFunc<T, TResult> {
    public abstract TResult Invoke(T arg);
}
```

Note in particular that all F# functions fundamentally take a single argument and produce a single result. This captures the concept of partial application—an F# function with multiple parameters will actually be an instance of a type like:

```
FSharpFunc<int, FSharpFunc<char, bool>>
```

That is, a function that takes an `int` and returns another function, which itself takes a `char` and returns a `bool`. The common case of full application is made fast by using a set of helper types in the F# core library.

When an F# function value is created using a lambda expression (the `fun` keyword), or as a result of a partial application of another function (as in the `add3a` case shown earlier), the F# compiler generates a closure class:

```
internal class Add3Closure : FSharpFunc<int, int> {
    public override int Invoke(int arg) {
        return arg + 3;
    }
}
```



These closures are similar to closures created by the C# and Visual Basic compilers for their lambda expression constructs. Closures are one of the most common compiler-generated constructs on the .NET Framework platform that do not have direct CLR-level support. They exist in almost all .NET programming languages and are used especially heavily by F#.

Function objects are common in F#, so the F# compiler uses many optimization techniques to avoid the need to allocate these closures. Using inlining, lambda-lifting, and direct representation as .NET methods when possible, the internal code generated by the F# compiler will often look somewhat different than described here.

## Type Inference and Generics

One notable feature of all the code examples so far is the lack of any type annotation. Although F# is a statically typed programming language, explicit type annotations are often not needed because F# makes extensive use of type inference.

Type inference will be familiar to C# and Visual Basic developers who use it for local variables, as in this C# 3.0 code:

```
var name = "John";
```

The `let` keyword in F# is similar, but type inference in F# goes substantially further, applying also to fields, parameters and return types. In the following example, the two fields `x` and `y` are inferred to have type `int`, which is the default for the `+` and `*` operators used on these values within the body of the type definition. The `Translate` method is inferred to have type `"Translate : int * int -> Point2D"`:

```
type Point2D(x,y) =  
    member this.X = x  
    member this.Y = y  
    member this.Magnitude =  
        x*x + y*y  
    member this.Translate(dx, dy) =  
        new Point2D(x + dx, y + dy)
```

Of course, type annotations can be used when needed or desired to tell the F# compiler what type is really expected for a certain value, field or parameter. This information will then be used for type inference. For example, you can change the definition of `Point2D` to use `float` instead of `int` by adding just a couple of type annotations:

```
type Point2D(x : float,y : float) =  
    member this.X = x  
    member this.Y = y  
    member this.Magnitude =  
        x*x + y*y  
    member this.Translate(dx, dy) =  
        new Point2D(x + dx, y + dy)
```

F# is in many ways a higher-level language than C#.

One of the important results of type inference is that functions not tied to a specific type are automatically generalized to be generic functions. So your code will become as generic as possible without you needing to explicitly specify all the generic types. This causes generics to play a fundamental role in F#. The compositional style of functional programming with F# also encourages small

reusable pieces of functionality, which benefit greatly from being as generic as possible. The ability to author generic functions without the complex type annotations is an important feature of F#.

For example, the following `map` function walks a list of values and generates a new list by applying its argument function `f` to each element:

```
let rec map f values =  
    match values with  
    | [] -> []  
    | x :: rest -> (f x) :: (map f rest)
```

Note that there are no type annotations needed, but the type inferred for `map` is `"map : ('a -> 'b) -> list<'a> -> list<'b>".` F# is able to infer from the use of pattern matching, and from the use of the parameter `f` as a function, that the types of the two parameters have a certain shape, but are not completely fixed. So F# makes the function as generic as possible while still having the types needed by the implementation. Note that generic parameters in F# are indicated using a leading `'` character, to distinguish them syntactically from other names.

Every piece of F# code gets an inferred type that's encoded in the CLR metadata.

Don Syme, the designer of F#, was previously the lead researcher and developer on the implementation of generics in the .NET Framework 2.0. The concept of a language like F# critically depends on having generics in the runtime, and Syme's interest in doing F# came in part from wanting to really take advantage of this CLR feature. F# leverages .NET generics heavily; for example, the implementation of the F# compiler itself has more than 9,000 generic type parameters.

Ultimately, type inference is just a compile-time feature, though, and every piece of F# code gets an inferred type that's encoded in the CLR metadata for an F# assembly.

## Tail Calls

Immutability and functional programming tend to encourage the use of recursion as a computational tool in F#. For example, an F# list can be walked and the sum of the squares of the values in the list collected using a simple piece of recursive F# code:

```
let rec sumOfSquares nums =  
    match nums with  
    | [] -> 0  
    | n :: rest -> (n*n) + sumOfSquares rest
```

While recursion is often convenient, it can use a lot of space on the call stack because each iteration adds a new stack frame. For sufficiently large inputs this can even lead to stack-overflow exceptions. To avoid this stack growth, recursive code can be written tail-recursively, meaning that recursive calls are always the last thing done, just before the function returns:

```
let rec sumOfSquaresAcc nums acc =  
    match nums with  
    | [] -> acc  
    | n :: rest -> sumOfSquaresAcc rest (acc + n*n)
```

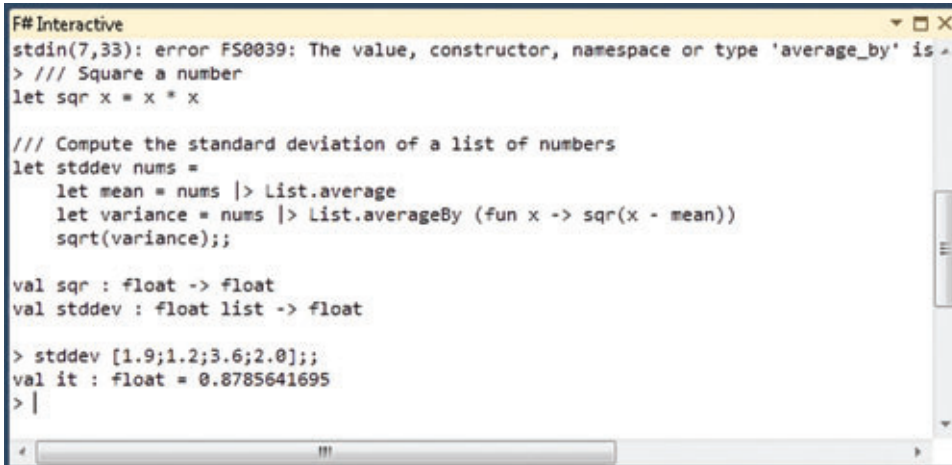


Figure 1 Executing Code in F# Interactive

The F# compiler implements tail-recursive functions using two techniques that aim to ensure the stack will not grow. For direct tail calls to the same function being defined, such as the call to `sumOfSquaresAcc`, the F# compiler automatically converts the recursive call into a while loop, thus avoiding making any call at all, and generating code very similar to an imperative implementation of the same function.

Tail recursion is not always as simple as this, though, and can instead be a result of multiple mutually recursive functions. In this case, the F# compiler relies on the CLR native support for tail calls.

The CLR has an IL instruction specifically to help with tail recursion: the `tail.` IL prefix. The `tail.` instruction tells the CLR it can discard the caller's method state prior to making the associated call. This means that the stack will not grow when taking this call. It also means, at least in principle, that it may be possible for the JIT to make the call efficiently using just a jump instruction. This is useful for F#, and ensures that tail recursion is safe in almost all cases:

```
IL_0009: tail.  
IL_000b: call    bool Program/SixThirtyEight::odd(int32)  
IL_0010: ret
```

In CLR 4.0, a few key improvements have been made to the treatment of tail calls. The x64 JIT had previously implemented tail calls very efficiently, but using a technique that could not be applied to all cases where the `tail.` instruction appeared. This meant some F# code that ran successfully on x86 platforms would fail with a stack overflow on x64 platforms. In CLR 4.0, the x64 JIT extends its efficient implementation of tail calls to more cases, and also implements the higher-overhead mechanism needed to ensure that tail calls are taken anytime they would be on the x86 JIT.

A detailed account of the CLR 4.0 improvements for tail calls is available on the CLR Code Generation blog ([blogs.msdn.com/clrcodegeneration/archive/2009/05/11/tail-call-improvements-in-net-framework-4.aspx](http://blogs.msdn.com/clrcodegeneration/archive/2009/05/11/tail-call-improvements-in-net-framework-4.aspx)).

## F# Interactive

F# Interactive is a command-line tool and Visual Studio tool window for interactively executing F# code (see **Figure 1**). This tool makes it easy to experiment with data, explore APIs and test application logic using F#.

F# Interactive is made possible by the CLR Reflection.Emit API. This API allows a program to generate new types and members at run time and call into this new code dynamically. F# Interactive uses the F# compiler to compile code the user inputs at the prompt, then uses Reflection.Emit to generate the types, functions and members instead of writing an assembly to disk.

One key result of this approach is that the user code being executed is fully compiled and fully JITed, including all the useful optimizations in both of these

steps, instead of being an interpreted version of F#. That makes the F# Interactive an excellent, high-performance environment for trying out new problem-solving approaches and interactively exploring large datasets.

## Tuples

Tuples in F# provide a simple way to package data and pass it around as a unit, without needing to define new custom types or use complicated parameter schemes such as out parameters to return multiple values.

```
let printPersonData (name, age) =  
    printfn "%s is %d years old" name age
```

```
let bob = ("Bob", 34)  
  
printPersonData bob
```

```
let divMod n m =  
    n / m, n % m
```

```
let d,m = divMod 10 3
```

Tuples are simple types, but have a few important properties in F#. Most significantly, they're immutable. Once constructed, the elements of a tuple cannot be modified. This allows tuples to be safely treated as just a combination of their elements. It also enables another important feature of tuples: structural equality. Tuples and other F# types such as lists, options, and user-defined records and unions are compared for equality by comparing their elements.

F# Interactive is made possible  
by the CLR Reflection.Emit API.

In the .NET Framework 4, tuples are now a core datatype defined in the base class libraries. When targeting the .NET Framework 4, F# uses the `System.Tuple` type to represent these values. Having support for this core type in `mscorlib` means F# users can easily share tuples with C# APIs and vice versa.

Although tuples are conceptually simple types, there are many interesting design decisions involved in building the `System.Tuple`



type. Matt Ellis covered the design process for Tuple in detail in a recent CLR Inside Out column ([msdn.microsoft.com/magazine/dd942829](http://msdn.microsoft.com/magazine/dd942829)).

## Optimizations

Because F# translates less directly to the CLR instructions, there's more room for optimization to be done in the F# compiler instead of just relying on the CLR JIT compiler. The F# compiler takes advantage of this and implements more significant optimizations in Release mode than the C# and Visual Basic compilers.

One simple example is intermediate tuple elimination. Tuples are frequently used to structure data while it's being processed. It's common for tuples to be created and then deconstructed within a single function body. When this happens, there's an unnecessary allocation of a tuple object. Because the F# compiler knows that creating and deconstructing a tuple can't have any important side effects, it will attempt to avoid allocating the intermediate tuple.

Once constructed, the elements of a tuple cannot be modified.

In this example, no tuple object needs to be allocated, as it is used only by being deconstructed in the pattern match expression:

```
let getValueIfBothAreSame x y =  
    match (x,y) with  
    | (Some a, Some b) when a = b -> Some a  
    | _ -> None
```

## Units of Measure

Units of measure, like meters and seconds, are commonly used in science, engineering and simulation, and are fundamentally a type system for working with numerical quantities of different kinds. In F#, units of measure are brought into the language's type system directly so that numerical quantities can be annotated with their units. These units are carried through computations, and errors are reported when units do not match. In the following example, it's an error to try to add kilograms and seconds, though note that it's not an error to divide kilograms by seconds.

```
// Kilograms  
[<Measure>] type kg  
// Seconds  
[<Measure>] type s  
  
let x = 3.0<kg>  
//val x : float<kg>  
  
let y = 2.5<s>  
// val y : float<s>  
  
let z = x / y  
//val z : float<kg/s>  
  
let w = x + y  
// Error: "The unit of measure 's'  
// does not match the unit of measure 'kg'"
```

Units of measure become a fairly lightweight addition thanks to F# type inference. Using type inference, user-provided unit annotations need to appear only on literals and when accepting data from outside sources. Type inference then propagates these through the program, and checks that all computations are being done correctly according to the units being used.

Although part of the F# type system, units of measure are erased at compilation time. This means the resulting .NET assembly does not include the information about units, and the CLR just treats unitized values as their underlying type—thereby incurring no performance overhead. This is in contrast to .NET generics, which are fully available at run time.

If, in the future, the CLR were to integrate units of measure into the core CLR type system, F# would be able to expose the unit information so it could be seen from other .NET programming languages.

## Get Interactive with F#

As you've seen, F# provides an expressive, functional, object-oriented and explorative programming language for the .NET Framework. It's integrated into Visual Studio 2010—including the F# Interactive tools for jumping straight in and experimenting with the language.

The language and tools leverage the full breadth of the CLR and introduce some higher-level concepts that are mapped onto the meta-data and IL of the CLR. Yet F# is ultimately just another .NET language and can be easily incorporated as a component of new or existing .NET projects, thanks to the common type system and runtime. ■

**LUKE HOBAN** is the program manager for the F# team at Microsoft. Before moving to the F# team, he was the program manager for the C# compiler and worked on C# 3.0 and LINQ.

**The Right Tool  
at the Right Price**  
*Integrated Desktop Load Test Engine*

**LOAD TEST**  
*Express 2010*

**Component Level Testing**

- SQL and Stored Procedures
- .NET Assemblies
- XML Web Services

[www.teoinnovations.com](http://www.teoinnovations.com)





## PROGRAMMING LANGUAGES

### Embarcadero Technologies

Embarcadero empowers software developers and database professionals with award-winning tools including Delphi Prism for rapidly developing .NET and ASP.NET applications.

## CODING & BUILD TOOLS

### AppDev

Award-winning, practical learning solutions for Microsoft developers integrated into Visual Studio 2010. Courses for Visual Studio, SQL, SharePoint and more.

### ArtinSoft

ArtinSoft's unique tools and services allow cost-effectively converting Visual Basic 6.0 applications to Visual Studio 2010 and .NET Framework 4

### Flexera

InstallShield makes it easy to build professional installations for desktop, server, Web, and mobile applications developed in Visual Studio 2010.

### GraFX

Vulcan.NET is the next generation of the xBase family of languages for the Microsoft .NET Platform, opening up new opportunities for xBase (FoxPro, Clipper and Visual Objects) programmers.



## APPLICATION LIFECYCLE M

### GamCom

GamCom provides software and consultancy services. Talmia is an innovative ALM process automation and reporting tool for TFS.

# /\*SHOWCASE YOUR INNO

Find extensions on the Visual Studio Gallery  
[www.visualstudiogallery.com](http://www.visualstudiogallery.com)

Join the Wave of Innovation  
<http://msdn.com/vsip>



**DevExpress**<sup>™</sup>  
Download • Compare • Decide!



**GrapeCity**® **ComponentOne**®

#### COMPONENTS & LIBRARIES

##### **DevExpress**

DevExpress engineers feature-complete Presentation Components, Reporting Systems, IDE Productivity Tools and Business Application Frameworks for Visual Studio .NET.

##### **GrapeCity**

GrapeCity provides complete Spreadsheet, Reporting, Analysis and Business Intelligence Tools for .NET, including the ActiveReports and FarPoint Spread product lines.

##### **ComponentOne**

ComponentOne, a leading component vendor, provides custom WinForms, WPF, ASP.NET AJAX, Silverlight, Mobile, and COM components, documentation tools, and Web Parts.

#### APPLICATION MIGRATION & MODERNIZATION

##### **Alchemy**

Provides modernize-as-you-migrate options for COBOL applications. NetCOBOL for .NET brings the latest Visual Studio 2010 productivity features to COBOL programmers.

#### MANAGEMENT TOOLS

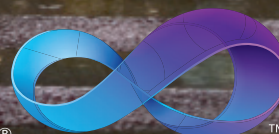
##### **dynaTrace**

Eliminate manual problem reproduction, rapidly isolate and diagnose .NET performance issues to code-level with dynaTrace's extension for VS Ultimate 2010

**LOOK WHAT  
WE DID  
WITH VISUAL  
STUDIO 2010!**

**VATION\*/**

Microsoft®  
**Visual Studio**®





# Address Scalability Bottlenecks with Distributed Caching

Iqbal Khan

After the explosion of Web applications to accommodate high-traffic usage, the next big wave has become service-oriented architecture (SOA). SOA is destined to become a standard way for developing extremely scalable applications, and cloud computing platforms like Windows Azure represent a giant leap in moving SOA toward achieving this goal.

SOA allows users to distribute applications to multiple locations, multiple departments within an organization, and multiple businesses across the Internet. Plus, it permits reuse of existing code within an organization and, more importantly, collaboration among different business units.

A SOA application is usually deployed in a server farm in a load-balanced environment. The goal is to allow the application to handle as much load as you throw at it. The question thus becomes: What are some of the considerations you should have in mind for improving both performance and scalability of your SOA application?

Although SOA, by design, is intended to provide scalability, there are many issues you must address before you can truly achieve scalability. Some of these issues involve how you code your SOA application, but the most important bottlenecks often relate to how you store and access your data. I'll explore those issues and provide some solutions in this article.

## Find Scalability Bottlenecks

A true SOA application should scale easily as far as the application architecture is concerned. A SOA application has two components: service components and client applications. The client application may be a Web application, another service or any other application that relies on the SOA service components to do its job.

One of the key ideas behind SOA is to break up the application into small chunks so these components can be run on multiple servers as separate services.

Ideally, these services should be stateless as much as possible. Stateless means they don't retain any data with them across multiple calls, allowing you to run the services on multiple computers. There's no dependence on where the data was the last time, so there's no data being kept on any particular server across multiple service calls.

As a result, the architecture of SOA applications is inherently scalable. It can easily grow onto multiple servers and across data-centers. However, as with every other application, SOA applications do have to deal with the data, and that can be a problem. This data

### This article discusses:

- Code for performance
- Choose the right communication protocol
- Using a distributed cache for scalability
- Synchronizing the cache with a database

### Technologies discussed:

ASP.NET, Windows Communication Foundation



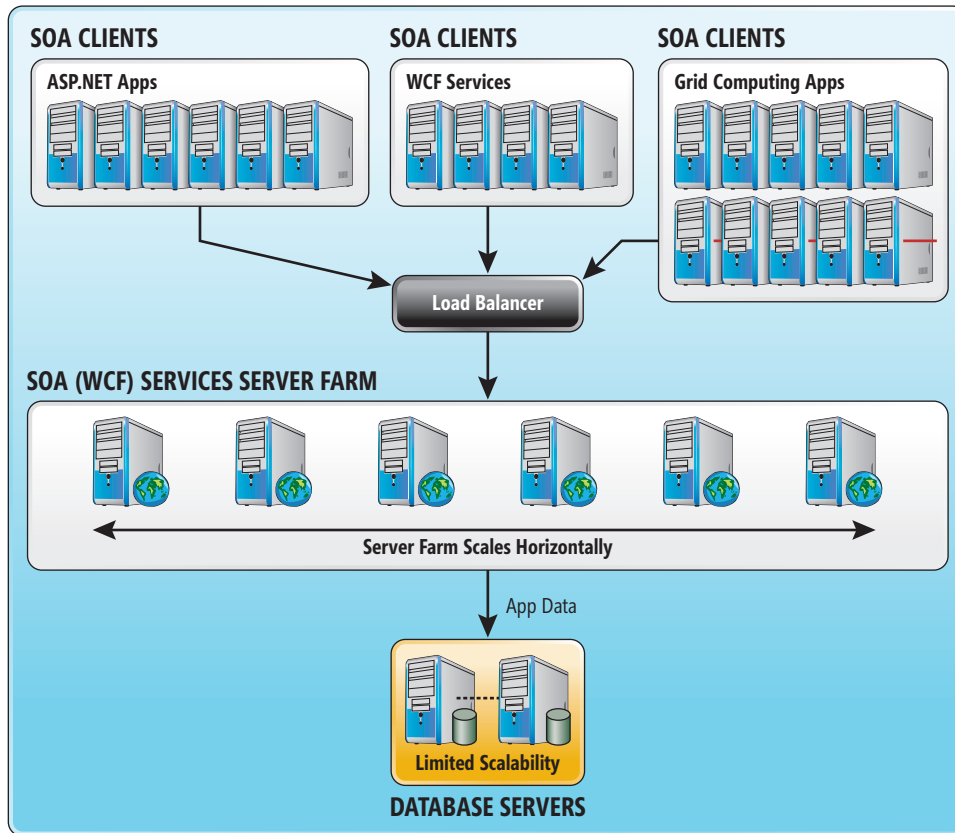


Figure 1 SOA Architecture with Potential Scalability Bottlenecks

access becomes the scalability bottleneck. Bottlenecks typically involve the application data, which is stored in some database, usually a relational database. If the SOA application is using session data, the storage of that data is also another potential scalability bottleneck.

One SOA application relying on other SOA applications is another likely area of poor performance and scalability. Say your application calls one service to do its job, but that service calls out to other services. Those services may be on the same intranet or across the WAN in other locations. Such a data trip can be costly. You can't scale the application effectively if you're making those calls over and over again, and these are areas where scalability bottlenecks occur, as shown in **Figure 1**.

## Code for Performance

There are a number of programming techniques that can help improve your SOA application performance.

One thing you can do is design your application to use "chunky" Web method calls. Don't make frequent calls between the SOA client application and the SOA service layer. There's usually a great distance between those because they're not running on the same computer or even in the same datacenter. The fewer calls you make from the client application to the service layers, the better the performance. Chunky calls do more work in one call than multiple calls to do the same work.

Another useful technique is to employ the asynchronous Web method calls supported by the Microsoft .NET Framework. This allows your

SOA client application to continue doing other things while the Web method of the service layer is being called and is executing.

The cost of serialization is another aspect to factor in so you don't serialize any unnecessary data. You should only send data that is required back and forth, allowing you to be highly selective about the type of serialization you want to perform.

## Choose the Right Communication Protocol

For SOA applications developed in Windows Communication Foundation (WCF), there are three different protocols that let SOA clients talk to SOA services. These are HTTP, TCP and named pipes.

If both your client and your service are developed in WCF and are running on the same machine, named pipes offer the best performance. A named pipe uses shared memory between client and server processes.

TCP is good if both SOA client and server are developed in WCF, but are running on different computers in the same intranet. TCP is faster than HTTP, but a TCP connection stays open across multiple calls and therefore you can't automatically route each WCF call to a different server. By employing the `NetTcpBinding` option that uses connection pools, you can expire TCP connections frequently to restart them so they get routed to a different server, thereby giving you a form of load balancing.

A true SOA application  
should scale easily.

Please note that TCP can't work reliably across the WAN because socket connections tend to break frequently. If your SOA client and service are not based on WCF or they're hosted in different locations, then HTTP is your best option. Although HTTP is not as fast as TCP, it offers great scalability due to load balancing.

## Use Caching to Improve Client Performance

Thoughtful use of caching can really improve SOA client performance. When a SOA client makes a Web method call to the service layer, you can cache the results at the client application's end. Then, the next time this SOA client needs to make the same Web method call, it gets that data from the cache instead.

Figure 2 WCF Client Caching

```
using System;
using Client.EmployeeServiceReference;

using Alachisoft.NCache.Web.Caching;

namespace Client {
    class Program {
        static string _sCacheName = "mySOAClientCache";
        static Cache _sCache = NCache.InitializeCache(_sCacheName);

        static void Main(string[] args) {
            EmployeeServiceClient client =
                new EmployeeServiceClient("WSHttpBinding_IEmployeeService");

            string employeeId = "1000";
            string key = "Employee:EmployeeId:" + employeeId;

            // first check the cache for this employee
            Employee emp = _sCache.Get(key);

            // if cache doesn't have it then make WCF call
            if (emp == null) {
                emp = client.Load("1000");

                // Now add it to the cache for next time
                _sCache.Insert(key, emp);
            }
        }
    }
}
```

By caching data at the client end, the SOA client application reduces the number of calls it's going to make to the service layer. This step boosts performance because it didn't have to make an expensive SOA service call. It also reduces overall pressure on the service layer and improves scalability. **Figure 2** shows a WCF client using caching.

In many situations, your client is physically removed from the service layer and is running across the WAN. In that case, you have no way of knowing whether the data you have cached has been updated. Therefore, you have to identify only those data elements for caching that you feel will not change for at least a few minutes to perhaps a few hours, depending on your application. You can then specify expiration for these data elements in the cache so the cache will automatically remove them at that time. This helps ensure that cached data is always fresh and correct.

### Distributed Caching for Service Scalability

The real scalability gains through caching are found in the SOA service layer. Scalability bottlenecks are not always removed de-

spite many of the programming techniques mentioned already because the major scalability bottlenecks are with data storage and access. Services often live in a load-balanced server farm, allowing the service itself to scale quite nicely—except the data storage can't scale in the same manner. Data storage thus becomes the SOA bottleneck.

You can scale the service layer by adding more servers to the server farm, increasing the computing capacity through these additional application servers. But all those SOA transactions still deal with some data. That data has to be stored somewhere, and that data storage can easily become the bottleneck.

The fewer calls you make  
from the client application  
to the service layers, the better  
the performance.

This data storage barrier to scalability can be improved at multiple levels. SOA services deal with two types of data. One is session-state data and the other is application data that resides in the database (see **Figure 3**). Both cause scalability bottlenecks.

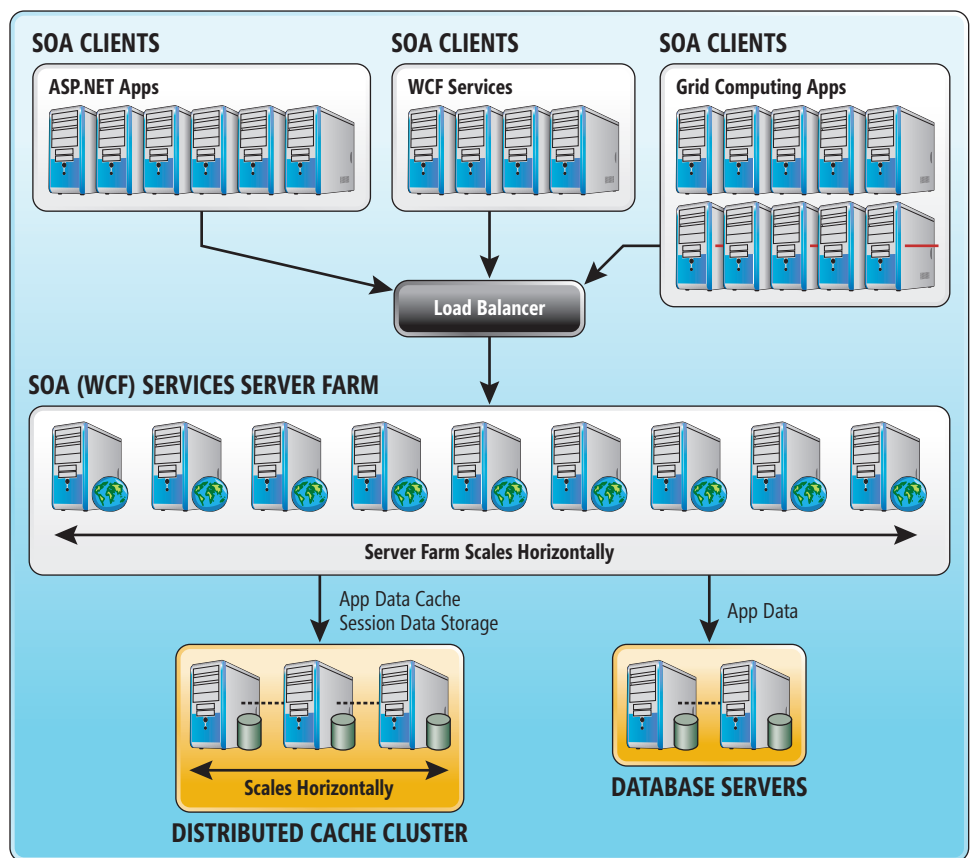


Figure 3 How Distributed Caching Reduces Pressure on a Database








# Deliver innovation with ease<sup>®</sup>

with Syncfusion's award-winning .NET components and controls



User interfaces, reporting, business intelligence.  
Any .NET platform. Just one package.

 ASP.NET  ASP.NET MVC  Windows Forms  WPF  Silverlight



Ready for  
  
Visual Studio  
2010



[www.syncfusion.com](http://www.syncfusion.com) 1 888-9 DOTNET

## Storing Session State in a Distributed Cache

One of the limitations of the default session-state storage is that it does not support Web farms because it is in-memory storage living inside the WCF service process. A much better alternative is to use ASP.NET compatibility mode and the ASP.NET session state in WCF services. This allows you to specify OutProc storage including StateServer, SqlServer, or a distributed cache as session state storage.

Enabling ASP.NET compatibility mode is a two-step process. First, you have to specify ASP.NET compatibility in your class definition, as shown in **Figure 4**. Then you have to specify this in your app.config file, as shown in **Figure 5**. Notice that **Figure 4** also demonstrates how to specify a distributed cache as your SessionState storage in the same web.config file.

StateServer and SqlServer session storage options do not scale well and, in the case of StateServer, it is also a single point of failure. A distributed cache is a much better alternative because it scales nicely and replicates sessions to multiple servers for reliability.

## Caching Application Data

Application data is by far the heaviest data usage in a WCF service, and its storage and access is a major scalability bottleneck. To address this scalability-bottleneck problem, you can use distributed caching in your SOA service-layer implementation. A distributed cache is used to cache only a subset of the data that is in the database based on what the WCF service needs in a small window of a few hours.

The real scalability gains  
through caching are found in the  
SOA service layer.

Additionally, a distributed cache gives a SOA application a significant scalability boost because this cache can scale out as a result of the architecture it employs. It keeps things distributed across multiple servers—and still gives your SOA application one logical view so you think it's just one cache. But the cache actually lives on multiple servers and that's what allows the cache to really scale. If you use distributed caching in between the service layer and the database, you'll improve performance and scalability of the service layer dramatically.

The basic logic to implement is that, before going to the database, check to see if the cache already has the data. If it does, take it from the cache. Otherwise, go to the database to fetch the data and put it in the cache for next time. **Figure 6** shows an example.

By caching application data, your WCF service saves a lot of expensive database trips and instead finds the frequently used transactional data in a nearby in-memory cache.

## Expiring Cached Data

Expirations let you specify how long data should stay in the cache before the cache automatically removes it. There are two types of

Figure 4 Specifying ASP.NET Compatibility for WCF Services in Code

```
using System;
using System.ServiceModel;
using System.ServiceModel.Activation;

namespace MyWcfServiceLibrary {
    [ServiceContract]
    public interface IHelloWorldService {
        [OperationContract]
        string HelloWorld(string greeting);
    }

    [ServiceBehavior (InstanceContextMode =
        InstanceContextMode.PerCall)]
    [AspNetCompatibilityRequirements (RequirementsMode =
        AspNetCompatibilityRequirementsMode.Allowed)]

    public class HelloWorldService : IHelloWorldService {
        public string HelloWorld(string greeting) {
            return string.Format("HelloWorld: {0}", greeting);
        }
    }
}
```

expirations you can specify: absolute-time expiration and sliding-or idle-time expiration.

If the data in your cache also exists in the database, you know that this data can be changed in the database by other users or applications that may not have access to your cache. When that happens, the data in your cache becomes stale, which you do not want. If you're able to make a guess as to how long you think it's safe for this data to be kept in the cache, you can specify absolute-time expiration. You can say something like "expire this item 10 minutes from now" or "expire this item at midnight today." At that time, the cache expires this item:

```
using Vendor.DistCache.Web.Caching;
...
// Add an item to ASP.NET Cache with absolute expiration
_sCache.Insert(key, employee, null,
    DateTime.Now.AddMinutes(2),
    Cache.NoSlidingExpiration,
    CacheItemPriority.Default, null);
```

You can also use idle-time or sliding-time expiration to expire an item if nobody uses it for a given period. You can specify something

Figure 5 Specifying ASP.NET Compatibility for WCF Services in Config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <sessionState cookieless="UseCookies"
      mode="Custom"
      customProvider="DistCacheSessionProvider"
      timeout="20">
      <providers>
        <add name="DistCacheSessionProvider"
          type="Vendor.DistCache.Web.SessionState.SessionStoreProvider"/>
      </providers>
    </sessionState>
    <identity impersonate="true"/>
  </system.web>

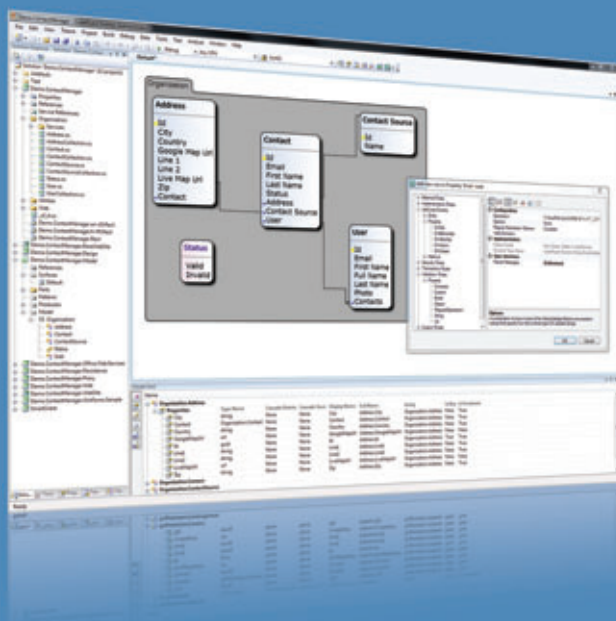
  <system.serviceModel>
    <!-- ... -->
    <serviceHostingEnvironment
      aspNetCompatibilityEnabled="true"/>
  </system.serviceModel>
</configuration>
```





Need a **HAND?**

Use **CODEFLUENT ENTITIES!**



#### APPLICATION BLOCKS

- Localization
- Data Binding
- Rules and Validation
- Concurrency
- Security
- Caching
- Blob handling

#### SUPPORTED ARCHITECTURES

- SOA, SmartClient
- Rich Client, RIA, Silverlight,
- Web, Webparts
- Client/Server, N-Tier
- Office
- SharePoint
- SaaS, Cloud

#### FEATURED TECHNOLOGIES

- .NET (2 to 4), C#, Linq
- ASP .NET (WebForms, MVC)
- Silverlight (2 to 4)
- WPF, WinForms
- WCF, ASMX
- SQL Server (2000 to 2008)
- Oracle Database (9 to 11)
- Office (97 to 2010)
- SharePoint (2007 to 2010)
- Visual Studio (2005 to 2010)

## CUSTOMER APPROVED MODEL-DRIVEN SOFTWARE FACTORY

**We understand the challenges that come with today's and tomorrow's technology integration and evolution.**

CodeFluent Entities is a fully integrated Model-Driven Software Factory which provides architects and developers a structured method and the corresponding tools to develop .NET applications, based on any type of architecture, from an ever changing business model and rules, at an unprecedented productivity level.

CodeFluent Entities is based on a pluggable producer logic, which, from a declarative model you designed, continuously generates ready-to-use, state-of-the-art, scalable, high-performance, and easily debuggable source code, components, and everything you need.

Download your **FREE** trial today at:

[www.CodeFluentEntities.com/Msdn](http://www.CodeFluentEntities.com/Msdn)



Contact: [info@softfluent.com](mailto:info@softfluent.com)  
[www.CodeFluentEntities.com](http://www.CodeFluentEntities.com)

Figure 6 WCF Service Using Caching

```
using System;
using System.Collections.Generic;
using System.ServiceModel;
using Vendor.DistCache.Web.Caching;

namespace MyWcfServiceLibrary {
    [ServiceBehavior]
    public class EmployeeService : IEmployeeService {
        static string _sCacheName = "myServiceCache";
        static Cache _sCache =
            DistCache.InitializeCache(_sCacheName);

        public Employee Load(string employeeId) {
            // Create a key to lookup in the cache.
            // The key for will be like "Employees:PK:1000".
            string key = "Employee:EmployeeId:" + employeeId;

            Employee employee = (_sCache[key]);
            if (employee == null) {
                // item not found in the cache.
                // Therefore, load from database.
                LoadEmployeeFromDb(employee);

                // Now, add to cache for future reference.
                _sCache.Insert(key, employee, null,
                    Cache.NoAbsoluteExpiration,
                    Cache.NoSlidingExpiration,
                    CacheItemPriority.Default);
            }

            // Return a copy of the object since
            // ASP.NET Cache is InProc.
            return employee;
        }
    }
}
```

like “expire this item if nobody reads or updates it for 10 minutes.” This is useful when your application needs the data temporarily and when your application is done using it, you want the cache to automatically expire it. ASP.NET compatibility-mode session state is a good example of idle-time expiration.

Application data is by far  
the heaviest data usage in  
a WCF service.

Notice that absolute-time expiration helps you avoid situations where the cache has an older or stale copy of the data than the master copy in the database. On the other hand, idle-time expiration serves a totally different purpose. It's meant really to simply clean up the cache once your application no longer needs the data. Instead of having your application keep track of this clean up, you let the cache take care of it.

## Managing Data Relationships in the Cache

Most data comes from a relational database, and even if it's not coming from a relational database, it's relational in nature. For example, you're trying to cache a customer object and an order object and both objects are related. A customer can have multiple orders.

When you have these relationships, you need to be able to handle them in a cache. That means the cache should know about the relationship between a customer and an order. If you update or remove the customer from the cache, you may want the cache to automatically remove the order object from the cache. This helps maintain data integrity in many situations.

If a cache can't keep track of these relationships, you'll have to do it yourself—and that makes your application more cumbersome and complex. It's a lot easier if you just tell the cache when you add the data about this relationship. The cache then knows if that customer is ever updated or removed, the order also has to be removed.

Absolute-time expiration helps  
you avoid situations where the  
cache has an older or stale copy.

ASP.NET has a useful feature called `CacheDependency` that allows you to keep track of relationships between different cached items. Some commercial caches also have this feature. Here's an example of how ASP.NET lets you keep track of relationships among cached items:

```
using Vendor.DistCache.Web.Caching;
...
public void CreateKeyDependency() {
    Cache["key1"] = "Value 1";

    // Make key2 dependent on key1.
    String[] dependencyKey = new String[1];
    dependencyKey[0] = "key1";
    CacheDependency dep1 =
        new CacheDependency(null, dependencyKey);

    _sCache.Insert("key2", "Value 2", dep2);
}
```

This is multi-layer dependency, meaning A can depend on B and B can depend on C. So, if your application updates C, both A and B have to be removed from the cache.

Figure 7 Synchronizing Data via SQL Dependency

```
using Vendor.DistCache.Web.Caching;
using System.Data.SqlClient;
...

public void CreateSqlDependency(
    Customers cust, SqlConnection conn) {

    // Make cust dependent on a corresponding row in the
    // Customers table in Northwind database

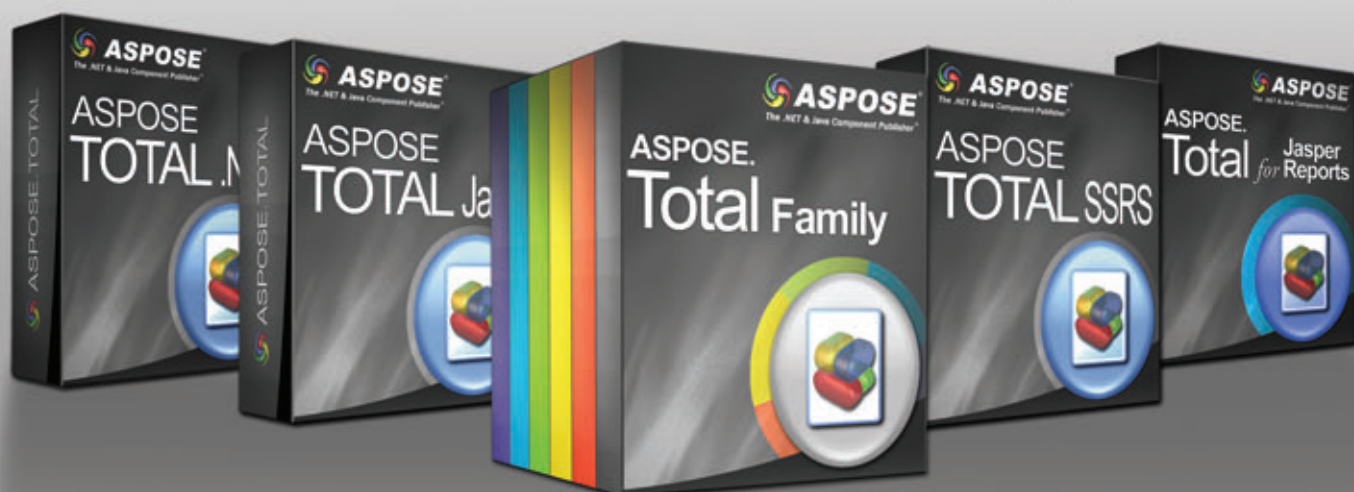
    string sql = "SELECT CustomerID FROM Customers WHERE ";
    sql += "CustomerID = @ID";
    SqlCommand cmd = new SqlCommand(sql, conn);
    cmd.Parameters.Add("@ID", System.Data.SqlDbType.VarChar);
    cmd.Parameters["@ID"].Value = cust.CustomerID;

    SqlCacheDependency dep = new SqlCacheDependency(cmd);
    string key = "Customers:CustomerID:" + cust.CustomerID;
    _sCache.Insert(key, cust, dep);
}
```





- ✓ **More Formats**
- ✓ **More Platforms**
- ✓ **Used by more than **50%** of Fortune 100 Companies!**



The TOTAL Solutions for .NET, Java, SQL Server Rendering Extensions, SharePoint, and JasperReports Exporters.  
*Aspose provides extensive file format processing capabilities for the most popular formats including:*

DOCX PDF PPT ODF Report SWF InfoPath  
XLSX BarCode MPP(Project) MSG(Outlook) ++

Get your FREE evaluation copy at [www.aspose.com](http://www.aspose.com).



## Synchronizing the Cache with a Database

The need for database synchronization arises because the database is really being shared across multiple applications, and not all of those applications have access to your cache. If your WCF service application is the only one updating the database and it can also easily update the cache, you probably don't need the database-synchronization capability.

But, in a real-life environment, that's not always the case. Third-party applications update data in the database and your cache becomes inconsistent with the database. Synchronizing your cache with the database ensures that the cache is always aware of these database changes and can update itself accordingly.

Synchronizing with the database usually means invalidating the related cached item from the cache so the next time your application needs it, it will have to fetch it from the database because the cache doesn't have it.

ASP.NET has a `SqlCacheDependency` feature that allows you to synchronize the cache with SQL Server 2005, SQL Server 2008 or Oracle 10g R2 and later—basically any database that supports the CLR. Some of the commercial caches also provide this capability. **Figure 7** shows an example of using SQL dependency to synchronize with a relational database.

ESB is a simple and powerful way for multiple applications to share data asynchronously.

One capability that ASP.NET does not provide, but some commercial solutions do, is polling-based database synchronization. This is handy if your DBMS doesn't support the CLR and you can't benefit from `SqlCacheDependency`. In that case, it would be nice if your cache could poll your database at configurable intervals and detect changes in certain rows in a table. If those rows have changed, your cache invalidates their corresponding cached items.

## Enterprise Service Bus for SOA Scalability

Enterprise Service Bus (ESB) is an industry concept where many technologies are used to build it. An ESB is an infrastructure for Web services that mediates communication among components. Put plainly, an ESB is a simple and powerful way for multiple applications to share data asynchronously. It is not meant to be used across

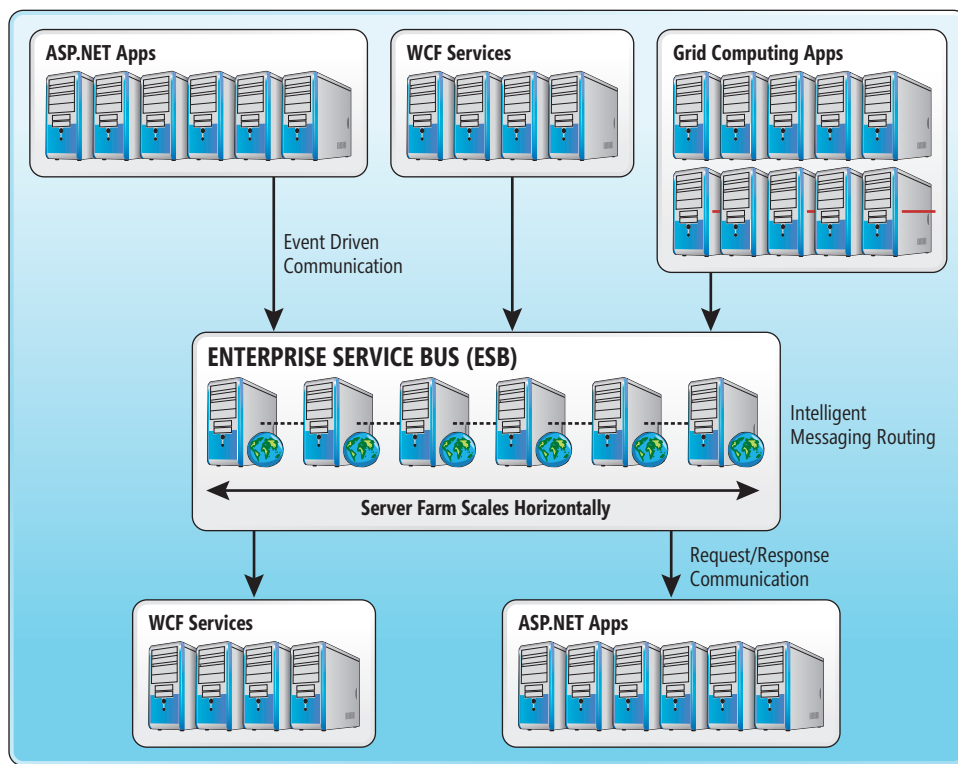


Figure 8 An ESB Created with a Distributed Cache

organizations or even across a WAN, however. Usually SOA applications are by design broken up into multiple pieces, so when they need to share data with each other, ESB is a powerful tool.

There are many ways to build an ESB. **Figure 8** shows an example of an ESB created with a distributed cache. Multiple loosely coupled applications or service components can use it to share data with each other in real time and across the network.

A distributed cache by its nature spans multiple computers. This makes it highly scalable, which meets the first criterion of an ESB. In addition, a good distributed cache replicates all its data intelligently to ensure that no data loss occurs if any cache server goes down. (I'll discuss this later.) Finally, a good distributed cache provides intelligent event-propagation mechanisms.

There are two types of events that a distributed cache must provide to be fit for an ESB. First, any client application of the ESB should be able to register interest in a data element on the ESB so if anybody modifies or deletes it, the client application is notified immediately. Second, the cache should allow client applications to fire custom events into the ESB so all other applications connected to the ESB that are interested in this custom event are immediately notified, no matter where they are on the network (of course, within the intranet).

With the help of an ESB, a lot of data exchange that would otherwise require SOA calls from one application to another can be done very easily through the ESB. Additionally, asynchronous data sharing is something a simple WCF service is not designed to do easily. But the ESB makes this job seamless. You can easily create situations where data is even pushed to the clients of the ESB if they have shown interest in it up front.



# Lightweight. Intuitive. Fast.

*The solution that helps you to focus on your real job: developing great software.*

## Seamless task tracking and source control

Browse your tasks, check on progress, automatically retrieve the correct code base and know who's working on the same project files.

## Full Visual Studio integration

Whether you're working with the GUI or the full Visual Studio integration, all PureCM features are available at your fingertips.

## High performance, scalable

Private developer workspaces with fast checkin operations and full support for distributed development, even when offline.



Agile planning and source control.  
Get your free evaluation resources at  
[www.purecm.com/start10-1](http://www.purecm.com/start10-1)



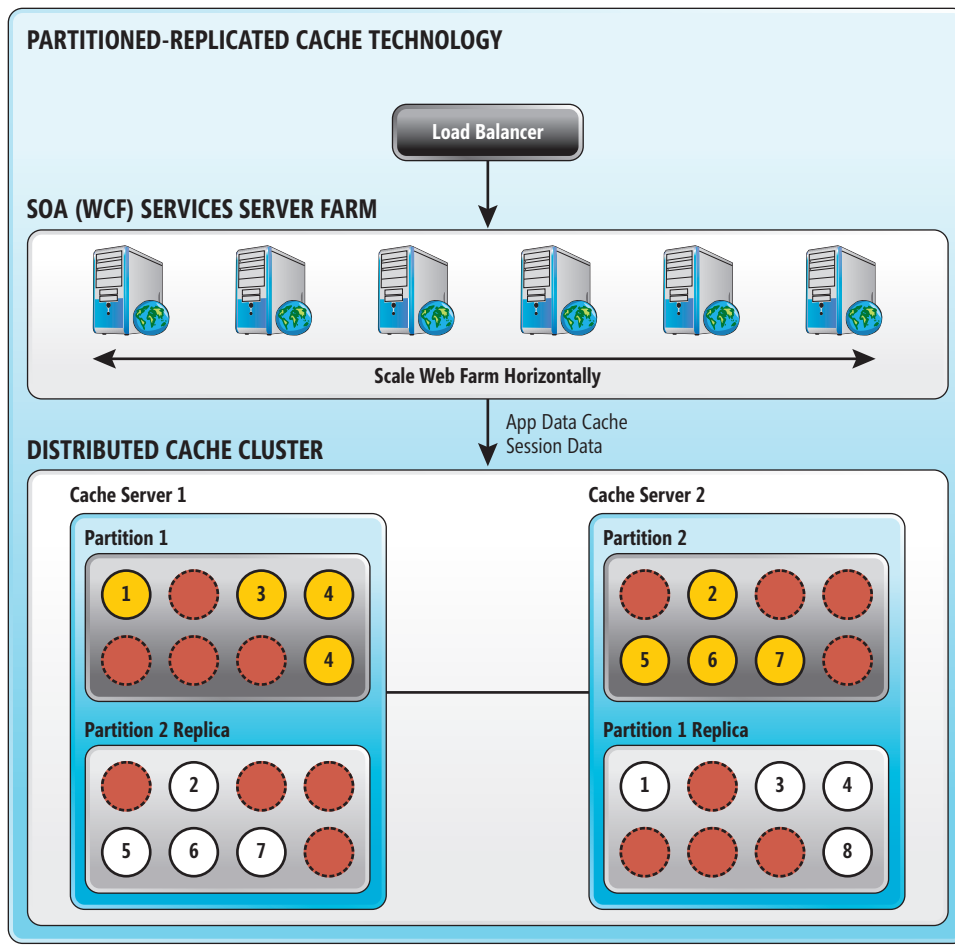


Figure 9 Partitioned-Replicated Caching Topology for Scalability

## Cache Scalability and High Availability

Caching topology is a term used to indicate how data is actually stored in a distributed cache. There are various caching topologies that are designed to fit different environments. I'll discuss three here: partitioned cache, partitioned-replicated cache and replicated cache.

High availability can be further enhanced through dynamic cache clustering.

Partitioned and partitioned-replicated are two caching topologies that play major roles in the scalability scenario. In both topologies, the cache is broken up into partitions, then each partition stored in different cache servers in the cluster. Partitioned-replicated cache has a replica of each partition stored on a different cache server.

Partitioned and partitioned-replicated caches are the most scalable topology for transactional data caching (where writes to the cache are as frequent as reads) because, as you add more cache servers to the cluster, you're not only increasing the transaction

capacity, you're also increasing the storage capacity of the cache because all those partitions together form the entire cache.

A third caching topology, replicated cache, copies the entire cache to each cache server in the cache cluster. This means the replicated cache provides high availability and is good for read-intensive usage. It is not good for frequent updates, however, because updates are done to all copies synchronously and are not as fast as with other caching topologies.

As shown in **Figure 9**, partitioned-replicated cache topology is ideal for a combination of scalability and high availability. You don't lose any data because of the replicas of each partition.

High availability can be further enhanced through dynamic cache clustering, which is the ability to add or remove cache servers from the cache cluster at run time without stopping the cache or the client applications. Because a distributed cache runs in a production environment, high availability is an important feature requirement.

## Next Steps

As you've seen, an SOA application can't scale effectively when the data it uses is kept in a storage that is not scalable for frequent transactions. This is where distributed caching really helps.

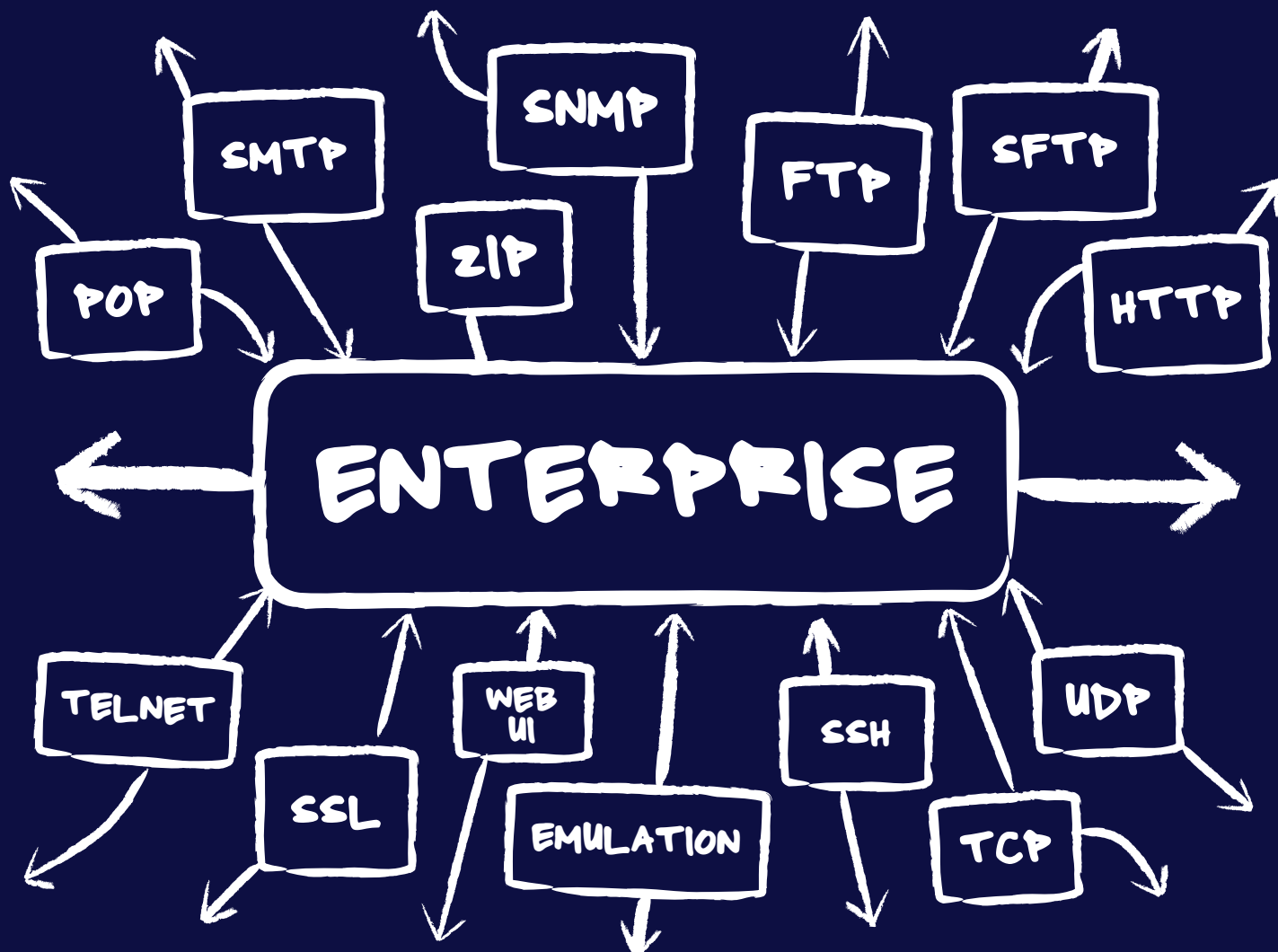
Distributed caching is a new concept but rapidly gaining acceptance among .NET developers as a best practice for any high-transaction application. The traditional database servers are also improving but without distributed caching, they can't meet the exploding demand for scalability in today's applications.

The techniques I've described should help take your SOA apps to new levels of scalability. Give them a try today. For more discussion on distributed caching, see the MSDN Library article by J.D. Meier, Srinath Vasireddy, Ashish Babbar, and Alex Mackman at [msdn.microsoft.com/library/ms998562](http://msdn.microsoft.com/library/ms998562).

**IQBAL KHAN** is president and technology evangelist at Alachisoft. Alachisoft provides NCache, an industry-leading .NET distributed cache for boosting performance and scalability in enterprise applications. Khan has a master's in computer science from Indiana University, Bloomington. You can reach him at [iqbal@alachisoft.com](mailto:iqbal@alachisoft.com).

**THANKS** to the following technical experts for reviewing this article:  
Kirill Gavrylyuk and Stefan Schackow





# Internet Connectivity for the Enterprise

Since 1994, Dart has been a leading provider of high quality, high performance Internet connectivity components supporting a wide range of protocols and platforms. Dart's three product lines offer a comprehensive set of tools for the professional software developer.



## PowerSNMP for ActiveX and .NET

Create custom Manager, Agent and Trap applications with a set of native ActiveX, .NET and Compact Framework components. **SNMPv1, SNMPv2, SNMPv3** (authentication/encryption) and **ASN.1** standards supported.

## PowerWEB for ASP.NET

AJAX enhanced user interface controls for responsive ASP.NET applications. Develop unique solutions by including streaming file upload and interactive image pan/zoom functionality within a page.

## PowerTCP for ActiveX and .NET

Add high performance Internet connectivity to your ActiveX, .NET and Compact Framework projects. Reduce integration costs with detailed documentation, hundreds of samples and an expert in-house support staff.

SSH  
UDP  
TCP  
SSL

FTP  
SFTP  
HTTP  
POP

SMTP  
IMAP  
S/MIME  
Ping

DNS  
Rlogin  
Rsh  
Rexec

Telnet  
VT Emulation  
ZIP Compression  
*more...*

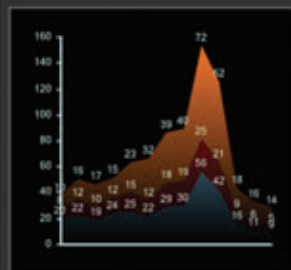
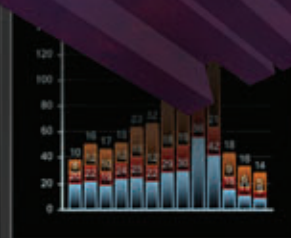
Ask us about Mono Platform support. Contact [sales@dart.com](mailto:sales@dart.com).

**Download a fully functional product trial today!**

 **DART.com**

Alien approved for 20+ Light Years!

# THE CONTROLS HAVE LANDED.



DATA

- Drive C
  - Program Files
    - Visual Studio 2008
    - ComponentOne Studio
      - Bin
      - PhotoShop
      - Corel 11
  - Drive D
    - My Images
    - My Music
- NET
- LOCAL
- 2010
  - Jan
  - Feb
  - Mar
  - Apr
  - May
  - Jun
  - Jul
  - Aug
  - Sep
  - Oct
  - Nov
  - Dec

USERCODE	BAND	LASTNAME	FIRSTNAME	CONTACTED	COMPANY	CUSTTYPE
ACMC	Clark	Allen		NoFilter	Microcomputer Consulting	2
AGCE	Gordon	Alan		Contains	Computer Engineering	4
AQSD	Quinn	Ann		NotContain	Software Designs	1
AWCC	Wheeler	Alice		BeginsWith	Computer Communications	2
BAMU	Ardmore	Beverly		EndsWith	Microcomputers Limited	1
CFA	Fredericks	Carey		Equals	Computer Applications	2
CZMM	Ziegler	Carl		NotEqual	Micro Mechanics	4
DESS	Elkins	David		IsEmpty	Software Specialists	1
DUPD	Underwood	Dennis		NotIsEmpty	Game Designs	1
EFAT	Frank	Ellen		Greater	Advanced Technologies	5
EPML	Parsons	Emily		GreaterOrEqual	Microdecisions Inc	4
FRGW	Rogers	Frank		Less	Computer Warehouse	2
GRMS	Rest	George		LessOrEqual	Micromed Systems	3

100'S OF CONTROLS, 7 PLATFORMS

Featuring WinForms • WPF • ASP.NET AJAX • Silverlight • iPhone • Mobile • ActiveX  
Grids • Charting • Reporting • Scheduling • Menus and Toolbars • Ribbon • Data Input • Editors • PDF



# CUSTOM MICROSOFT VISUAL STUDIO 2010-READY CONTROLS

SEE A DEMO, GET FREE SOCKS!  
VISIT US @ TECH ED 2010  
BOOTH #1740

Experience all the rich, new features of  
**Visual Studio 2010** along with the ability to:

- Improve your entire Web Form apps in minutes with AJAX 4.0 controls.
- Style your Silverlight applications using visual brush creation and ComponentOne ClearStyle technology.
- Add more data visualization such as charts and gauges in WinForms and ASP.NET AJAX.
- Add docking and floating capabilities to WPF windows.



ComponentOne®

## Studio Enterprise 2010

With a sighting this big, you have to download it to believe.

*Get yours today at:*  
**COMPONENTONE.COM/HERE**

Component**One**

ComponentOne Sales: 1.800.858.2739 or 1.412.681.4343

© 1987-2010 ComponentOne LLC. All rights reserved. iPhone and iPod are trademarks of Apple Inc. While supplies last. Void where prohibited or restricted by law. All other product and brand names are trademarks and/or registered trademarks of their respective holders.

# Resource Contention Concurrency Profiling in Visual Studio 2010

Maxim Goldin

As **multicore processors** become ever more commonplace, software developers are building multithreaded applications that take advantage of the additional processing capacity to achieve better performance. Using the power of parallel threads, you can split the overall work into separate tasks and execute those tasks in parallel.

Threads, however, often need to communicate with each other to complete a task, and sometimes need to synchronize their behavior if an algorithm or data access requires it. For example, simultaneous write access to the same data should be granted to threads in a mutually exclusive fashion to avoid data corruption.

Synchronization is frequently accomplished through the use of shared synchronization objects, where the thread acquiring the object is granted either shared or exclusive access to the sensitive code or data. When access is no longer required, the thread relinquishes ownership and other threads can attempt to acquire access. Depending on the type of synchronization used, simultaneous

requests for ownership might allow multiple threads to access shared resources at the same time, or some of the threads might be blocked until the object is released from previous acquisition. Examples include critical sections in C/C++ that use `EnterCriticalSection` and `LeaveCriticalSection` access routines, the `WaitForSingleObject` function in C/C++ and the lock statement and the `Monitor` class in C#.

The choice of synchronization mechanism must be made with care, because improper synchronization between threads can reduce rather than enhance the performance gains that are the objective of multithreading. Thus, it is increasingly important to be able to detect situations where threads are blocked due to lock contentions that make no progress.

The performance tools in Visual Studio 2010 include a new profiling method—resource contention profiling—that helps you detect concurrency contention among threads. You can find a great first look at this feature in John Robbins' Wintellect blog post at [wintellect.com/CS/blogs/jrobbins/archive/2009/10/19/vs-2010-beta-2-concurrency-resource-profiling-in-depth-first-look.aspx](http://wintellect.com/CS/blogs/jrobbins/archive/2009/10/19/vs-2010-beta-2-concurrency-resource-profiling-in-depth-first-look.aspx).

In this article, I walk through a contention-profiling investigation and explain the data that can be collected using both the Visual Studio 2010 IDE and command-line tools. I'll also show you how you can analyze the data in Visual Studio 2010, and you'll see how to move from one analysis view to another while conducting your contention investigation. Then I will fix the code and compare profiling results of the modified application with the original profiling results to validate that the fix reduces the number of contentions.

## This article discusses:

- Profiling data collection
- Resource contention blocks
- Function and thread details
- Chasing down a problem

## Technologies discussed:

Visual Studio 2010



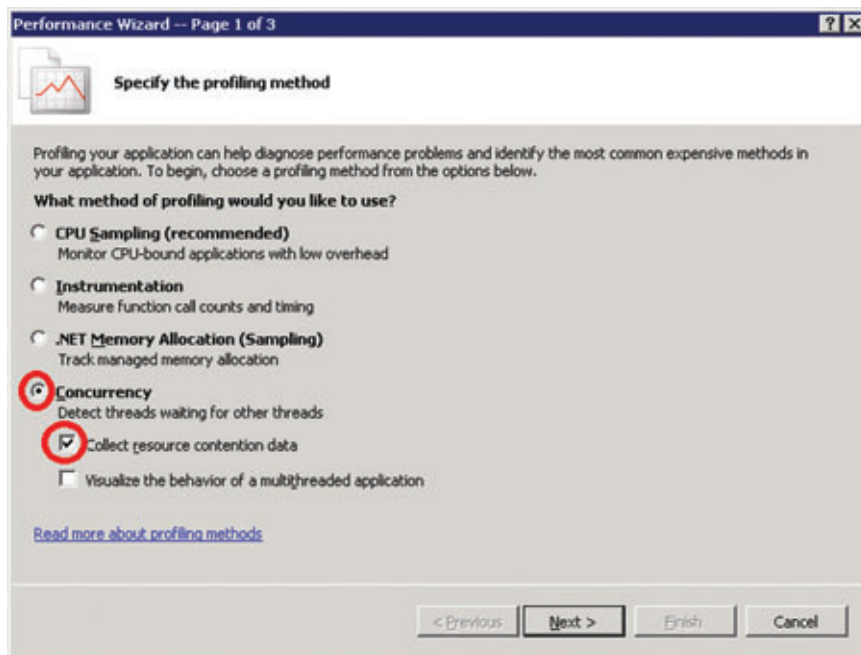


Figure 1 Enabling Concurrency Resource Profiling

## Start with the Problem

As an example, I'll use the same matrix multiplication application Hazim Shafi used in his blog post "Performance Pattern 1: Identifying Lock Contention" ([blogs.msdn.com/hshafi/archive/2009/06/19/performance-pattern-1-identifying-lock-contention.aspx](http://blogs.msdn.com/hshafi/archive/2009/06/19/performance-pattern-1-identifying-lock-contention.aspx)). The code example is written in C++, but the concepts I'll discuss are equally applicable to managed code.

## Not all synchronization causes lock contention.

The example matrix multiplication application uses several threads to multiply two matrixes. Each thread gets a portion of the job and runs the following code snippet:

```
for (i = myid*PerProcessorChunk;
     i < (myid+1)*PerProcessorChunk;
     i++) {
    EnterCriticalSection(&mmlock);
    for (j=0; j<SIZE; j++) {
        for (k=0; k<SIZE; k++) {
            C[i][j] += A[i][k]*B[k][j];
        }
    }
    LeaveCriticalSection(&mmlock);
}
```

Each thread has its own ID (myid), and is responsible for calculating the number of rows (one or more) in the resulting matrix C, using matrixes A and B as an input. Close code inspection shows that no truly ambiguous write-sharing happens, and each thread writes to a different row of C. Yet the developer decided to guard the

assignment to the matrix with a critical section. I thank the developer for this, as it gives me a good opportunity to demonstrate the new Visual Studio 2010 performance tools to easily find the redundant synchronization.

## Profiling Data Collection

Assuming you have a Visual Studio project with the code shown earlier (although it's not required—you can attach the profiler to any application that's already running), you start contention profiling by clicking Launch Performance Wizard on the Analyze menu.

On the first page of the wizard, shown in **Figure 1**, choose Concurrency and make sure the "Collect resource contention data" option is checked. Note that resource contention concurrency profiling works on any version of the Windows OS. The "Visualize the behavior of a multithreaded application" option requires Windows Vista or Windows 7.

On the second page of the wizard, make sure the current project is targeted. On the

last page of the wizard, make sure "Launch profiling after the wizard finishes" option is checked, then click Finish. The application starts running under the profiler. When it exits, the profiling data file appears in the Performance Explorer window (see **Figure 2**).

The profiling report automatically opens in Visual Studio and displays the performance investigation results in the Summary view, which is shown in **Figure 3**.

## Profiling Data Analysis

Not all synchronization causes lock contention. If a lock is available, an attempt to take an ownership of the lock does not block thread execution and no contention happens. In Resource Contention Profiling mode, the profiler collects data only for synchronization events that cause contention and does not report successful (unblocked) resource acquisitions. If your application does not cause any contentions, no data will be collected. If you get data, it means your application has lock contentions.

For each contention, the profiler reports which thread was blocked, where the contention occurred (resource and call stack), when the contention occurred (timestamp) and the amount of time (length) that the thread was blocked trying to acquire a lock, enter a critical section, wait for a single object, and so on.

When you open the file, you'll first see the Summary view (**Figure 3**), with three main areas you can use for brief diagnostics: 1. The contentions chart shows the number of contentions per second plotted for the lifetime of your application. You can visually inspect contention spikes or select a time interval and either zoom into it or filter the results. Filtering re-analyzes the data and removes data outside the selected interval.

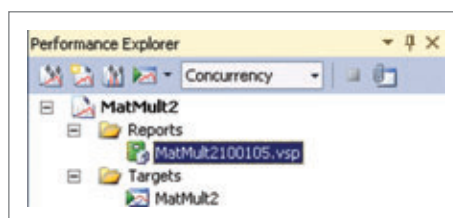


Figure 2 Performance Profiling Result File in Performance Explorer

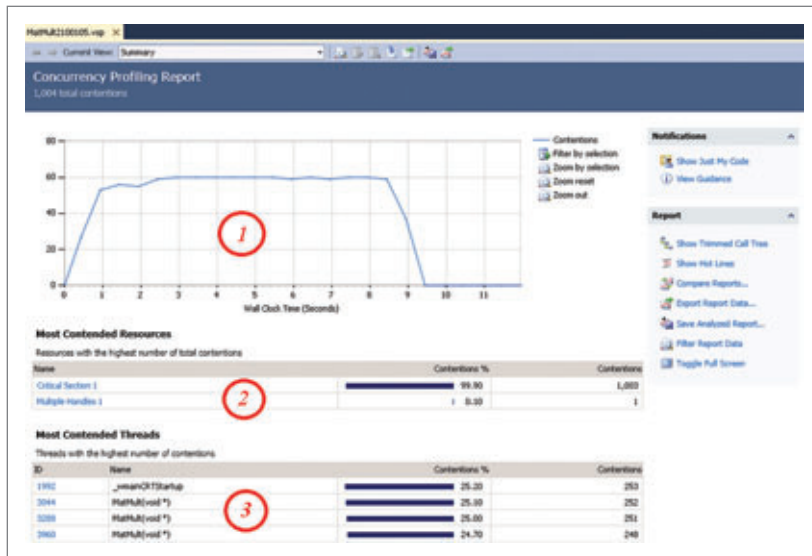


Figure 3 Summary View of the Profiling Report

2. The Most Contended Resources table lists the resources that caused the most detected contentions.
3. The Most Contended Threads table lists the threads with the highest number of contentions. This table uses the number of contentions as a criterion, not the length of the contentions. Therefore, you might have a thread that's blocked in a single contention for a long time, but it won't be displayed in Summary view. On the other hand, a thread that experienced many very short contentions, with each contention blocking the thread for only a very short time, would be presented in the Summary view. If you see a resource that's responsible for the majority of contentions, inspect that resource in more detail. If you observe a thread that experiences a large number of contentions you did not expect, inspect the contentions of the thread.

For example, in **Figure 3**, you can see that Critical Section 1 is responsible for nearly all (99.90 percent) contentions in that application. Let's investigate that resource more closely.

Resource names and thread IDs on the Summary view are hyperlinks. Clicking on Critical Section 1 transfers you to the Resource Details view (see **Figure 4**), where the context is set to the specific resource—Critical Section 1.

## Resource Details

The upper part of the Resource Details view shows a time-based chart where each horizontal line belongs to a thread. The lines are labeled by the thread root function unless you name the managed thread in your code (for example, by using the C# `System.Threading.Thread.Name` property). A block on this line represents a contention of the thread on the resource. The block length is the contention length. Blocks from different lines might overlap in time, which means several threads blocked on the resource at the same time.

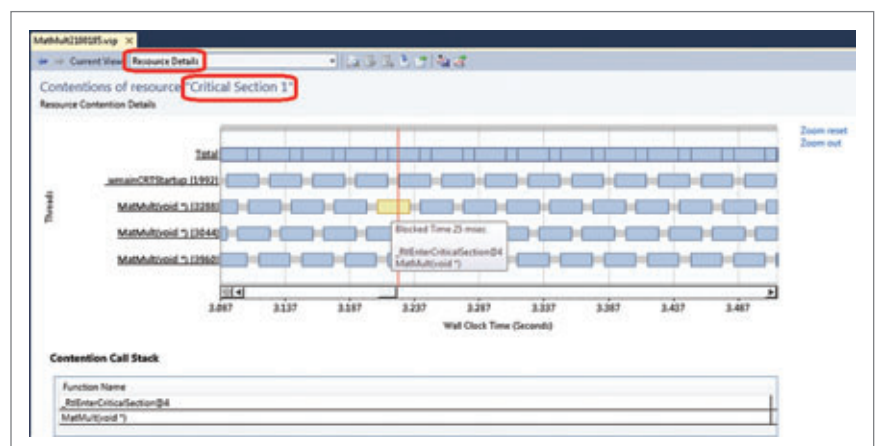


Figure 4 Resource Details View

The Total line is special. It doesn't belong to any specific thread, but contains all contentions of all threads on this resource (it is actually a projection of contention blocks to the line). As you can see, Critical Section 1 was quite busy—it doesn't seem to have any empty slots on its Total line.

You can zoom into a specific portion of the chart by selecting a time range using the left mouse button (left-click at the point in the chart you want to start and then drag the pointer to the right). There are two links on the upper-right part of the chart—Zoom reset and Zoom out. Zoom reset restores the original chart view. Zoom out takes you back step by step, un-zooming the chart the same way you zoomed in.

The overall pattern of contention blocks might lead you to some conclusions about your application execution. For example, you can see that contentions of various threads are heavily overlapped in time, which hints at a less than optimal parallelization.

Each thread is blocked on the resource much longer than it's running, and it's yet another indication of the application's inefficiency.

The overall pattern of contention blocks might lead you to some conclusions about your application execution.

## Function Details

The bottom part of the Resource Details view is a contention call stack—no data is displayed until you select a specific contention. When you select a block, the corresponding stack shows up in the bottom panel. You can also hover over a contention block on the chart without clicking on it, and a pop-up window will give you the stack and the contention length.



# We didn't invent the Internet...

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



applications

powered by 

connectivity

powered by 

## 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.

To learn more please visit our website →

[www.nsoftware.com](http://www.nsoftware.com)

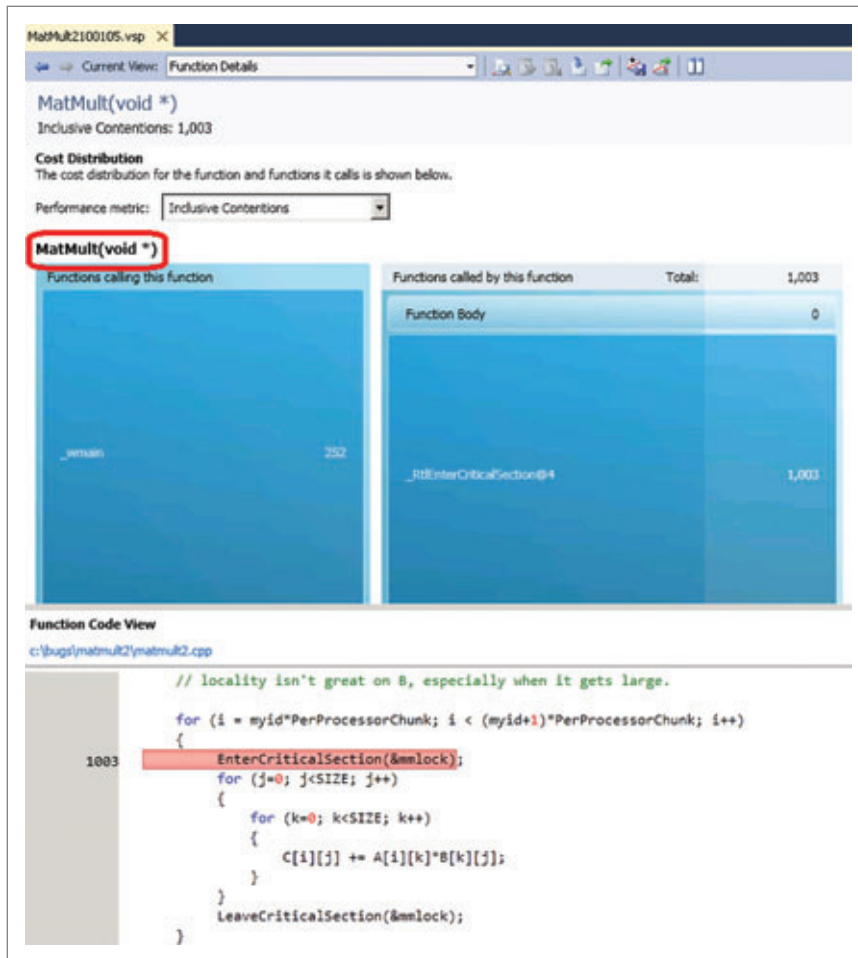


Figure 5 Function Details View

As you can see in the contention call stack, one of the example app functions called MatMult is listed, so you know it was the cause of the contention. To determine which line of the function code is responsible for the contention, double-click the function name in the call stack panel. That takes you to the Function Details view, shown in Figure 5.

In this view you see a graphical presentation of functions that called MatMult, as well as functions that were called inside of it. The bottom section of the view clearly shows that EnterCriticalSection(&mmlock) is responsible for the threads being blocked all the time.

When you know which line of your code is responsible for contentions, you may reconsider your decision to implement synchronization that way. Is it the best way to protect your code? Is protection required at all?

In the example app, using a critical section in this code is unnecessary because threads don't share writes to the same result matrix rows. The Visual Studio performance tools bring you to the point where you can comment out the use of mmlock, significantly speeding up the application. If only it were always that easy!

For a more in-depth description of Function Details view, see the Visual Studio Profiler Team blog at [blogs.msdn.com/profiler/archive/2010/01/19/vs2010-investigating-a-sample-profiling-report-function-details.aspx](https://blogs.msdn.com/profiler/archive/2010/01/19/vs2010-investigating-a-sample-profiling-report-function-details.aspx).

## Thread Details

As I mentioned earlier, Summary view provides a good starting point for your investigation. By looking at the Most Contended Resources and Most Contended Threads tables, you can decide how to proceed. If you find that one of the threads looks suspicious because you didn't expect it to be in the top list of contended threads, you might decide to give the thread a closer look.

Click the thread ID on the Summary view to jump to the Thread Details view (see Figure 6). Though this view looks similar to the Resource Details view, it has a different meaning because it displays contentions in the context of the selected thread. Each horizontal line represents a resource the thread was contending for during the thread lifetime. On this chart you won't see contention blocks overlapping in time because that would mean the same thread was blocked on more than one resource at the same time.

Note that WaitForMultipleObjects (which I'm not showing here) is handled separately and is represented with a single chart line for the set of objects. This is because the profiler

treats all parameter objects of WaitForMultipleObjects as a single entity.

Any manipulations you can do in Resource Details view (zooming the chart in and out, selecting specific contentions and viewing the length in milliseconds, and the calling stack) are applicable to the Thread Details view as well. Double-click the function name in the Contention Call Stack panel to navigate to the Function Details view of that function.

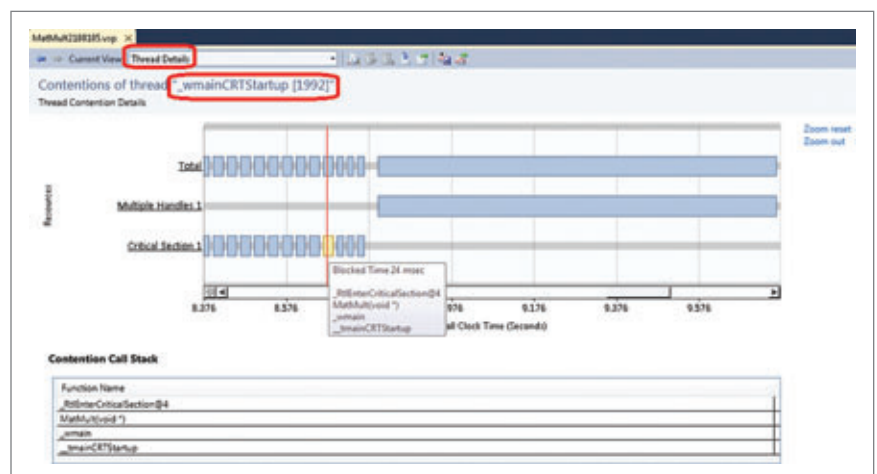


Figure 6 Thread Details View with Selected Contention Block



## Silverlight, .NET, WPF, WCF, WF, C API, C++ Class Lib, COM & more!

Develop your application with the same robust imaging technologies used by **Microsoft, HP, Sony, Canon, Kodak, GE, Siemens, the US Air Force and Veterans Affairs Hospitals.**

LEADTOOLS provides developers easy access to decades of expertise in color, grayscale, document, medical, vector and multimedia imaging development. Install LEADTOOLS to eliminate months of research and programming time while maintaining high levels of quality, performance and functionality.

- Image Formats & Compression:** Supports 150+ image formats and compressions including TIFF, EXIF, PDF, JPEG2000, JBIG and CCITT.
- Display Controls:** ActiveX, COM, Win Forms, Web Forms, WPF and Silverlight.
- Image Processing:** 200+ filters, transforms, color conversion and drawing functions supporting region of interest and extended grayscale data.
- OCR/ICR/OMR:** Full page or zonal recognition for multithreaded 32 and 64 bit development.
- Forms Recognition and Processing:** Automatically identify forms and extract user filled data.
- Barcode:** Detect, read and write 1D and 2D barcodes for multithreaded 32 and 64 bit development.
- Document Cleanup/Preprocessing:** Deskew, despeckle, hole punch, line and border removal, inverted text correction and more.
- PDF and PDF/A:** Read and write searchable PDF with text, images and annotations.
- Annotations:** Interactive UI for document mark-up, redaction and image measurement (including support for DICOM annotations).
- Medical Web Viewer Framework:** Plug-in enabled framework to quickly build high-quality, full-featured, web-based medical image delivery and viewer applications.
- Medical Image Viewer:** High level display control with built-in tools for image mark-up, window level, measurement, zoom/pan, cine, and LUT manipulation.
- DICOM:** Full support for all IOD classes and modalities defined in the 2008 DICOM standard (including Encapsulated PDF/CDA and Raw Data).
- PACS Communications:** Full support for DICOM messaging and secure communication enabling quick implementation of any DICOM SCU and SCP services.
- JPIP:** Client and Server components for interactive streaming of large images and associated image data using the minimum possible bandwidth.
- Scanning:** TWAIN 2.0 and WIA (32 and 64-bit), autodetect optimum driver settings for high speed scanning.
- DVD:** Play, create, convert and burn DVD images.
- DVR:** Pause, rewind and fast-forward live capture and UDP or TCP/IP streams.
- Multimedia:** Capture, play, stream and convert MPEG, AVI, WMV, MP4, MP3, OGG, ISO, DVD and more.
- Enterprise Development:** Includes WCF services and WF activities to create scalable, robust enterprise applications.

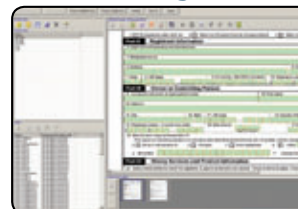
### Multimedia



### Barcode



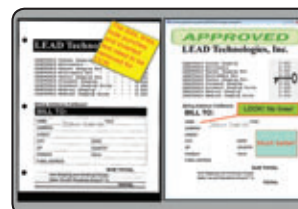
### Form Recognition & Processing



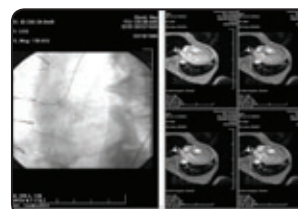
### Mark-up



### Document



### DICOM Medical



High Level Design • Low Level Control



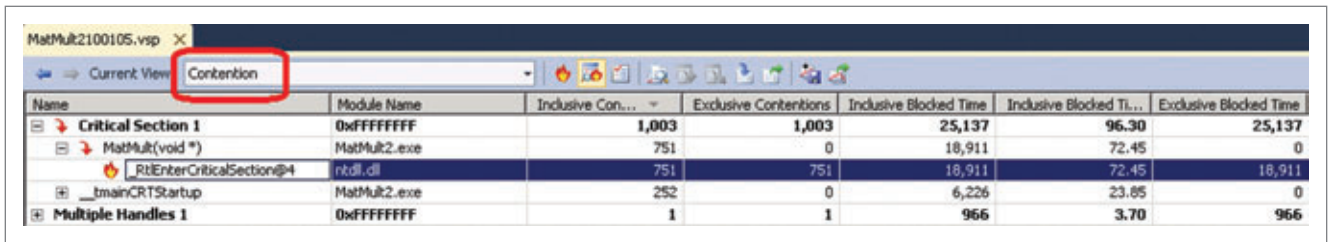


Figure 7 Contention View with Hot Path Applied to Critical Section 1

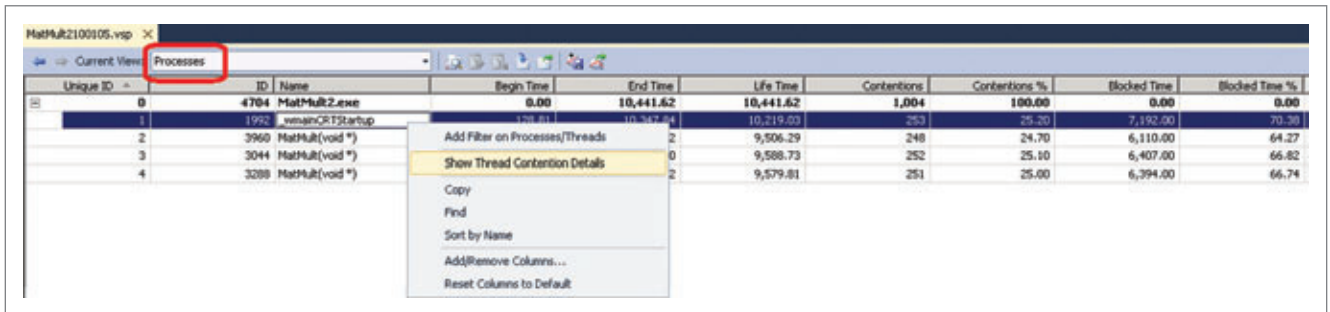


Figure 8 Processes View

In the example you can see that the thread spends more time being blocked than running in the early part of the execution, and then it's blocked for a long time on some set of multiple handles. As the last block is caused by waiting for other threads to complete, early contentions indicate non-optimal thread usage, causing the thread to be in a blocked state more than in an executing state.

## Chasing Down the Problem

As you might have noticed, the chart axis labels are hyperlinks. This allows switching between detailed views of resources and threads, setting the required context for the view each time. This can be helpful in an iterative approach to finding and solving a problem. For example, you can inspect resource R1 that blocked many threads. You can go from the Resource Details view to a

detailed view of thread T1 and find out that it was blocked not only on R1, but sometimes also on resource R2. You can then dive into the details of R2 and observe all threads that were blocked by R2. Next you can click on the label of thread T2 that draws your attention to check all resources that blocked T2, and so on.

Contention profiling data won't give you an explicit answer to the question of who is holding a lock at any given time. But given fair use of synchronization object between threads, and your knowledge of the application's behavior, you can identify a possible lock owner (a thread that succeeded in synchronization lock acquisition) by pivoting your data from resource details to thread details and back.

For example, suppose in the Thread Details view you see a thread T that's blocked on resource R at time t. You can switch to the Resource Details view of R by clicking on the R label, and see all threads that were blocked on R during the application lifetime. At time t you'll see a number of them (including T) blocked on R. A thread that's not blocked on R at time t is a possible lock holder.

I noted earlier that the chart's Total line is a projection of all contention blocks. The Total label is also a hyperlink, but from Resource Details view it takes you to the Contention view (see Figure 7), which is a collection of contention call trees per resource. The hot path of the appropriate resource call tree is activated for you. This view shows contentions and blocking-time statistics for each resource and for each node (function) in the resource call tree. Unlike the other views, this one aggregates contention stacks to the resource call tree, just like in other profiling modes, and gives you statistics for the whole run of the application.

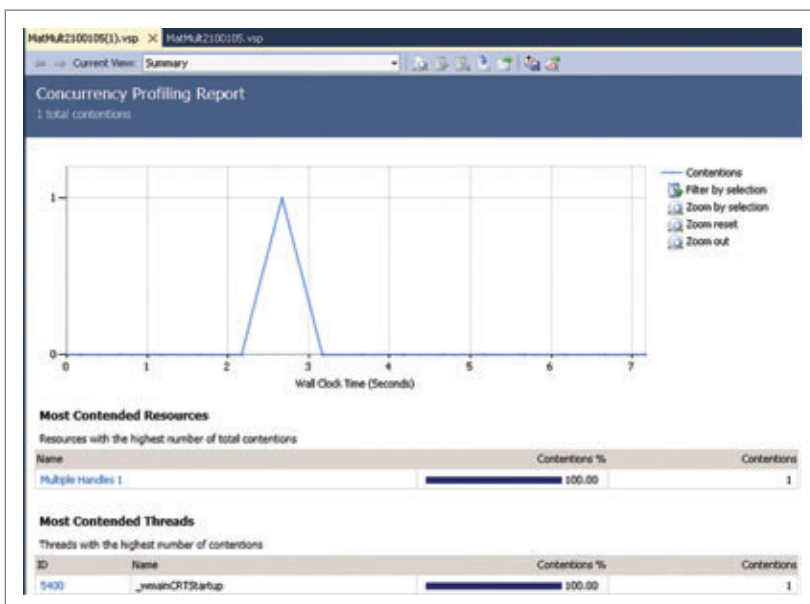


Figure 9 Summary View of Profiling Results for Fixed Code



Now v4.0!

# Why is Amyuni PDF so interesting?

## Proven

Choose a PDF technology that is integrated into thousands of applications behind millions of desktops worldwide.

## High-Performance

Develop with the fastest PDF conversion on the market, designed to perform in multithreaded and 64-bit Windows environments.

## Rapid Integration

Integrate PDF conversion, creation and editing into your .NET and ActiveX applications with just a few lines of code.

## Expertise

Produce accurate and stable PDF documents using reliable tools built by experts with over ten years of experience.

## OEM Licenses

License and distribute products quickly and easily with a PDF technology that does not rely on external open-source libraries.

## Customization

Let our experienced consultants help you turn your software requirements into customized PDF solutions.



We understand the challenges that come with PDF integration. From research and development, through design and implementation, we work with you every step of the way.

Get 30 days of FREE technical support with your trial download!

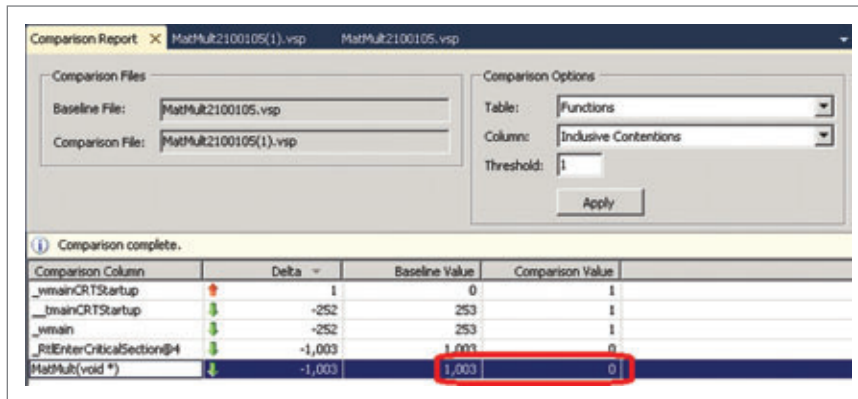
# www.amyuni.com

All trademarks are property of their respective owners. © 1999-2009 AMYUNI Technologies. All rights reserved.

USA and Canada  
Toll Free: 1 866 926 9864  
Support: (514) 868 9227  
Info: sales@amyuni.com

Europe  
Sales: (+33) 1 30 61 07 97  
Support: (+33) 1 30 61 07 98  
Customizations: management@amyuni.com

**AMYUNI**   
Technologies



Explorer (select one file, press Shift or Ctrl and then select another), then right-click and select Compare Performance Reports.

A Comparison Report appears, as shown in **Figure 10**. In the example app, you can see that number of Inclusive Contentions of the MatMult function dropped from 1,003 to 0.

## Alternative Data Collection Methods

If you create your performance session for either Sampling or Instrumentation profiling, you can always convert it later to Concurrency mode. One way to do it quickly is to use the

Figure 10 Comparison Report

From the Contention view, you can go back to the Resource Details view of any resource using the context menu. Point to a resource, right-click your mouse and select Show Contention Resource Details. Other interesting actions are also available in the context menu. As a general suggestion, explore the context menus in Profiler views—they can be quite helpful!

Click the Total label of the Thread Details view to display the Processes view, where the thread is selected (see **Figure 8**). In this view you can see when the thread was started relative to the application start time, when it was terminated, how long it ran, how many contentions it experienced, and how long it was blocked across all contentions (in milliseconds, and in percentage of the thread's lifetime).

Again, it is possible to come back to the Thread Details view for any thread by using the context menu—select a thread of interest, right-click and select Show Thread Contention Details.

Another possible investigation flow would be to display the Processes view directly when the file is opened, sort the threads by clicking on the title of one of the available columns (for example, sorting threads by the number of contentions), select one of the threads and then switch to the contention details chart of the thread through the context menu.

## Fix the Problem and Compare Results

After you find the root cause of the lock contentions in the application, you can comment out the mmlock critical section and then rerun profiling:

```
for (i = myid*PerProcessorChunk;
     i < (myid+1)*PerProcessorChunk;
     i++) {
    // EnterCriticalSection(&mmlock);
    for (j=0; j<SIZE; j++) {
        for (k=0; k<SIZE; k++) {
            C[i][j] += A[i][k]*B[k][j];
        }
    }
    // LeaveCriticalSection(&mmlock);
}
```

You would expect the number of contentions to decrease, and indeed profiling of the modified code reports only one lock contention, as shown in **Figure 9**.

We can also compare the new and previous performance results in Visual Studio. To do this, select both files in Performance

profiling mode menu in the Performance Explorer. Just select the mode you'd like to be in, and you're good to go.

You can also go through the Properties setting of your session. Point to your session in Performance Explorer, right-click to display the context menu, then select Properties. The General tab of Property Pages gives you control over your profiling session mode and other profiling parameters.

Summary view provides  
a good starting point for  
your investigation.

Once your profiling mode is set for Concurrency (or Sampling, for that matter), you can either launch your application (it's already in your Targets list if you used Performance Wizard, or you can add it there manually), or you can attach to an application that's up and running. Performance Explorer gives you the controls to do these tasks, as **Figure 11** shows.

The Visual Studio UI automates a number of the steps necessary to collect profiling data. However, it is possible to collect profiling data by using command-line tools, which can be useful for automated runs and scripts.

To launch your application in contention profiling mode, open the Visual Studio command prompt (which puts all profiler binaries in your path, either x86 or x64 tools), and then do the following:

1. VSPerfCmd.exe /start:CONCURRENCY,RESOURCEONLY /output:<YourOutputFile>
2. VSPerfCmd.exe /launch:<Your Application> /args:<"Your Application Arguments">
3. Run your scenario
4. VSPerfCmd.exe /detach
  - This step is not required if your application terminates, but it causes no harm so you can add it to your scripts.
5. VSPerfCmd.exe /shutdown

Now you can open YourOutputFile.VSP in Visual Studio for analysis.

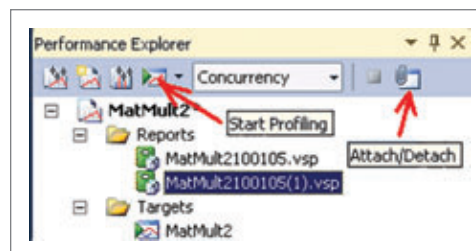


Figure 11 Profiling Controls of Performance Explorer



Figure 12 Analysis Views

View	Description
Summary	Summary information is presented to serve as a starting point for your investigation. This is the first view you see, and it opens automatically after a profiling session is over and the result file is ready.
Call Tree	An aggregated call tree of all contention stacks. Here you can see which stacks were responsible for your contentions.
Modules	A list of modules that contain functions, each resulting in a contention. Each module has a list of relevant functions and the number of detected contentions.
Caller/Callee	A three-panel view that presents function F, all functions that call F, and functions that are called by F (only calls resulted in contentions, of course).
Functions	A list of all detected functions on any contention stack, with associated data.
Lines	The function lines in the source code.
Resource Details	Details about a specific resource (for example, a lock), showing all threads that were blocked on it during the application lifetime.
Thread Details	Details about a specific thread, showing all resources (such as locks) the thread was blocked on.
Contention	Similar to the call tree view, but here call trees are separated per contention resource. In other words, this view presents a set of call trees, each containing stacks that were blocked on a specific resource.
Marks	A list of automatically and manually recorded marks, where each mark is associated with its timestamp and the values of Windows counters.
Processes	A list of inspected processes, where each process has a list of its threads, and each thread is attributed with the number of contentions it experienced and the summarized length of blocked time.
Function Details	Details about a specific function, including the functions it calls and collected data.
IPs	A list of instruction pointers where contention happened (well, a list of functions like EnterCriticalSection, WaitForSingleObject and so on, because this is where contention actually happens).

If you have an application that's running already, you can attach the profiler to it using these steps:

1. VSPerfCmd.exe /start:CONCURRENCY,RESOURCEONLY /output:<YourOutputFile>
2. VSPerfCmd.exe /attach:<PID or Process Name>
3. Run your scenario
4. VSPerfCmd.exe /detach
5. VSPerfCmd.exe /shutdown

A more detailed explanation of available command-line options can be found at [msdn.microsoft.com/library/bb385768\(VS.100\)](http://msdn.microsoft.com/library/bb385768(VS.100)).

## Other Analysis Data Views

A variety of Visual Studio views let you closely inspect the collected data. Some views give a picture of the application lifetime in whole, while others focus on specific contentions—use those you find most valuable.

When you analyze the results of profiling, you can use transitions from one view to another through hyperlinks, double-clicks or con-

text menus, or you can switch directly to any available view through a drop-down menu. **Figure 12** briefly describes each of the views.

It is possible to collect profiling data by using command-line tools, which could be useful for automated runs and scripts.

The new resource contention profiling features in Visual Studio should help you discover performance issues by using thread synchronization in the code, and allow you to improve your application runtime by changing, reducing or eliminating unnecessary synchronization. ■

**MIKE GOLDIN** is a senior software design engineer at Microsoft. He has worked on the Visual Studio Engineering team since 2003. He can be reached at [mgoldin@microsoft.com](mailto:mgoldin@microsoft.com), and he blogs at [blogs.msdn.com/mgoldin](http://blogs.msdn.com/mgoldin).

**THANKS** to the following technical experts for reviewing this article:

Steve Carroll, Anna Galaeva, Daryush Laqab, Marc Popkin-Paine, Chris Schmich and Colin Thomsen

## Build interactive diagrams easier than you ever imagined

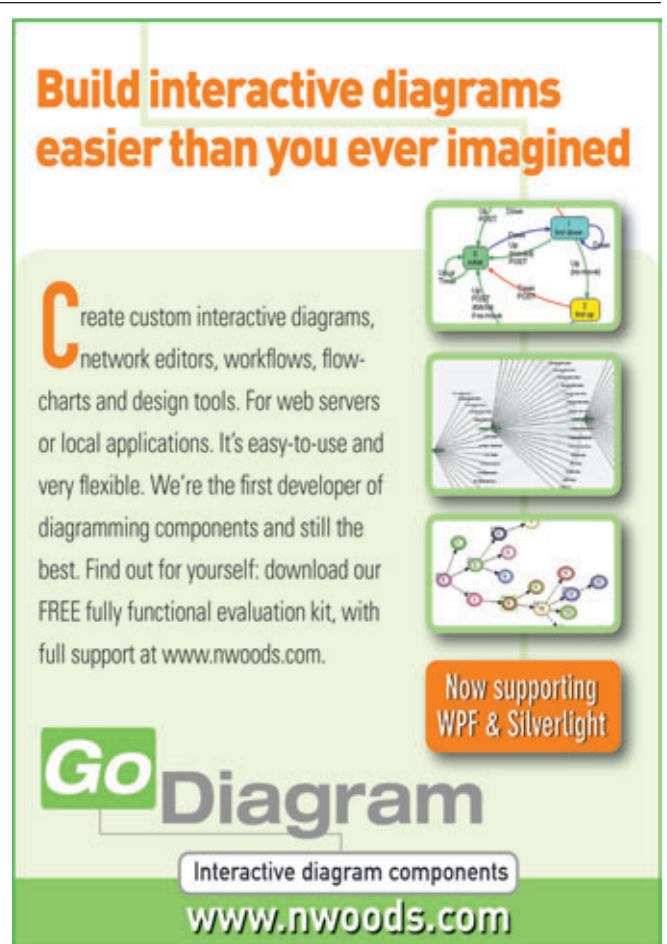
Create custom interactive diagrams, network editors, workflows, flowcharts and design tools. For web servers or local applications. It's easy-to-use and very flexible. We're the first developer of diagramming components and still the best. Find out for yourself: download our FREE fully functional evaluation kit, with full support at [www.nwoods.com](http://www.nwoods.com).

Now supporting WPF & Silverlight

# GoDiagram

Interactive diagram components

[www.nwoods.com](http://www.nwoods.com)





# XCEED DataGrid for WPF

## Serious recognition

### **Microsoft®** *Visual Studio® Team System 2010*

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

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

### **IBM®** *U2 SystemBuilder™*

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

**Vincent Smith,**  
*U2 Tools Product Manager at IBM*





# Datagrids, transformed

- › Display & edit data in **stunning 2D or 3D**
- › **Highest-performance** WPF datagrid
- › Most **adopted**, most **mature** WPF control
- › **150** features, **10** major releases in **3** years



The smooth-scrolling Tableflow™ view provides a rich, fluid, and high-performance experience. Its innovative group navigation control redefines datagrid usability.



The Cardflow™ 3D view lets you add a true 3D experience to your application. Also offered is a classic 2D card view.

Ship Country	ShipCity	OrderDate	Freight
USA	Albuquerque	03/19/1997	\$708.95
USA	Albuquerque	02/16/1998	\$280.61
USA	Albuquerque	03/26/1998	\$174.05
USA	Albuquerque	09/27/1996	\$150.15
USA	Albuquerque	08/30/1996	\$147.26
USA	Albuquerque	11/05/1996	\$142.08
USA	Albuquerque	08/02/1996	\$98.03
USA	Albuquerque	09/25/1996	\$74.16
USA	Albuquerque	03/31/1998	\$61.14
USA	Albuquerque	06/16/1997	\$58.98

The first WPF datagrid on the market and under constant development. Build apps you can trust in mission-critical situations.

Try it live on [xceed.com](http://xceed.com)

**XCEED**  
MULTI-TALENTED COMPONENTS

# Take Control of Logging and Tracing in Windows Azure

Mike Kelly

Like many programmers, when I first started writing code I used print statements for debugging. I didn't know how to use a debugger and the print statements were a crude but effective way to see what my program was doing as it ran. Later, I learned to use a real debugger and dropped the print statements as a debugging tool.

Fast forward to when I started writing code that runs on servers. I found that those print statements now go under the more sophisticated heading of “logging and tracing,” essential techniques for any server application programmer.

Even if you could attach a debugger to a production server application—which often isn't possible due to security restrictions on the machines hosting the application—the types of issues server applications run into aren't easily revealed with traditional

debuggers. Many server applications are distributed, running on multiple machines, and debugging what's happening on one machine isn't always enough to diagnose real-world problems.

Moreover, server applications often run under the control of an operations staff that wouldn't know how to use a traditional debugger—and calling in a developer for every problem isn't desirable or practical.

In this article I'll explain some basic logging, tracing and debugging techniques used for server applications. Then I'll dive into how these techniques can be employed for your Windows Azure projects. Along the way you'll see how logging and tracing are used with some real-world applications, and I'll show you how to use Windows PowerShell to manage diagnostics for a running service.

## This article discusses:

- Logging, tracing and debug output
- Tracing and logging in Windows Azure
- Selectively enabling tracing and logging
- Managing logging for a running service

## Technologies discussed:

Windows Azure, Visual Studio 2010, Windows PowerShell

## Code download available at:

[code.msdn.microsoft.com/mag201006Azure](http://code.msdn.microsoft.com/mag201006Azure)

## A Logging Strategy

Ideally, any server application—and basically any Web application, including those running under Windows Azure—has a logging and tracing strategy designed in from the beginning. The logging information should be robust enough to describe nearly everything that's happening within each component. However, just as those print statements I added to my first programs could produce a lot of output, so too can logging. Well-designed logging and tracing thus includes ways of adjusting the type and volume of logging from any component. This allows operational staff and developers to focus on a particular misbehaving component, perhaps even on a particular machine, to get much more information on exactly



what's happening inside it—without generating a lot of noise in the log that can be distracting and perhaps slow down the application significantly.

Furthermore, because server applications are commonly distributed applications, information must be collected and aggregated from multiple machines (perhaps in different application roles) to get a full picture of what was going on when a particular problem occurred. So a way to identify a transaction thread through the machines is important, allowing the aggregation after the fact.

The logging available in Windows Azure has matured through the Community Technology Preview (CTP) releases. The early logging wasn't much more sophisticated than a print statement, captured as text in Windows Azure table storage. Starting with the PDC09 release, Windows Azure began to offer a much more full-featured logging and tracing infrastructure, based on the Event Tracing for Windows (ETW) framework.

This ETW framework is supported in ASP.NET through classes in the System.Diagnostics namespace. The Microsoft.WindowsAzure.Diagnostics namespace, which inherits from and extends standard System.Diagnostics classes, enables the use of System.Diagnostics as a logging framework in the Windows Azure environment. **Figure 1** shows how ETW is implemented by Windows Azure Diagnostics.

ETW provides a model in which code logs to one or more TraceSources. The level of logging allowed through each source is controlled by a SourceSwitch. Sources are in turn connected to one or more consumers, which persist the logging information in different ways.

Windows Azure provides a standard consumer or listener to persist the logging information you generate either to Windows Azure table storage or to blob storage. You also can write your own consumer if you want to do something different with the event data, or use an off-the-shelf consumer (although some might have to be modified to work in the Windows Azure environment).

The ETW framework associates a TraceEventType with each event, as shown in **Figure 2**. The first five severity rows are the values most commonly used and they indicate the relative importance of the trace output. Note that the Suspend, Resume and Transfer types are used by Windows Communication Foundation (WCF).

If you're looking only for really bad things, you'll want to be sure to capture Critical and probably Error values. If you want lots of information about what's going on, look at everything above Verbose.

Your logging strategy should include consistent use of the event type and copious logging entries with the values further down the hierarchy. It should be possible to virtually follow the execution flow

of your application if logging for all the highlighted values is enabled in your application. This can be invaluable in troubleshooting an error or problem in production.

You can hook up listeners, sources and switches to enable different levels of output programmatically, but this is typically done through configuration files. You can configure the output in app.config (for Windows Azure worker roles) or web.config (for Windows Azure Web roles). However, as I'll explain in detail later in the article, putting this in ServiceConfiguration.cscfg lets you adjust logging and tracing options while the Windows Azure service is running, without having to redeploy any updates to the running code or to even stop the service. Windows Azure also exposes a RESTful interface to allow control over some logging options remotely. Windows PowerShell can make use of the RESTful interface.

Well-designed logging and tracing includes ways of adjusting the type and volume of logging.

## Logging, Tracing and Debug Output

The terms logging, tracing and debug output can sometimes be used interchangeably. In this article, I'm going to distinguish among four different types of what can generally be called *diagnostic output* in your code. These are roughly ordered from most verbose to least verbose.

- *Debug Output*: This is information that appears only in the debug builds of your application and is excluded at compile time from release builds (based on whether the DEBUG preprocessor

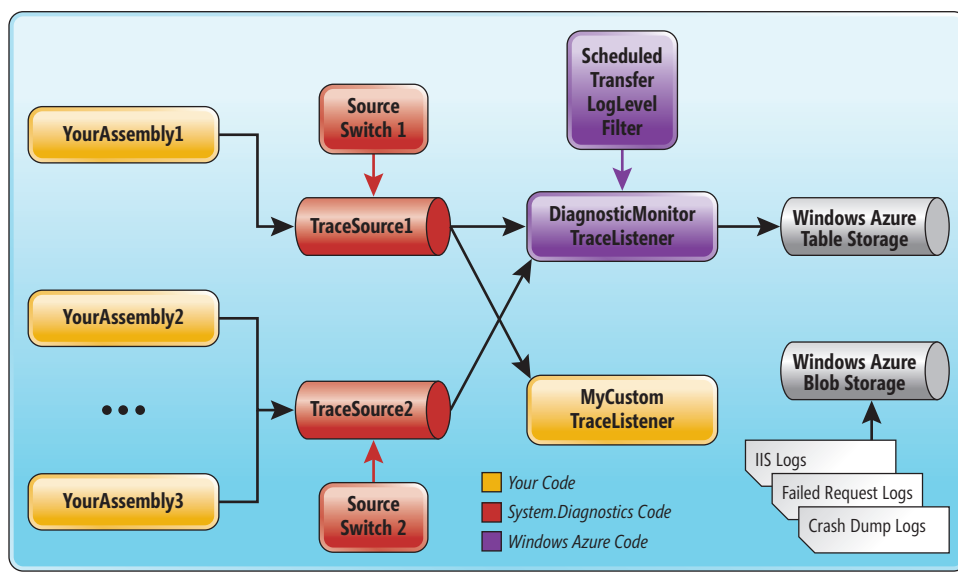


Figure 1 High-Level Overview of Windows Azure Diagnostics

Figure 2 Trace Event Types

TraceEventType	Value	Meaning
Critical	0x0001	Fatal error or application crash
Error	0x0002	Recoverable error
Warning	0x0004	Non-critical problem—may be an indication of more serious problems to come
Information	0x0008	Informational message
Verbose	0x0010	Debugging trace (such as detailed execution flow information, parameters, and so forth)
Start	0x0100	Starting of a logical operation
Stop	0x0200	Stopping of a logical operation
Suspend	0x0400	Suspension of a logical operation
Resume	0x0800	Resuming a logical operation
Transfer	0x1000	Transfer to a new activity

symbol is defined at compile time, which Visual Studio by default defines only in debug builds). Typically, debug output includes things like Asserts that you add to help find cases where code is not complying with expected preconditions, leading to bugs, or even dumps of data structures. Adding these helps you debug algorithms during debugging and testing.

The terms logging, tracing and debug output can sometimes be used interchangeably.

- **Tracing:** These are statements that are intended to help track the flow of control and state of the program as it's executing. Imagine running a debugger, stepping along through code and checking the values of key variables in the Watch window. Tracing statements are intended to replicate that experience in cases where you can't attach a debugger. They should ideally provide enough context that you can see which path is taken at each control point in the application and sort of follow along in the code from reading the trace statements. Tracing is enabled when the TRACE preprocessor symbol is defined at compile time, and can be in both release and debug builds. (By default, Visual Studio defines TRACE in both debug and release builds, but you can of course change this.)
- **Event Logging:** These are statements that are intended to capture

major events in the course of running the application—for instance, the start of a transaction or the addition of an item to a database. Event logging is different from tracing in that it captures major states rather than detailed flow of control.

- **Error Logging:** These are special cases of event logging in which you capture exceptional or potentially dangerous situations. Examples include any caught exception; cases where you can't access a resource on another machine you expect to be able to access (and which, of course, your application will gracefully handle but would like to note); and cases where errors come back from APIs from which you don't expect errors.

Error logging can also be useful to operations staff in situations where problems aren't yet occurring, but indications are that they soon will—for instance, a quota that is nearing its maximum or a transaction that is succeeding but taking more time than usual. These sorts of logging events can help operations staff proactively address problems before they occur, avoiding downtime in the application.

Most good developers have gotten used to including debug output in their applications to help them diagnose problems during development, and many have developed some sort of solution for error logging.

However, you need to be sure you're considering not just the debug output and error logging options, but also have a robust strategy for tracing and event logging, which can really help diagnose problems that occur only under stressful loads in production environments.

Also, carefully consider whether much of what you now think of as debug output shouldn't instead be tracing and available in both release and debug builds. A misbehaving application in production will typically be running the release build. If you have the tracing statements present but disabled (as I'll explain how to do later), you can selectively enable them to get very rich debug-like output from the release build, helping you diagnose problems.

## Tracing and Logging in Windows Azure

You can use Windows Azure logging right out of the box—it's part of the Windows Azure SDK. There are some advantages to using a logging framework like Logger.NET, Enterprise Library, log4net or Ukadc.Diagnostics. These add additional structure to your logging messages and also can help provide some of the configurability mentioned earlier. However, most have not been tweaked to work in the Windows Azure environment and some are much more than just a logging framework.

For the sample code for this article, I decided to use just the standard Windows Azure logging and tracing APIs with a thin

PartitionKey	RowKey	Timestamp	EventTickCount	DeploymentId	Role	RoleInstance	Level	EventId	Pid	Tid	Message
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: LoggingWorkerRole entry point called
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working
DE3399463443000000	deployment(KI)___LoggingWorker	2010-01-18 17:15:29	63399463443395421	deployment(KI)	LoggingWorkerRole	deployment(KI)MySampleLogg	5	0	7004	6952	Information: Working

Figure 3 Logs Persisted in Development Storage

Figure 4 Configuring Trace Sources and Listeners

```
<configuration>
  <system.diagnostics>
    <sharedListeners>
      <add type="Microsoft.WindowsAzure.Diagnostics.
DiagnosticMonitorTraceListener, Microsoft.WindowsAzure.Diagnostics,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
name="AzureDiagnostics">
        <filter type="" />
      </add>
    </sharedListeners>
    <sources>
      <source name="ConfigTrace">
        <listeners>
          <add name="AzureDiagnostics" />
        </listeners>
      </source>
      <source name="WorkerTrace">
        <listeners>
          <add name="AzureDiagnostics" />
        </listeners>
      </source>
    </sources>
    <switches>
      <add name="ConfigTrace" value="Verbose"/>
      <add name="WorkerTrace" value="Error"/>
    </switches>
    <trace>
      <listeners>
        <add name="AzureDiagnostics" />
      </listeners>
    </trace>
  </system.diagnostics>
</configuration>
```

layer on top to provide dynamic configuration. You will probably find building some helper classes and a framework for your logging and tracing strategy on top of this helpful, or watch for the other frameworks to provide Windows Azure versions.

When you create a new service in Windows Azure using Visual Studio, it's already enabled to do basic logging. The boilerplate Worker Role and Web Role code generated by the Windows Azure templates has the diagnostics listener already configured and enabled.

To enable simple logging in a Windows Azure service, start a new project in Visual Studio using the Windows Azure Cloud Service (Visual C#) template. Give the project a name. For my sample I used MSDNSampleLoggingService. Click OK.

The New Cloud Service Project wizard will run. In the wizard, add one Worker Role and one Web Role to the project. Rename the Worker Role to LoggingWorkerRole and the Web Role to LoggingWebRole, then click OK. Visual Studio will create the project.

At this point, you can explore the generated code. Look at the app.config in the LoggingWorkerRole project and note that the following code appears:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>
    <trace autoflush="false" indentsize="4">
      <listeners>
        <add name="AzureDiagnostics" type="Microsoft.WindowsAzure.Diagnostics.
DiagnosticMonitorTraceListener, Microsoft.WindowsAzure.Diagnostics,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
      </listeners>
    </trace>
  </system.diagnostics>
</configuration>
```

This hooks up the standard Windows Azure diagnostics listener to your code, meaning that any logging and tracing you do from the worker role will be directed to the Windows Azure

listener (DiagnosticMonitorTraceListener) unless you change this. You'll find a similar entry in the web.config for the Web Role created for this service.

If you look at the WorkerRole.cs file in the worker role project, you'll also see this line in the OnStart method:

```
DiagnosticMonitor.Start("DiagnosticsConnectionString");
```

And in the Run method, you'll see a call to trace:

```
// This is a sample worker implementation. Replace with your logic.
Trace.WriteLine("LoggingWorkerRole entry point called", "Information");
```

Finally, if you look in the ServiceConfiguration.cscfg file in the Service root, you'll see this line for both the Worker Role and the Web Role:

```
<Setting name="DiagnosticsConnectionString"
value="UseDevelopmentStorage=true" />
```

You can use Windows Azure logging right out of the box.

This tells the Windows Azure listener which storage account to use for persisting the logging and tracing information. In this case, the logging information will be stored in development storage on your local machine. By switching this to a Windows Azure cloud-storage account, you can have the logs go to cloud storage. Here is an example of the format for that from the sample code provided with this article:

```
<Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=Xxxxxxx;AccountKey=Yyyyyyy" />
```

The AccountName and AccountKey values need to be customized to your particular Azure account and key. You get this information from the storage account portal for your service at windows.azure.com. AccountName is the first portion of the URL for the table and blob storage endpoints (the part before ".table.core.windows.net"). AccountKey is the base-64 encoded Primary Access Key for your storage account.

Note that because there is a distinct storage account used for diagnostics, you can choose to store your diagnostics information separate from your other application data. To do this, set up a separate storage account by clicking New Service on the portal page,

Figure 5 Tracing Source Levels and TraceEventType

SourceLevel	Value	System.Diagnostics.TraceEventType									
		Transfer	Resume	Suspend	Stop	Start	Verbose	Information	Warning	Error	Critical
Off	0x0										
All	0xFFFF	X	X	X	X	X	X	X	X	X	X
Critical	0x1										X
Error	0x3									X	X
Warning	0x7								X	X	X
Information	0xF							X	X	X	X
Verbose	0x1F						X	X	X	X	X
ActivityTracing	0xFF00	X	X	X	X	X					



Figure 6 Standard Azure Logging Options

Type of Log	Windows Azure Storage Format	Collected by Default?	Notes
Windows Azure logs generated from your code	Table	✓	Trace listener must be added to web.config or app.config file as shown in the sample code. Stored in WADLogsTable.
IIS 7.0 logs	Blob	✓	Web Roles only. Stored in a Blob container under the path wad-iis-logfiles\<deployment ID>\<web role name>\<role instance>\W3SVC1.
Windows Diagnostic Infrastructure logs	Table	✓	Information about the diagnostics service itself. Stored in WADDiagnosticInfrastructureLogsTable.
Failed request logs	Blob		Web Roles only. Enable by setting tracing options under system.WebServer settings in web.config. Stored in a blob container under the path wad-iis-failedreqlogfiles\<deployment ID>\<web role name>\<role instance>\W3SVC1.
Windows Event logs	Table		Enable by altering DiagnosticMonitor Configuration.WindowsEventLog when setting up initial configuration. Stored in WADWindowsEventLogsTable. Steve Marx's blog (blog.smarx.com/posts/capturing-filtered-windows-events-with-windows-azure-diagnostics) explains how to use this.
Performance counters	Table		Enable by altering DiagnosticMonitor Configuration.PerformanceCounters. Stored in WADPerformanceCountersTable. The sample code Worker Role sets up a performance counter.
Crash dumps	Blob		Enable by calling CrashDumps.EnableCollection. Stored in a blob container under the path wad-crash-dumps. Because ASP.NET handles most exceptions, this is generally useful only for a Worker Role.
Custom error logs	Blob		Beyond the scope of this article, but see Neil Mackenzie's blog (nmackenzie.spaces.live.com/blog/cns!B863FF075995D18A!537.entry) for a helpful example of how to use this.

selecting Storage Account, then giving it a name (MyAppLogs, for instance). You might want to set up an affinity group so the storage for your logs is in the same region as your service.

Now that you've taken a quick tour of the tracing code in Windows Azure services, you can run the simple Web Role project you've created. Note that the default listener provided by Windows Azure also directs the output to the Output window in Visual Studio for debug builds, so you'll see the OnStart message appear in the debug window:

```
Information: LoggingWorkerRole entry point called
```

What if you want to look at logs after the service runs? By default, Windows Azure does not persist the logs to storage. You can tell it to do so by adding a few lines of code to your role's OnStart method:

```
TimeSpan tsOneMinute = TimeSpan.FromMinutes(1);
DiagnosticMonitorConfiguration dmc =
DiagnosticMonitor.GetDefaultInitialConfiguration();

// Transfer logs to storage every minute
dmc.Logs.ScheduledTransferPeriod = tsOneMinute;
// Transfer verbose, critical, etc. logs
dmc.Logs.ScheduledTransferLogLevelFilter = LogLevel.Verbose;
// Start up the diagnostic manager with the given configuration
DiagnosticMonitor.Start("DiagnosticsConnectionString", dmc);
```

Adding this code to WorkerRole.cs and rerunning will cause Windows Azure to transfer the logs to development storage every minute. You also can choose to do an on-demand transfer of logs (see the code in admin.aspx.cs in my sample app for how to do this) or use the Windows PowerShell commands described later in this article. Remember that once you transfer logs to storage, you'll be charged for the storage space and it's up to you to delete the information when it's no longer needed.

Once you've gotten the logs into Windows Azure storage, you'll need a tool to look at the storage tables to see the logs. I used Cerebrata's Cloud Storage Studio (cerebrata.com). Cerebrata has since come out with a tool called Azure Diagnostics Manager, and there are also free tools available on CodePlex (codeplex.com) for looking

at cloud storage and diagnostics. The logs are put in a table called WADLogsTable, which you can see in Figure 3.

You'll notice a couple of things when you look at the logs in storage. First, Windows Azure automatically associates some information with each logged event: a timestamp, a tick count (which provides more detailed timing with 100-nanosecond granularity), and information about the deployment, role and role instance. This allows you to narrow down logs to specific instances if you want.

Second, there's a Level and an EventId associated with each event. Level corresponds to the values in Figure 2—those Trace events logged as Information will have a Level value of 4, while those logged as Error will have a Level of 2. Generic events sent through Trace.WriteLine (as the boilerplate code does) will have Level 5 (Verbose).

The EventId is a value you specify. The basic Trace.WriteLine call shown earlier doesn't let you specify it; you have to use other Trace methods to pass the EventId.

Figure 7 Failed Request Logging for LoggingWebRole

```
<tracing>
  <traceFailedRequests>
    <add path="*">
      <traceAreas>
        <add provider="ASP" verbosity="Verbose" />
        <add provider="ASPNET"
          areas="Infrastructure,Module,Page,AppServices"
          verbosity="Verbose" />
        <add provider="ISAPI Extension" verbosity="Verbose" />
        <add provider="WWW Server"
          areas="Authentication,Security,Filter,StaticFile,CGI,Compression,Cache,RequestNotifications,Module"
          verbosity="Verbose" />
      </traceAreas>
      <failureDefinitions timeTaken="00:00:15" statusCodes="400-599" />
    </add>
  </traceFailedRequests>
</tracing>
```



### Speedy Dry Cleaning

★★★★★  
Fast and friendly service!

### The Auto Co.

★★★★★

Sponsored by

**Microsoft**



**EPSON**

*small business*

# BIG IMPACT

**Nominate Someone Today!  
Over \$30,000 in prizes!**



### Betty's Bake Shop

★★★★★  
Great selection!

### PhotoSnap

★★★★★

**ONE LUCKY SMALL BUSINESS WILL WIN A \$15,000 OFFICE MAKEOVER.  
ADDITIONAL WINNERS WILL BE ANNOUNCED BI-WEEKLY.**

Nominate and spread the word, every vote counts! Go to:

**[www.neweggbusiness.com/bigimpact](http://www.neweggbusiness.com/bigimpact)**



**One-Stop Shopping**



**Volume Savings**



**One Account  
Multiple Users**



**Lower Shipping Cost**



**Check Out Using  
Purchase Order**



**Availability Status**



Registration on [NeweggBusiness.com](http://NeweggBusiness.com) is Free

**Get Microsoft Visual Studio 2010 and Microsoft Server 2008 Today!**



32-116-871 **Visual Studio 2010 Professional w/MSDN \$1099.00**

32-116-866 **Visual Studio 2010 Professional Upgrade \$499.00**

32-116-813 **Windows Server Standard 2008 R2 64-bit 5 CLT \$899.99**

**[www.neweggbusiness.com/microsoft](http://www.neweggbusiness.com/microsoft)**

**ONCE YOU KNOW, YOU NEWEGG.®**

Follow us on:



©2000-2010 Newegg Inc. All rights reserved. Newegg is not responsible for errors and reserves the right to cancel orders arising from such errors. All third party logos are the ownership of the respective owner.

## Selectively Enabling Tracing and Logging

A typical application consists of multiple logical components. For instance, you might have a database component that deals with the data model in Windows Azure storage. Your Web Role might in turn be divided into an administrative component and a user component (and that might even be divided further into logical components based on the needs of your application).

You can tie the logging and tracing options—what type of logging is enabled and at what level of detail—to these components. This allows you to selectively enable tracing in only the components for which you need it, avoiding a lot of clutter.

The key approach here is to not call Trace directly, but to use multiple TraceSource instances, typically one per namespace. A TraceSource has an associated SourceSwitch that controls whether the source is enabled, as well as what level of output is desired. Importantly, the SourceSwitch values are not set at compile time, but at run time. As a result, you can enable or disable diagnostic output from various parts of your application without having to recompile it, or even redeploy a different version.

The key approach here is to not call Trace directly, but to use multiple TraceSource instances, typically one per namespace.

WorkerDiagnostics.cs and WebDiagnostics.cs contain the configuration of the trace sources and switches in the sample code. Here's an excerpt:

```
// Trace sources
public TraceSource ConfigTrace;
public TraceSource WorkerTrace;
// Add additional sources here

// Corresponding trace switches to control
// level of output for each source
public SourceSwitch ConfigTraceSwitch { get; set; }
public SourceSwitch WorkerTraceSwitch { get; set; }
// Add additional switches 1:1 with trace sources here
```

Then, in the config file for your role, you hook these up to listeners as shown in **Figure 4**. This first sets up the standard Windows Azure diagnostics listener as a shared listener so it can be referred to in the <sources> items. It then configures two sources: a WorkerTrace source

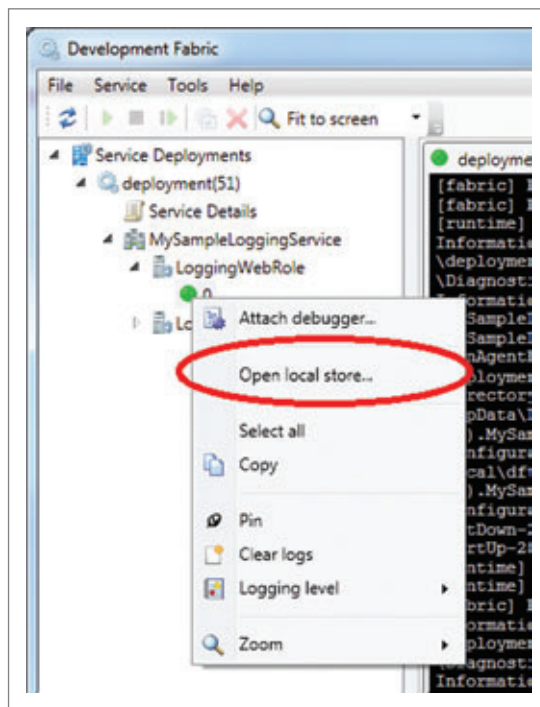


Figure 8 Opening Logs Saved to Local Development Fabric Storage

and a ConfigTrace source. It also sets up corresponding switches to enable you to adjust the level of output. ConfigTrace gives you the most verbose output; WorkerTrace gives you Errors only.

You don't have to name the switches the same as the sources, but it makes life easier. If they're not the same, you add a switchName attribute to the source element to indicate the name of the switch that controls output for this source. This allows you to share a single switch across multiple trace sources. Note that the trace source and switch names are case-sensitive and must exactly match the case you pass to the constructor in your code.

You can avoid the switch altogether if you want by simply adding a switch-Value attribute to the source element specifying the desired switch value for this source. The switch values you use are actually parsed from the config file as one of the SourceLevels defined in **Figure 5**, which also shows how the

TraceEventType you pass to TraceSource calls interacts with the SourceLevel set for the source to determine what passes through.

You might have noticed that the SourceLevel is just a bitmask that is ANDed at run time with the TraceEventType to determine whether to log the event. To get combinations like Warning and Activity Tracing, specify the numeric value for the bitfield as the switch value rather than using the symbolic values shown.

In addition to switches, a listener can have a TraceFilter, which adds more sophisticated runtime logic as to whether a particular message is allowed through. Writing a custom TraceFilter is beyond the scope of this article, but you'll find a helpful example in the Ukadc.Diagnostics project documentation on CodePlex ([ukadcdiagnostics.codeplex.com/wikipage?title=LoggingPrimer](http://ukadcdiagnostics.codeplex.com/wikipage?title=LoggingPrimer)).

## Changing What Is Logged at Run Time

This is the default way that ASP.NET tracing works and it will work fine with services deployed to Windows Azure. The problem is that you'd like to be able to change the switch values at run time, and Windows Azure doesn't allow you to just replace the web.config or app.config without redeploying the service. The generic ASP.NET solution for this is to use WebConfigurationManager to alter configuration values, but Windows Azure doesn't currently allow you to do this for cloud-deployed services, either.

The solution is to mirror the values for these switches in Service-Configuration.cscfg. Windows Azure lets you edit that file (or upload a new one) through the development portal while your service is running. You'll have to write some additional code to make this work, though.

The default System.Diagnostics code knows about settings only in app.config or web.config, but your roles will get run-time



# Thanks for Making DevExpress

# #1

WinForms  
Controls



ASP.NET AJAX  
Controls



Silverlight  
Controls



WPF  
Controls



Visual Studio  
Tools



Application  
Frameworks



AWARD-WINNING DEVELOPMENT TOOLS  
For a Free Trial Version Visit: [DevExpress.com/eval](http://DevExpress.com/eval)

Feature-Complete Presentation Components • Easy-to-Use Reporting Controls  
IDE Productivity Tools • Business Application Frameworks

**DevExpress**  
Download • Compare • Decide!



# Devexpress™







WinForms  
Controls



ASP.NET AJAX  
Controls



Silverlight  
Controls



WPF  
Controls



Visual Studio  
Tools



Application  
Frameworks



**AWARD-WINNING DEVELOPMENT TOOLS**  
For a Free Trial Version Visit: [DevExpress.com/eval](http://DevExpress.com/eval)

Feature-Complete Presentation Components • Easy-to-Use Reporting Controls  
IDE Productivity Tools • Business Application Frameworks

**DevExpress™**  
Download • Compare • Decide!



# Read more

## DevExpress awards and user comments

**DevExpress.com/awards**  
**DevExpress.com/comments**



**#1 PRODUCT**

ComponentSource® Awards 2009-10



**#1 PUBLISHER**

ComponentSource® Awards 2009-10



**AWARD-WINNING DEVELOPMENT TOOLS**  
**For a Free Trial Version Visit: [DevExpress.com/eval](http://DevExpress.com/eval)**

Feature-Complete Presentation Components • Easy-to-Use Reporting Controls  
IDE Productivity Tools • Business Application Frameworks

**DevExpress™**  
Download • Compare • Decide!

Figure 9 Windows Azure Management Diagnostics Cmdlets

Name	Description
Get-ActiveTransfers	Returns the set of active diagnostic transfers with associated transfer information.
Get-CommonConfigurationLogs	Gets the common configuration values for all logging buffers. This includes the time interval at which changes in configuration are polled for and the buffer size allocated for in-memory logs.
Get-DiagnosticAwareRoleInstances	Returns a list of IDs of active role instances that have a diagnostic monitor running.
Get-DiagnosticAwareRoles	Lists the set of roles that have successfully started at least one diagnostic monitor.
Get-DiagnosticConfiguration	Gets the buffer configuration for the specified buffer name (Logs, Directories, PerformanceCounters, WindowsEventLogs or DiagnosticInfrastructureLogs).
Set-CommonConfigurationLogs	Sets the common configuration values for all logging buffers.
Set-FileBasedLog	Sets the buffer configuration for file-based logs.
Set-InfrastructureLog	Sets the buffer configuration for the logs generated by the underlying Windows Azure diagnostics infrastructure. The diagnostic infrastructure logs are useful for troubleshooting the diagnostics system itself.
Set-PerformanceCounter	Sets the buffer configuration for performance counter data being collected by your service.
Set-WindowsAzureLog	Sets the buffer configuration for basic Windows Azure logs being generated by your service.
Set-WindowsEventLog	Sets the buffer configuration for Windows event logs being generated by your service.
Start-OnDemandTransfer	Starts an on-demand transfer of the specified data buffer. This moves the data to Windows Azure storage (either table or blob storage).
Stop-ActiveTransfer	Stops active on-demand transfer, given a transfer ID.

notification of changes in `ServiceConfiguration.cscfg` through the `RoleEnvironmentChanging` and `RoleEnvironmentChanged` events. You can then decide whether to recycle (restart) the role or simply update a configuration value. The latter is what you want for tracing switches. Restarting the role may make intermittent problems disappear. The sample code for this article shows how to do this by adding a couple of values to `ServiceConfiguration.cscfg` (note that you have to also edit `ServiceDefinition.csdef`, which provides the schema) and adding some code to your roles.

## Testing on Development Fabric

What about testing on development fabric, where you don't have the Windows Azure portal to edit the configuration as you do for cloud-deployed services? First, determine the deployment ID Windows Azure has assigned to your running development fabric service. You can see this by showing the development fabric UI from the system tray while the service is running. This will be a number like 177.

1. Go to the directory where your service binaries have been put by build—typically `\bin\debug` or `\bin\release` under your service code. You'll find the copy of `ServiceConfiguration.cscfg` that was created when you built the app.
2. Next, using a text editor, edit this file to use the tracing switch you want. For instance, in the sample code, change `WebTrace` from `Off` to `Verbose`.
3. Next, in a Windows Azure SDK command prompt (Start | All Programs | Windows Azure SDK v1.1 | Windows Azure SDK Command Prompt), run this command:

```
csrun /update:NNN;ServiceConfiguration.cscfg
```

NNN is the Windows Azure deployment ID you found earlier. This will do in development fabric what clicking the `Configure` button on the Windows Azure development portal does for cloud-deployed services—update the configuration settings and trigger the events.

## Other Diagnostic Information

While this article has focused on tabular data that your application logs using `System.Diagnostics`, Windows Azure can also capture IIS logs and Failed Request Tracing (formerly known as Failed Request Buffering, or FREB) logs, among others. Some of these are placed into Windows Azure blob storage, some in Windows Azure table storage. **Figure 6** lists the available logs and where they're stored. Note that for those not enabled by default, you typically have to make a change in `web.config` or `app.config` for Windows Azure to collect these logs. Let's drill into one example not collected by default to show how this works.

## Windows Azure also will capture IIS logs and FREB logs.

As an example, let's look at how to enable FREB logging from IIS on your Web Role. To see this in action, download the sample code for the `MSDNSampleLoggingService` provided with this article. Open `web.config` for the `LoggingWebRole` and find the section labeled `<system.webServer>`. Notice that the lines shown in **Figure 7** have been added to the default Windows Azure `web.config`. This results in failure logging for any requests that take longer than 15 seconds or with status codes between 400 and 599 (the `failureDefinitions` element).

If you open `about.aspx.cs` in the `LoggingWebRole` project, you'll note that in the `PageLoad` method I've added an arbitrary delay of 18 seconds with this line of code:

```
System.Threading.Thread.Sleep(18000);
```

This will force the load of this page to be considered a failed request under the definition specified earlier.

To see the FREB log, rebuild and deploy the app into the development fabric and then find the development fabric controller



```
PS C:\Users\Mike> get-diagnosticconfiguration -deploymentID fc9a95546b8e4f3baa390879a269f55d -Verbose -StorageAccountKey
c10DUPUhFLX9JPtDLNFW7yzAwUomMTrkweqn65UzAhZJsN1LzGtIpbInfFK7tTreUtaHwonLnNuZygFngv3Uu== -StorageAccountName myferrylog
=
cmdlet Get-DiagnosticConfiguration at command pipeline position 1
Supply values for the following parameters:
(Type '?' for Help.)
BufferName: Logs
RoleName: LoggingWorkerRole
InstanceId: LoggingWorkerRole_IN_0

ScheduledTransferLogLevelFilter ScheduledTransferPeriod BufferQuotaInMB
Verbose 00:01:00 0

PS C:\Users\Mike>
```

Figure 10 Diagnostics Configuration for a Running Service Using Windows PowerShell

in the notification area of the taskbar (you may have to click the Show Hidden Icons button on the taskbar because it's often hidden as inactive). Right-click it and select Show Development Fabric UI. While your application is running, this will show information about the application.

Expand the Web Role and right-click on the role instance (0). Select Open local store to open the folder on the local machine where logs are being saved (see Figure 8). Within that folder, the logs are in the \directory\DiagnosticStore folder. This is because the Web Role in the sample code is configured to store diagnostics information in development storage. If instead you set the DiagnosticsConnectionString to a cloud-storage account, the persisted logs will be in the blob storage associated with that storage account. You can use Cloud Storage Studio to look at the blob storage containers to see the logs.

### Managing Diagnostics for a Running Service

While you may deeply instrument your code with logging, you typically don't want all the logging information persisted into storage while your production service is running. You might want only error and critical information to go to the persisted logs, while more detailed information (logged as Verbose or Information) is suppressed.

But what if a problem occurs? You don't want to redeploy a new version of your service or the problem might magically go away—you probably know how effective rebooting can be at making elusive problems disappear.

Instead, it's more effective to increase the amount of information going to the logging tables or blob storage while allowing the misbehaving code to continue running. This is more likely to

reveal the cause of the problem in your application while it's currently operating.

Earlier I described how to fine-tune the details of what logging is passed through to Windows Azure diagnostics for a particular TraceSource. That's a sort of upstream editing of what information gets logged. In this section, I'll show you the general Windows Azure diagnostics settings that determine how information that passes through a TraceSource gets into persisted storage.

Windows PowerShell cmdlets can manage many aspects of your running Windows Azure services, including diagnostics.

Windows PowerShell cmdlets can manage many aspects of your running Windows Azure services, including diagnostics. You run these from your local machine and they connect over the Internet to the Windows Azure cloud servers running your service, providing information and adjusting parameters. Windows PowerShell is installed with Windows 7 and can be downloaded for Windows Vista from [microsoft.com/powershell](http://microsoft.com/powershell). Download the Windows Azure Service Management CmdLets from [code.msdn.microsoft.com/azurecmdlets](http://code.msdn.microsoft.com/azurecmdlets) and follow the directions for installing them. The Windows Azure diagnostics-related commands are shown in Figure 9.

```
Windows PowerShell
PS C:\Users\Mike> Set-WindowAzureLog -LogLevelFilter Error -transferperiod 2 -deploymentID fc9a95546b8e4f3baa390879a269f55d -role LoggingWebRole -StorageAccountName myferrylogs -StorageAccountKey c10DUPUhFLX9JPtDLNFW7yzAwUomMTrkweqn65UzAhZJsN1LzGtIpbInfFK7tTreUtaHwonLnNuZygFngv3Uu==
PS C:\Users\Mike> Get-DiagnosticConfiguration -deploymentID fc9a95546b8e4f3baa390879a269f55d -role LoggingWebRole -instanceID LoggingWebRole_IN_0 -bufferName Logs -StorageAccountName myferrylogs -StorageAccountKey c10DUPUhFLX9JPtDLNFW7yzAwUomMTrkweqn65UzAhZJsN1LzGtIpbInfFK7tTreUtaHwonLnNuZygFngv3Uu==

ScheduledTransferLogLevelFilter ScheduledTransferPeriod BufferQuotaInMB
Error 00:02:00 0

PS C:\Users\Mike>
```

Figure 11 Changing Diagnostics Configuration from Windows PowerShell



# I KNOW A PLACE THAT'S DIFFERENT, BUT FAMILIAR

Code in the cloud ... with the development technologies you already know

With **Windows Azure™**, you have a flexible, on-demand infrastructure that taps into all the IT resources you need. The advantages?

**You have the freedom to focus.**

With the cloud as your development platform, you can focus on what you do best: create applications, without the restrictions on performance, capacity, or deployment that come with using your own hardware rather than the cloud.

***Find out more.***

See how Windows Azure can help get you coding in the cloud. Plus, check out some offers exclusive to MSDN® subscribers. Go to **[www.windowsazure.com/msdn](http://www.windowsazure.com/msdn)** now.

**You can code in the language you choose.**

Windows Azure enables you to build applications in multiple languages, including .NET, PHP, and Java.

**You're already equipped for the cloud.**

You can start developing applications for Windows Azure right away, using familiar tools such as Microsoft® Visual Studio® 2008.





```

PS C:\Users\Mike> Get-ActiveTransfers -deploymentID fc9a95546b8e4f3baa390879a269f55d -role LoggingWebRole -instanceID LoggingWebRole_IN_0 -StorageAccountName nyferrylogs -StorageAccountKey c10dWPUhfLX9JPtDLNFW7yzAuUonHTrkueqn65WshZJsN1zGtIpbInfFK7tTreUtwAHuonLnNuZygFngv3Uw==

PS C:\Users\Mike> Start-OnDemandTransfer -deploymentID fc9a95546b8e4f3baa390879a269f55d -role LoggingWebRole -instanceID LoggingWebRole_IN_0 -dataBufferName Directories -from 2010-04-26T12:40:00-0800 -to 2010-04-26T13:00:00-0800 -NotificationQueueName LogXfer -LogLevelFilter Verbose -StorageAccountName nyferrylogs -StorageAccountKey c10dWPUhfLX9JPtDLNFW7yzAuUonHTrkueqn65WshZJsN1zGtIpbInfFK7tTreUtwAHuonLnNuZygFngv3Uw==

Guid
-----
1d9b8b88-da08-4c25-92db-73fc89cbe96c

PS C:\Users\Mike> Get-ActiveTransfers -deploymentID fc9a95546b8e4f3baa390879a269f55d -role LoggingWebRole -instanceID LoggingWebRole_IN_0 -StorageAccountName nyferrylogs -StorageAccountKey c10dWPUhfLX9JPtDLNFW7yzAuUonHTrkueqn65WshZJsN1zGtIpbInfFK7tTreUtwAHuonLnNuZygFngv3Uw==

CurrentStatus      : Published (OK to end/cancel)
DataBufferName     : Directories
DeploymentId        : fc9a95546b8e4f3baa390879a269f55d
NotificationQueueName : logxfer
RequestId          : 1d9b8b88-da08-4c25-92db-73fc89cbe96c
InstanceId         : LoggingWebRole_IN_0
RoleName           : LoggingWebRole

```

Figure 12 Using Windows PowerShell to Initiate a Transfer of IIS Logs

For example, to find the current transfer parameters for a particular role instance, pass the deployment ID (from the Windows Azure developer portal where you deploy the service) and the storage account name and key you used for `DiagnosticsConnectionString` in the `app.config` or `web.config` for the service role (see [Figure 10](#)). Notice that Windows PowerShell prompts for a couple of missing required parameters—the role instance and the name of the buffer I want. Logs is the standard Windows Azure logs. The result shows that the filter level is `Verbose` and a transfer is scheduled every minute.

If I didn't pass a notification queue, the transfer seemed not to happen.

To change the configuration for this role, use the `Set-Diagnostic-Configuration` cmdlet, as shown in [Figure 11](#). Note that I changed the transfer period from one minute to two minutes and the filter from `Verbose` to `Error`, meaning that only events logged as `Error` and `Critical` will be sent to the persisted storage.

Perhaps the most useful thing you can do remotely from Windows PowerShell is to force a transfer of log information immediately. [Figure 12](#) shows how to do this.

First, I query for any existing on-demand transfer of logs. There is a restriction in the current Windows Azure diagnostics implementation that only one on-demand transfer of a particular type can take place at a time. Seeing that none is in progress, I request one, passing in the deployment ID of the service, the role and instance, the type of log I want transferred and the time interval of data to transfer. (For the log type, `Directories` means file-based logs, including the IIS logs. Logs would be the Windows Azure table-based logs sent through `TraceSource`.)

I also pass a notification queue name, which is where Windows Azure diagnostics will send a notification that the transfer has

completed. Through experimentation, I found that if I didn't pass a notification queue, the transfer seemed not to happen. I get back a GUID identifying the transfer request. I then query for the request status and see that it's published, meaning it's in progress.

The current version of the Windows Azure Service Management CmdLets doesn't seem to show when the request has been completed, but if you query the blob storage for your diagnostics storage you'll see the logs pop up in the Containers hierarchy shortly (or in Windows Azure table storage if you requested transfer of information stored in table storage).

### Wrapping Up

Using a combination of adjusting configuration parameters for the `TraceSources` you define in your code and using the Windows Azure Service Management CmdLets to move information into persisted storage, you should be able to fully control what diagnostic output you get from your Windows Azure services.

Of course, the most important thing you can do to troubleshoot your production code is to develop a robust strategy for diagnostic output early in development, then follow that strategy throughout coding. The use of `TraceSources` and the tools that Windows Azure provides for configuring the verbosity of diagnostic output that flows from your application to storage will help you to tap into that to get just the amount of information you need when a problem arises.

There's nothing worse than feeling like the code behaving erratically on a server is a black box, opaque to you. Solid diagnostics code and the tools described here will let you open the covers of that box and peer inside. ■

**MIKE KELLY** is a consultant focused on software development and helping integrate acquisitions to larger corporations. He previously worked for 15 years at Microsoft in a variety of product development roles and as director of Emerging Practices on the Engineering Excellence team. He can be reached at [himself@mikekellyconsulting.com](mailto:himself@mikekellyconsulting.com).

**THANKS** to the following technical experts for reviewing this article: Sumit Mehrotra, Michael Levin and Matthew Kerner from Microsoft, as well as Neil Mackenzie and Steven Nagy

The leading set of Charting, Diagramming, Gauge and Map Components  
for .NET-centric technologies.



Choose the right set of components for your  
Presentation, Scientific, Financial and Business Intelligence applications.

Our products will deliver rich data visualization functionality,  
with exceptional features to your projects - on budget and on time.



# Encoding Videos Using Microsoft Expression Encoder 3 SDK

Adam Miller

In one of my favorite movie scenes of all time, Clark W. Griswold (Chevy Chase in “Christmas Vacation”) gets trapped in his attic while hiding Christmas presents. To keep warm, he dons pink gloves, a green hat and a brown fur stole pulled from a dusty chest. At the bottom of the chest he finds home movies from his youth, and passes the time watching them (with tears in his eyes), using an old film projector.

Home movies have come a long way since then, but people still have to deal with one of the same issues: How do I show my movie to friends and family? Sites like YouTube, Vimeo and Facebook make sharing easy; but at 100-plus megabytes per minute for high-definition video, getting the data to those sites can be a time-consuming task. Chances are, your portable device, gaming system or home theater media center won't even play the file. To solve these problems, you need to convert the video to another format. This process is known as encoding.

## This article discusses:

- How video encoding works
- Encoding for various types of videos
- Adding Silverlight Smooth Streaming
- Using the Microsoft Expression Encoder 3 SDK

## Technologies discussed:

C#, Microsoft Expression Encoder 3, Silverlight, Silverlight Smooth Streaming

## About Expression Encoder

The Microsoft video encoding tool, Expression Encoder 3, is part of the Expression family of products for creating compelling UIs for Web and desktop applications. Expression Encoder comes in free and paid versions; the paid version is part of both Expression Studio 3 Suite (\$599) and Expression Web 3 Suite (\$149). The free download does not support encoding to Silverlight Smooth Streaming or H.264 video or using H.264 video as a source, but it does let you encode to Windows Media Video files and it has a nice SDK. Many of the code samples in this article require the paid version of the program; however, all the code samples will build in the free version of the SDK. You'll just receive an `InvalidMediaException` or a `FeatureNotAvailableException` when running.

If you aren't ready to purchase Expression Suite, you can get started with Expression Encoder by downloading the free version from [microsoft.com/expression](http://microsoft.com/expression). It's also available as part of Expression Professional MSDN Subscription, or Visual Studio Professional with MSDN Premium Subscription. Keep in mind that \$149 for a professional video encoding software application with this feature set, wide range of input formats and supported output targets is a relative steal. Similar video encoding solutions can cost upward of \$3,000.

No matter which version you choose, you'll want to install the Encoder 3 QFE. It adds support for additional file types and input devices, improves performance in certain situations, and includes minor bug fixes. The QFE installer can be found on the Expression Encoder page on the Microsoft Expression Web site.

## Supported Formats

The following are supported input video formats:

- Windows Media Video (.wmv)
- DVD video (.vob)
- MPEG (.mpg, .mpeg)
- Audio Video Interleave (.avi)
- Microsoft Digital Video Recording (.dvr-ms)

The paid version adds the following formats (plus a handful of other formats):

- MPEG-4 (.mp4, .m4v)
- Quicktime (.mov)
- AVC HD (.mts)
- Mobile Device Video (.3gp, .3g2)

For the most part, Expression Encoder supports any media file Windows Media Player can play. If you want to support even more files (and be able to play them in Windows Media Player), you can install a codec pack such as K-Lite Codec Pack ([codecguide.com](http://codecguide.com)) or Community Combined Codec Pack ([cccp-project.net](http://cccp-project.net)). Both are based on the open source ffdshow project and will add support for VP6-encoded Flash (.flv) files, H.264 video in the Matroska (.mkv) container, and Ogg (.ogg) video files.

The free version of Expression Encoder supports only the Microsoft VC-1 as an output codec. However, this still allows you to encode videos for Silverlight (single bitrate only), Xbox 360, Zune and Zune HD. Also, the VC-1 codec is no slouch; its compression is as good (if not better in certain situations) as H.264. Upgrading to the paid version lets you output Silverlight Smooth Streaming video (multi-bitrate) as well as H.264, which means you can encode videos playable on the iPhone, PS3, Flash player (version 10 supports H.264/.mp4) and countless other devices.

## Encoding 101

Supporting certain output devices requires changing some of the video profile settings, so you'll need to understand the basics of video encoding. Re-encoding video is actually the process of decompressing a video and re-compressing it using another codec or manually changing attributes such as size, aspect ratio or frame rate. Although there are lossless compression methods, they're rarely used because the resulting video files are still quite large. So in order to reduce the amount of space necessary to store (and therefore transfer) the video, an algorithm, known as a codec, is used to compress and decompress the video. The compressed video stream is then stored according to a specification known as a container (such as WMV or MP4). Containers and codecs are often not an exclusive contract, so although H.264 is the most common codec found in the MP4 container, other codecs could be used.

Bitrate, expressed in kilobits per second, defines how much data should be used to store the compressed video. Reducing the bitrate tells the encoder to compress the video at a higher rate, degrading video quality. There are different ways to tell the encoder how to determine the video's bitrate. The simplest way is to use a constant bitrate (CBR), which forces the encoder to use the same amount of data for every second of video. A variable bitrate (VBR) can be used to tell the encoder what the overall bitrate of the file should be, but the encoder is allowed to raise or lower the bitrate based

Figure 1 Creating a Video for Zune HD

```
using Microsoft.Expression.Encoder;

namespace TestApp
{
    class Program
    {
        static void Main(string[] args)
        {
            MediaItem src = new MediaItem(@"C:\WMdownloads\AdrenalineRush.wmv");
            Job job = new Job();
            job.MediaItems.Add(src);
            job.ApplyPreset(Presets.VC1ZuneHD);
            job.OutputDirectory = @"C:\EncodedFiles";
            job.Encode();
        }
    }
}
```

on the amount of data needed for a particular section of the video. Variable constrained bitrate is similar to unconstrained VBR, except that you give not only an average bitrate to use, but also a maximum bitrate that can't be exceeded.

Variable constrained bitrate is useful when encoding Silverlight Smooth Streaming video. It helps ensure the bitrate doesn't exceed the client bandwidth, forcing the client to request a lower-quality stream. CBR and VBR indicate the amount of compression to use by specifying an overall video file size.

Sites like YouTube, Vimeo  
and Facebook make the act  
of sharing easy; but at  
100-plus megabytes per minute  
of high-definition video, getting  
the data to those sites can be a  
time-consuming task.

Alternatively, you can tell the encoder to use a quality-based VBR. Instead of specifying the overall size of the video, you specify a percentage of quality (that is, how much data) of the decompressed source video to retain. It takes less data to retain good quality for a cartoon, for example, than for a nature or action-filled video. So if you have a high-quality source and your goal is to convert the source to another format and retain optimal quality, consider using quality-based VBR. These definitions are just the tip of the iceberg, but they are core to choosing your output settings. You'll find additional encoding definitions throughout this article as they apply to code samples.

## Using the SDK

To follow the code samples, you'll want to use a good-quality video. If you don't have any high-resolution video lying around, you can get some nice HD videos from [microsoft.com/windows/](http://microsoft.com/windows/)



Figure 2 Video and Audio Profile Settings for Zune HD

```
MediaItem src = new MediaItem(@"C:\WMdownloads\AdrenalineRush.wmv");
src.OutputFormat = new WindowsMediaOutputFormat();

src.OutputFormat.VideoProfile = new AdvancedVC1VideoProfile();
src.OutputFormat.VideoProfile.Bitrate = new
VariableConstrainedBitrate(1000, 1500);
src.OutputFormat.VideoProfile.Size = new Size(480, 272);
src.OutputFormat.VideoProfile.FrameRate = 30;
src.OutputFormat.VideoProfile.KeyFrameDistance = new TimeSpan(0, 0, 4);

src.OutputFormat.AudioProfile = new WmaAudioProfile();
src.OutputFormat.AudioProfile.Bitrate = new
VariableConstrainedBitrate(128, 192);
src.OutputFormat.AudioProfile.Codec = AudioCodec.WmaProfessional;
src.OutputFormat.AudioProfile.BitsPerSample = 24;

Job job = new Job();
job.MediaItems.Add(src);
job.OutputDirectory = @"C:\EncodedFiles";
job.Encode();
```

windowsmedia/musicandvideo/hdvideo/contentshowcase.aspx. I'll use the Adrenaline Rush video as the source for these examples.

After installing Expression Encoder 3, create a new Visual Studio C# Console Application project. Add references to Microsoft.Expression.Encoder.dll and Microsoft.Expression.Encoder.Utilities.dll, located at \Program Files (x86)\Microsoft Expression\Encoder 3\SDK. You'll also need to add a reference to WindowsBase, which you'll find in the .NET tab of the Add References dialog. Many of the classes used will be in the Microsoft.Expression.Encoder namespace, so add a using statement for it.

The first item to instantiate will be a MediaItem object. The MediaItem constructor takes a string as the only parameter to the constructor. Pass the path to the file you're using as the source for the encoding project:

```
MediaItem src = new MediaItem(@"C:\WMdownloads\AdrenalineRush.wmv");
```

Creating a MediaItem object takes just a second or two. The SDK is doing a fair amount of work behind the scenes, though, gathering information about the source video, such as its height, width, frame rate (the frequency that individual images should be displayed on the screen) and duration. Information about the audio stream is also gathered at this time.

Next you create an instance of the Job class (which has only a parameterless constructor), and add your MediaItem to its list of MediaItems. The Job class serves as the manager for desired output formats (known as profiles):

```
Job job = new Job();
job.MediaItems.Add(src);
```

Now you need to tell the job which audio and video profiles to use during encoding; the easiest way is to use one of the profiles defined in the UI. To create a video for the Zune HD, for example, you can use the VC1ZuneHD preset:

```
job.ApplyPreset(Presets.VC1ZuneHD);
```

Finally, specify an output directory and start the encoding process:

```
job.OutputDirectory = @"C:\EncodedFiles";
job.Encode();
```

Your Program.cs file should be similar to **Figure 1**.

There's one last thing to do before running the application: If you're using a 64-bit version of Windows, you'll need to modify the project to build to x86. In the Visual Studio menu bar, select Project and (Project Name) Properties. In the dialog box that

opens, select the build tab and change the Platform Target from "Any CPU" to "x86."

You are now ready to run the application and create a video playable on the Zune HD. The encoding process will take a couple minutes to complete and is extremely CPU-intensive. Video encoding benefits from being a parallel computed task, so multi-core computers have a big advantage here.

Expression Encoder also includes presets for encoding to online services such as YouTube, Vimeo and Facebook. 720p video recorded from my Panasonic Lumix DMC-ZS3 digital camera consumes about 110MB per minute of recorded video. Converting the video using the YouTube HD preset (also 720p) reduces the video to just 16MB. This makes it much more efficient to upload and store locally. Converting it to an .mp4 file also makes it compatible with many more video editing programs.

## Custom Settings

To manually produce the same output as the VC1ZuneHD preset, you'd need to use code similar to **Figure 2** to set the video and audio profiles.

For the code in **Figure 2** to compile, you'll need to add references to Microsoft.Expression.Encoder.Utilities and System.Drawing. Also add using statements for Microsoft.Expression.Encoder.Profiles and System.Drawing. The OutputFormat essentially specifies the container for the output file. I say essentially because encoding for Silverlight works just a little bit differently (as I'll discuss shortly).

The VideoProfile specifies the video codec to use, along with the detailed settings to use when encoding. Similarly, the AudioProfile specifies the audio codec to use along with its settings. When constructing a VariableConstrainedBitrate, the first parameter specifies the average bitrate and the second parameter specifies the maximum bitrate. The size setting indicates the box the encoded video should fit in. The correctly scaled size for the Adrenaline Rush video is actually 480x272 to maintain the aspect ratio, but if I entered 480x480 the resulting video still would be 480x272.

**Figure 2**'s KeyFrameDistance property refers to a video-encoding concept I haven't yet discussed. The way the most video encoding works

Figure 3 Adding Silverlight Smooth Streaming

```
MediaItem src = new MediaItem(@"C:\WMdownloads\AdrenalineRush.wmv");
src.OutputFormat = new WindowsMediaOutputFormat();

src.OutputFormat.VideoProfile = new AdvancedVC1VideoProfile();
src.OutputFormat.VideoProfile.KeyFrameDistance = new TimeSpan(0, 0, 2);
src.OutputFormat.VideoProfile.SmoothStreaming = true;

src.OutputFormat.VideoProfile.Streams.Clear();
src.OutputFormat.VideoProfile.Streams.Add(new StreamInfo(new
VariableConstrainedBitrate(2000, 3000), new Size(1280, 720)));
src.OutputFormat.VideoProfile.Streams.Add(new StreamInfo(new
VariableConstrainedBitrate(1400, 1834), new Size(848, 476)));
src.OutputFormat.VideoProfile.Streams.Add(new StreamInfo(new
VariableConstrainedBitrate(660, 733), new Size(640, 360)));

src.OutputFormat.AudioProfile = new WmaAudioProfile();
src.OutputFormat.AudioProfile.Bitrate = new
VariableConstrainedBitrate(128, 192);
src.OutputFormat.AudioProfile.Codec = AudioCodec.WmaProfessional;
src.OutputFormat.AudioProfile.BitsPerSample = 24;

Job job = new Job();
job.MediaItems.Add(src);
job.OutputDirectory = @"C:\EncodedFiles";
job.Encode();
```

# GET THE FASTEST CONTROLS...



At Infragistics, we make sure our **NetAdvantage for .NET** controls make every part of your User Interface the very best it can be. That's why we've tested and re-tested to make sure our **Data Grids are the very fastest** grids on the market and our **Data Charts outperform** any you've ever experienced. Use our controls and not only will you get the fastest load times, but your apps will always look good too. Fast and good-looking...that's a killer app. Try them for yourself at [infragistics.com/wow](http://infragistics.com/wow).

**Infragistics**  
KILLER APPS. NO EXCUSES.

Infragistics Sales 800 231 8588  
Infragistics Europe Sales +44 (0) 800 298 9055  
Infragistics India +91-80-6785-1111  
[twitter.com/infragistics](http://twitter.com/infragistics)



is to store only the changes from one frame to the next, rather than the entire picture for each video frame. Key frames are the frames that contain the entire image. This code will create key frames every four seconds. Key frames will be created automatically when there are large changes in the video such as a scene change, but you should also create them at pre-defined intervals to support seeking within the movie during playback.

## Silverlight Smooth Streaming

Silverlight Smooth Streaming dynamically switches the bitrate of the media file being played based on current network conditions. A Smooth Streaming project consists of individual videos stored in .ismv files, as well as .ism and .ismc metadata files that support Smooth Streaming playback.

Silverlight Smooth Streaming  
dynamically switches the  
bitrate of the media file being  
played based on current  
network conditions.

To create a Silverlight Smooth Streaming project, multiple changes must be made. First, change the `KeyFrameDistance` to two seconds. The video will still play if the `KeyFrameDistance` is left at four seconds, but you may notice hiccups in playback when the player switches bitrates. The Silverlight player will request the video in two-second chunks, so playback is more consistent if there's a key frame at the beginning of each request. You also need to add the following line:

```
src.OutputFormat.VideoProfile.SmoothStreaming = true;
```

Setting `SmoothStreaming` to `true` tells the encoder to output the videos to .ismv files and create the .ism and .ismc files. Having only one bitrate isn't really a smooth streaming project, so to create multiple output bitrates, you need to add multiple streams to the `VideoProfile`. Do this using code similar to **Figure 3**.

Here the code specifies three different bitrates and sizes to encode. For optimum quality, the video size needs to shrink as the bitrate is reduced. When specifying your own bitrates, you can use the IIS Smooth Streaming settings in the Expression Encoder 3 UI. Note that it's not possible to gain quality by encoding a video at a higher resolution than the source file. And it only makes sense to encode at a higher bitrate than the source file if using a weaker compression method. If the SDK was able to determine the bitrate of the source file, it will be present in the `MediaItem's SourceVideo Profile` property:

```
int bitrate = ((ConstantBitrate)src.SourceVideoProfile.Bitrate).Bitrate;
```

If the SDK couldn't obtain the bitrate of the source file, you can get a pretty close estimate based on the file size. Here's the formula:

Approximate bitrate in kb/s = (file size in kilobytes \* 8 / video duration in seconds) - audio bitrate in kb/s

You can use the `System.IO.FileInfo` class to get the source-file size, and the SDK to get the duration (`MediaItem.FileDuration`

property) and possibly the audio bitrate. If you don't know the audio bitrate, use 128 or 160 to estimate (most audio bitrates are between 64 and 192); you may also be able to get the audio bitrate in the Windows Media Player Properties window (Press Alt to show the menu, then File | Properties).

## Monitoring Progress

Because an encoding job can take hours to complete, it's helpful to be able to see the encoding progress. The SDK provides a simple way to monitor the encoding process via an event you can add a handler for:

```
job.EncodeProgress += new EventHandler<EncodeProgressEventArgs>(OnProgress);
```

Add a method like the following to handle the event:

```
static void OnProgress(object sender, EncodeProgressEventArgs e)
{
    Console.Clear();
    Console.WriteLine((100 * (e.CurrentPass - 1) + e.Progress) /
        e.TotalPasses + "%");
}
```

Multi-pass encoding is a new concept relevant to this code sample. When using a variable bitrate to encode, the process is done in two steps, known as *passes*. During the first pass, the source video is analyzed to determine which parts are most complex and would benefit from an increased bitrate. During the second pass, the video is encoded using the information obtained during the first pass. Thus, if you use a constant bitrate, there's no need to use the `CurrentPass` or `TotalPasses` properties of the `EncodeProgressEventArgs` class.

## Combining Videos

If you want to encode only part of a video or combine multiple videos into one, the SDK provides support. To modify the start and stop time for a source media item, you can modify the `Clips` property. To encode only the first six seconds of a video, use code similar to:

```
src.Sources[0].Clips[0].StartTime = new TimeSpan(0);
src.Sources[0].Clips[0].EndTime = new TimeSpan(0, 0, 6);
```

To add other videos as source files, you can append additional videos to the `Sources` property of your `MediaItem`. This will encode the source files in order to a single output file:

```
MediaItem src = new MediaItem(@"C:\WMdownloads\AdrenalineRush.wmv");
src.Sources.Add(new Source(@"C:\WMdownloads\Video2.wmv"));
```

## Live Encoding

Expression Encoder also supports encoding from live sources such as a webcam. The concept (and code) is similar to encoding video files, but you use a different set of classes. These are found in the `Microsoft.Expression.Encoder.Live` namespace.

### Figure 4 Encoding Live Video

```
using (LiveJob job = new LiveJob())
{
    LiveDevice videoDevice = job.VideoDevices[0];
    LiveDevice audioDevice = job.AudioDevices[0];
    LiveDeviceSource liveSource = job.AddDeviceSource(videoDevice, audioDevice);
    job.ActivateSource(liveSource);
    WindowsMediaBroadcastOutputFormat outputFormat =
        new WindowsMediaBroadcastOutputFormat();
    outputFormat.BroadcastPort = 8080;
    job.OutputFormat = outputFormat;
    Console.WriteLine("Press enter to stop encoding...");
    job.StartEncoding();
    Console.ReadLine();
    Console.WriteLine("Stopping");
    job.StopEncoding();
}
```

WINDOWS FORMS / WPF / ASP.NET / ACTIVEX

# WORD PROCESSING COMPONENTS

( WHAT YOU SEE IS WHAT YOU GET )

Visit our booth #2444 at:

Microsoft®  
tech·ed  
North America | 2010

June 7-10, 2010 • New Orleans, LA

**TX**  
**TEXT CONTROL**®  
word processing components

Word Processing Components  
for Windows Forms & ASP.NET

[www.textcontrol.com](http://www.textcontrol.com)

Microsoft®  
**Visual Studio**  
PARTNER

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



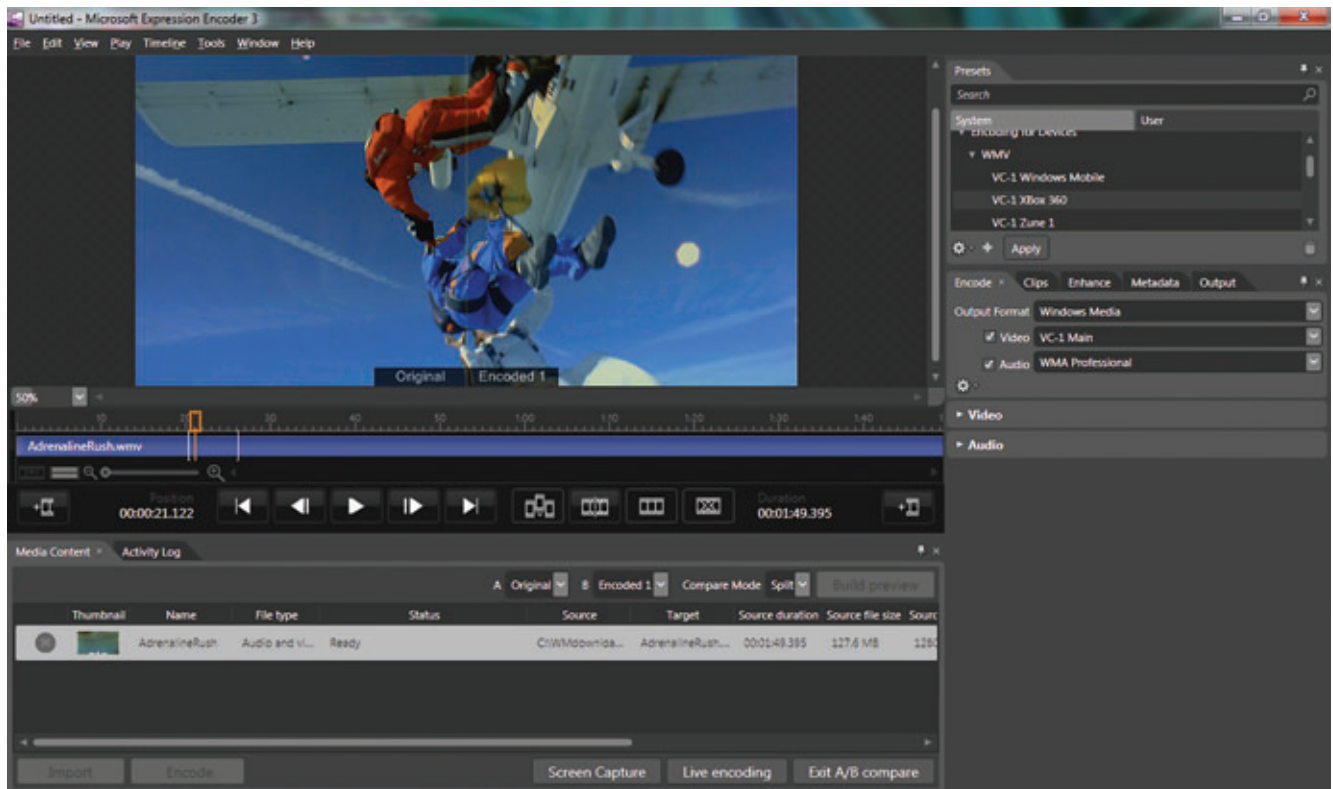


Figure 5 A/B Compare in Expression Encoder 3

The first class to use is `LiveJob`. `LiveJob` works like `Encoder.Job`—it handles the work of encoding the video. However, in a live scenario the `OutputFormat` is a property of `LiveJob` instead of a `MediaItem` object (which is not necessary). When a `LiveJob` object is instantiated, it will look for video input devices attached to your computer and populate `VideoDevices` and `AudioDevices` properties. You can then use these as an input source for the encoder. **Figure 4** shows an example.

LiveJob works like Encoder.Job—  
it handles the work of encoding  
the video.

This will start a live encoding session using a webcam (assuming you have one connected) and broadcast it on your local machine on port 8080. To view the live encoding, open Windows Media Player and select `File | Open URL` and enter `mms://localhost:8080`. After some buffering, you should see the video from your webcam, though you'll notice a 20- to 30-second lag due to the time it takes to encode and transport the stream. You could potentially use this video as a source for Windows Media Services or IIS Media Services to broadcast to the world.

## Additional Tools

If you aren't sure whether the encoding settings you've chosen will give you the output quality you need, the Expression Encoder 3 UI provides a handy feature called A/B Compare. This lets you encode

five seconds of video surrounding the current playback position. The encoded video will appear split-screen with your source video (see **Figure 5**), so you can easily compare the quality of the encoded video with the original.

You can then save the current settings as a user-defined preset by clicking `Edit | Save current settings as preset`. The preset will be stored as an XML file, which you can use with the SDK:

```
job.ApplyPreset(@"C:\WMdownlaods\NewPreset.xml");
```

If you're already thinking about how easy it would be to automate the video conversion process with a console application, take a look at the `Convert-Media PowerShell Module for Expression Encoder`, available at [convertmedia.codeplex.com](http://convertmedia.codeplex.com). This PowerShell module wraps the Expression Encoder SDK, providing a command-line encoding interface without writing any code. As with all CodePlex projects, it is open source.

Hopefully you now understand the core terminology related to video encoding and can make educated decisions on which codec and bitrate to use. You also know how to use the Expression Encoder 3 SDK to encode videos for specific targets such as Xbox 360, iPhone and Silverlight, as well as live streaming video. So don't wait to be trapped in your attic like Clark W. Griswold to realize the value of your home videos and forgotten memories. Convert them to a format that will make them accessible to the world. ■

**ADAM MILLER** is a software engineer for Nebraska Global in Lincoln, Neb. You can follow Miller's blog at [blog.milrr.com](http://blog.milrr.com).

**THANKS** to the following technical expert for reviewing this article:  
*Ben Rush*

# INSPIRE! EMPOWER! IMPRESS!



You've got the data, but time, budget and staff constraints can make it hard to present that valuable information in a way that will impress. With Infragistics' **NetAdvantage for Silverlight Data Visualization**, you can create Web-based data visualizations and dashboard-driven applications on Microsoft Silverlight (and coming soon for WPF) that will not only impress decision makers, it actually empowers them. Go to [infragistics.com/sldv](http://infragistics.com/sldv) today and get inspired to create killer apps.

**Infragistics**  
KILLER APPS. NO EXCUSES.

Infragistics Sales 800 231 8588  
Infragistics Europe Sales +44 (0) 800 298 9055  
Infragistics India +91-80-6785-1111  
[twitter.com/infragistics](http://twitter.com/infragistics)



# VSLive!

**JOIN US AT MICROSOFT HQ!**

## **CALLING ALL DEVELOPERS!**

**Join Us August 2-6, in Redmond, Washington  
for Five Code-Packed Days on the  
Microsoft Campus!**

VSLive! delivers an unbiased blend of practical, pragmatic and immediately-applicable knowledge, plus a glimpse of the future of technology. Interact one-on-one with the Microsoft development team and connect with industry experts during in-depth sessions on:

- Silverlight, WPF, XAML, VS10 Xaml designer, Blend, WCF RIA Services
- Web Forms, ASP.NET MVC, AJAX
- Cloud Computing
- Azure, Amazon, AppFabric, REST services, "Dallas", WCF, Windows Workflow
- SharePoint, Office
- SQL Server, BI, reporting, analysis, ADO.NET EF, ANDS, OData, Sync services
- Visual Studio features, TFS, languages, parallel extensions

And on top of all this great education, you'll have **EXCLUSIVE** access to check out Microsoft's corporate headquarters!

**Register by June 30th and Save Big!**  
**Go to [vslive.com](http://vslive.com) to Reserve  
Your Place on Campus!**

Supported by:

**Microsoft**



**msdn**

Visual Studio  
MAGAZINE



Microsoft  
**Visual Studio**



AUGUST 2-6, 2010

**ATTEND VSLIVE! REDMOND**  
AND LEARN ONE-ON-ONE WITH THE  
MICROSOFT DEVELOPMENT TEAM!

**Microsoft**



DETAILS AND REGISTRATION AT

**VSLIVE.COM**

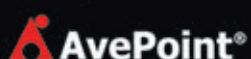
USE PRIORITY CODE MSDN7A

Platinum Sponsors



VERSANT

Gold Sponsors





# MULTIPLE TRACKS. KILLER CONTENT.

## VSLIVE! REDMOND TRACKS

Cloud Computing	Data Management	SharePoint	Silverlight/WPF	Visual Studio 2010/.NET 4	Web
<b>CONFERENCE DAY 1: TUESDAY, AUGUST 3, 2010</b>					
Getting Started in Silverlight – A Jumpstart to Productivity <i>Tim Huckaby</i>	What's New in ASP.NET 4 WebForms <i>Wallace McClure</i>	What's New in the SharePoint 2010 Development Platform <i>Ted Pattison &amp; Mike Fitzmaurice</i>	New IDE and Languages Features in VS 2010 using VB and C# <i>Ken Getz</i>	Microsoft Learning Performance Based Exam Scoring	
Transitioning from Windows Forms to WPF <i>Miguel Castro</i>	ASP.NET MVC Quick Primer <i>Gus Emery</i>	Introduction to SharePoint Development with Visual Studio 2010 <i>Mike Morton</i>	Best Kept Secrets in Visual Studio 2010 and .NET 4.0 <i>Deborah Kurata</i>	Microsoft Session To Be Announced	
WPF & Silverlight: Data Visualization, NUI, and Next Generation of User Experience <i>Tim Huckaby</i>	Advanced ASP.NET MVC <i>Gus Emery</i>	Silverlight Development with SharePoint 2010 <i>Mike Ammerlaan</i>	What's New in the .NET 4.0 BCL <i>Jason Bock</i>	Microsoft Session To Be Announced	
Intro to Windows Phone 7 Development <i>Walt Ritscher</i>	Understanding ASP.NET Under the Covers <i>Miguel Castro</i>	Feature Upgrade enhancements in SharePoint 2010 <i>Ted Pattison</i>	What's New in Visual Studio 2010 Debugging <i>Brian Peek</i>	Microsoft Session To Be Announced	
<b>CONFERENCE DAY 2: WEDNESDAY, AUGUST 4, 2010</b>					
Bind Anything to Anything in Silverlight and WPF <i>Ken Getz</i>	AJAX with the UpdatePanel, WebForms, and the AJAX Control ToolkitX <i>Wallace McClure</i>	Creating SharePoint 2010 Workflows with SharePoint Designer 2010 <i>Eilene Hao Klaka</i>	So Many Choices, So Little Time: Understanding Your .NET Data Access Options <i>Leonard Label</i>	Microsoft Session To Be Announced	
CSLA 4, Silverlight and WPF <i>Rockford Lhotka</i>	Leveraging Client Capabilities with jQuery in Visual Studio and ASP.NET <i>Robert Boedigheimer</i>	Creating Workflow Templates with Visual Studio 2010 <i>Mike Fitzmaurice</i>	Windows Server AppFabric Caching <i>Jon Flanders</i>	Microsoft Session To Be Announced	
Using Microsoft Prism – Loose Coupling for Application Development <i>David Platt</i>	jQuery for ASP.NET Developers – Part 1 <i>Jeffrey McManus</i>	Integrating SharePoint 2010 and Azure: Evolving Towards the Cloud <i>Steve Fox</i>	Building RESTful Services Using Windows Communication Foundation <i>Jon Flanders</i>	Microsoft Session To Be Announced	
Silverlight, WCF RIA Services, and Your Business Objects <i>Deborah Kurata</i>	jQuery for ASP.NET Developers – Part 2 <i>Jeffrey McManus</i>	Client-side Development for SharePoint 2010 using JavaScript and jQuery <i>Paul Stubbs</i>	Building RESTful Applications with the Open Data Protocol <i>Stephen Forte</i>	Microsoft Session To Be Announced	
Generating Dynamic UI in WPF 4 and Silverlight 4 <i>Billy Hollis</i>	ASP.NET Request Processing <i>Robert Boedigheimer</i>	PowerPivot and Excel Services <i>Andrew Brust</i>	Can You Use MEF Too Much? Hint – NO! <i>Jon Flanders</i>	Microsoft Session To Be Announced	
<b>CONFERENCE DAY 3: THURSDAY, AUGUST 5, 2010</b>					
Multi-Touch Madness! <i>Brian Peek</i>	Azure Platform Overview <i>Vishwas Lele</i>	SharePoint 2010 Support for Social Computing and Communities <i>Rob Foster</i>	Why Software Sucks <i>David Platt</i>	Microsoft Session To Be Announced	
Deep Dive into Windows Phone 7 Development <i>Walt Ritscher</i>	Architecting for Azure <i>Vishwas Lele</i>	Developing Social Computing Solutions <i>Rob Foster</i>	The Daily Scrum <i>Stephen Forte</i>	SharePoint 2010 WCM for Developers with Visual Studio 2010 <i>Kirk Evans</i>	
Design, Don't Decorate <i>Billy Hollis</i>	Developing an Azure Based Monte Carlo Simulator <i>Vishwas Lele</i>	SharePoint Developers Deep Dive into Claim-based Security <i>Ted Pattison</i>	What's New in VS 2010 for TFS? <i>Brian Randell</i>	Understanding How the Sandbox Works <i>Maurice Prather</i>	
Using WPF for Good and Not Evil <i>David Platt</i>	What is Microsoft Dallas? <i>Michael Stiefel</i>	Creating No-Code SharePoint Applications with SharePoint Designer <i>Rob Howard</i>	Model-Driven Software Development in .NET with CodeFluent Entities <i>Daniel Cohen-Zardi &amp; Simon Mourier</i>	Best Practices for Upgrading Web Parts <i>Maurice Prather</i>	
Reserved for Late-Breaking Content	How Do You Decide Between Relational Database or Table Storage in the Cloud? <i>Michael Stiefel</i>	Creating Extensions for the Visual Studio 2010 SharePoint Tools <i>Jon Flanders</i>	Reserved for Late-Breaking Content	Application Lifecycle Management for Developers in SharePoint 2010 <i>Mike Morton</i>	

Don't miss pre- and post-conference workshops—go online for details!

REGISTER TODAY AT

 **VSLIVE.COM/REDMOND**

USE PRIORITY CODE MSDN7



**AUGUST 2-6, 2010**

REDMOND, WA | MICROSOFT CAMPUS

# VSLive!

*Join us at Microsoft HQ!*

## CONFERENCE PROGRAM ANNOUNCED **70 SESSIONS—Pick Yours Now!**

ALL attendees will be granted **EXCLUSIVE** access to check out Microsoft's corporate headquarters and the Microsoft employee store!

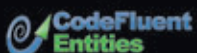
**RESERVE YOUR PLACE ON  
CAMPUS BY JUNE 30TH AT  
VSLIVE.COM/REDMOND  
AND SAVE \$200!**

**VSLIVE! PRESENTS**

**SharePoint Live**

August 3-6

Platinum Sponsors



VERSANT

Gold Sponsors



AvePoint®

Supported by:

Microsoft



Visual Studio  
MAGAZINE

Microsoft  
Visual Studio



# Enforcing Complex Business Data Rules with WPF

Brian Noyes

**Microsoft Windows Presentation** Foundation (WPF) has a rich data-binding system. In addition to being a key enabler for loose coupling of the UI definition from the supporting logic and data through the Model-View-ViewModel (MVVM) pattern, the data-binding system has powerful and flexible support for business data-validation scenarios. The data-binding mechanisms in WPF include several options for evaluating the validity of input data when you create an editable view. Plus, WPF templating and styling capabilities for controls give you the ability to easily customize the way you indicate validation errors to the user.

To support complex rules and to display validation errors to the user, you generally need to employ a combination of the available validation mechanisms. Even a seemingly simple data input form can present validation challenges when the business rules get complex.

Common scenarios involve both simple rules at an individual property level, and cross-coupled properties where the validity of one property depends on the value of another property. However, the validation support in WPF data binding makes it easy to address these challenges.

In this article, you'll see how to use the `IDataErrorInfo` interface implementation, `ValidationRules`, `BindingGroups`, exceptions, and validation-related attached properties and events to address your data-validation needs. You'll also see how to customize the display of validation errors with your own `ErrorTemplates` and `ToolTips`. For this article, I assume you are already familiar with the basic data-binding capabilities of WPF. For more background on that, see John Papa's December 2007 *MSDN Magazine* article, "Data Binding in WPF" ([msdn.microsoft.com/magazine/cc163299](http://msdn.microsoft.com/magazine/cc163299)).

## This article discusses:

- Data validation in WPF
- Exceptions and validation rules
- Cross-coupled properties
- Custom validation error display

## Technologies discussed:

Windows Presentation Foundation

## Code download available at:

[code.msdn.microsoft.com/mag201006WPF](http://code.msdn.microsoft.com/mag201006WPF)

## Data Validation Overview

Almost any time you enter or modify data in an application, you need to ensure that the data is valid before it gets too far away from the source of those changes—in this case, the user. Moreover, you need to give users a clear indication when the data they entered is invalid, and hopefully also give them some indication of how to correct it. These things are fairly easy to do with WPF as long as you know which capability to use and when.

When you use data binding in WPF to present business data, you typically use a `Binding` object to provide a data pipeline between a single property on a target control and a data source

object property. For validation to be relevant, you're typically doing TwoWay data binding—meaning that, in addition to data flowing from the source property into the target property for display, the edited data also flows from target to source as shown in **Figure 1**.

There are three mechanisms for determining whether data entered through a data-bound control is valid. These are summarized in **Figure 2**.

When a user enters or modifies data in TwoWay data binding, a workflow kicks off:

- Data is entered or modified by the user through keystrokes, mouse, touch, or pen interaction with the element, resulting in a change of a property on the element.
- Data is converted to the data-source property type, if needed.
- The source property value is set.
- The Binding.SourceUpdated attached event fires.
- Exceptions are caught by the Binding if thrown by the setter on the data-source property, and can be used to indicate a validation error.
- IDataErrorInfo properties are called on the data source object, if implemented.
- Validation error indications are presented to the user and the Validation.Error attached event fires.

As you can see, there are several points in the process where validation errors can result, depending on which mechanism you choose. Not shown in the list is where the ValidationRules fire. That's because they can fire at various points in the process, depending on the value you set for the ValidationStep property on the ValidationRule, including before type conversion, after conversion, after the property is updated or when the changed value is committed (if the data object implements IEditableObject). The default value is RawProposedValue, which happens before type conversion. The point when the data is converted from the target control property type to the data source object property type usually happens implicitly without touching any of your code, such as for a numeric input in a TextBox. This type-conversion process can throw exceptions, which should be used to indicate a validation error to the user.

If the value can't even be written to the source object property, clearly it is invalid input. If you choose to hook up ValidationRules, they are invoked at the point in the process indicated by the ValidationStep property, and they can return validation errors based on whatever logic is embedded in them or called from them. If the source object property setter throws an exception, that should almost always be treated as a validation error, as with the type conversion case.

Finally, if you implement IDataErrorInfo, the indexer property you add to your data source object for that interface will be called for the property that was being set to see if there is a validation error based on the returned string from that interface. I'll cover each of these mechanisms in more detail a bit later.

When you want validation to occur is another decision you'll have to make. Validation happens when the Binding writes the data to the underlying source object property. When validation takes place is specified by the UpdateSourceTrigger property of the Binding, which is set to PropertyChanged for most properties. Some properties, such as TextBox.Text, change the value to

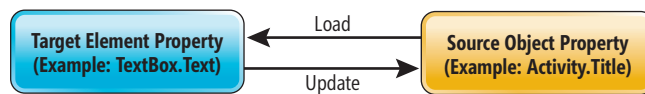


Figure 1 Data Flow in TwoWay Data Binding

FocusChange, which means that validation happens when the focus leaves the control that's being used to edit data. The value can also be set to Explicit, which means that validation has to be explicitly invoked on the binding. The BindingGroup that I discuss later in the article uses Explicit mode.

In validation scenarios, particularly with TextBoxes, you typically want to give fairly immediate feedback to the user. To support that, you should set the UpdateSourceTrigger property on the Binding to PropertyChanged:

```
Text="{Binding Path=Activity.Description, UpdateSourceTrigger=PropertyChanged}"
```

It turns out that for many real validation scenarios, you'll need to leverage more than one of these mechanisms. Each has its pros and cons, based on the kind of validation error you're concerned with and where the validation logic can reside.

## Business Validation Scenario

To make this more concrete, let's walk through an editing scenario with a semi-real business context and you'll see how each of these mechanisms can come into play. This scenario and the validation rules are based on a real application I wrote for a customer in which a fairly simple form required the use of almost every validation mechanism due to the supporting business rules for validation. For the simpler application used in this article, I'll employ each of the mechanisms to demonstrate their use, even though they're not all explicitly required.

Let's suppose you need to write an application to support field technicians who perform in-home customer support calls (think the cable guy, but one who also tries to up-sell additional features and services). For each activity the technician performs in the field, he needs to enter an activity report that tells what he did and relates it to several pieces of data. The object model is shown in **Figure 3**.

The main piece of data users fill out is an Activity object, including a Title, the ActivityDate, an ActivityType (a drop-down selection of predefined activity types) and a Description. They also need to relate their activity to one of three possibilities. They need to select either a Customer the activity was performed for from a list of customers assigned to them or an Objective of the company the activity was related to from a list of company objectives, or they can manually enter a Reason if neither a Customer nor an Objective apply for this activity.

Here are the validation rules the application needs to enforce:

- Title and Description are required fields.
- The ActivityDate must be no earlier than seven days prior to the current date and no later than seven days in the future.
- If the ActivityType *Install* is selected, the Inventory field is required and should indicate the pieces of equipment from the technician's truck that were expended. The inventory items need to be entered as a comma-separated list with an expected model number structure for the input items.
- At least one Customer, Objective or Reason must be provided.



These may seem like fairly simple requirements, but the last two in particular are not so straightforward to address because they indicate cross-coupling between properties. The running application with some invalid data—indicated by the red box—is shown in **Figure 4**.

## Exception Validation

The simplest form of validation is to have an exception that's raised in the process of setting the target property treated as a validation error. The exception could result from the type conversion process before the Binding ever sets the target property; it could result from an explicit throw of an exception in the property setter; or it could result from a call out to a business object from the setter where the exception gets thrown further down the stack.

When you want validation to occur is another decision you will have to make.

To use this mechanism, you simply set the `ValidatesOnExceptions` property to true on your Binding object:

```
Text="{Binding Path=Activity.Title, ValidatesOnExceptions=True}"
```

When an exception is thrown while trying to set the source object property (`Activity.Title` in this case), a validation error will be set on the control. The default validation error indication is a red border around the control as shown in **Figure 5**.

Because exceptions can occur in the type conversion process, it's a good idea to set this property on input Bindings whenever there's any chance of the type conversion failing, even if the backing property just sets the value on a member variable with no chance of an exception.

For example, suppose you were to use a `TextBox` as the input control for a `DateTime` property. If a user enters a string that can't be converted, `ValidatesOnExceptions` is the only way your Binding could indicate an error, because the source object property will never be called.

If you need to do something specific in the view when there is invalid data, such as disable a command, you can hook the `Validation.Error` attached event on the control. You'll also need to set the `NotifyOnValidationError` property to true on the Binding.

```
<TextBox Name="ageTextBox"
  Text="{Binding Path=Age,
    ValidatesOnExceptions=True,
    NotifyOnValidationError=True}"
  Validation.Error="OnValidationError".../>
```

## ValidationRule Validation

In some scenarios, you might want to tie the validation in at the UI level and need more complicated logic to determine whether the input is valid. For the sample application, consider the validation rule for the Inventory field. If data is entered, it needs to be a comma-separated list of model numbers that follow a specific pattern. A `ValidationRule` can easily accommodate this because it depends entirely on the value being set. The `ValidationRule` can use a `string.Split` call to turn the input into a string array, then use a regular expression to check whether the individual parts comply

with a given pattern. To do this, you can define a `ValidationRule` as shown in **Figure 6**.

Properties exposed on a `ValidationRule` can be set from the XAML at the point of use, allowing them to be a little more flexible. This validation rule ignores values that can't be converted to a string array. But when the rule can execute the `string.Split`, it then uses a `Regex` to validate that each string in the comma-separated list complies with the pattern set through the `InventoryPattern` property.

When you return a `ValidationResult` with the valid flag set to false, the error message you provide can be used in the UI to present the error to the user, as I'll show later. One downside to `ValidationRules` is that you need an expanded Binding element in the XAML to hook it up, as shown in the following code:

```
<TextBox Name="inventoryTextBox"...>
  <TextBox.Text>
    <Binding Path="Activity.Inventory"
      ValidatesOnExceptions="True"
      UpdateSourceTrigger="PropertyChanged"
      ValidatesOnDataErrors="True">
      <Binding.ValidationRules>
        <local:InventoryValidationRule
          InventoryPattern="^\d?(\d{3})\d?(\d{3})\d?(\d{4})$"/>
      </Binding.ValidationRules>
    </Binding>
  </TextBox.Text>
</TextBox>
```

In this example, my Binding will still raise validation errors if an exception occurs due to the `ValidatesOnExceptions` property being set to true, and I also support `IDataErrorInfo` validation based on the `ValidatesOnDataErrors` being set to true, which I'll talk about next.

If you have multiple `ValidationRules` attached to the same property, those rules can each have different values for the `ValidationStep` property or they can have the same value. Rules within the same `ValidationStep` are evaluated in order of declaration. Rules in earlier `ValidationSteps` obviously run before those in later `ValidationSteps`. What may not be obvious is that if a `ValidationRule` returns an error, none of the subsequent rules are evaluated. So the

Figure 2 Binding Validation Mechanisms

Validation Mechanism	Description
Exceptions	By setting the <code>ValidatesOnExceptions</code> property on a Binding object, if an exception is raised in the process of trying to set the modified value on the source object property, a validation error will be set for that Binding.
ValidationRules	The Binding class has a property to supply a collection of <code>ValidationRule</code> -derived class instances. These <code>ValidationRules</code> need to override a <code>Validate</code> method that will be called by the Binding whenever the data in the bound control changes. If the <code>Validate</code> method returns an invalid <code>ValidationResult</code> object, a validation error is set for that Binding.
IDataErrorInfo	By implementing the <code>IDataErrorInfo</code> interface on a bound data-source object and setting the <code>ValidatesOnDataErrors</code> property on a Binding object, the Binding will make calls to the <code>IDataErrorInfo</code> API exposed from the bound data-source object. If non-null or non-empty strings are returned from those property calls, a validation error is set for that Binding.

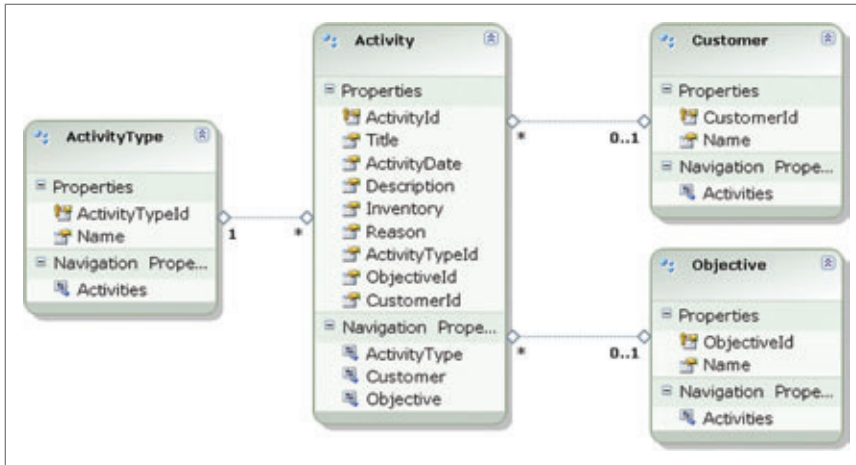


Figure 3 Object Model for the Sample Application

first validation error will be the only one indicated when the errors result from ValidationRules.

## IDataErrorInfo Validation

The IDataErrorInfo interface requires the implementer to expose one property and one indexer:

```
public interface IDataErrorInfo {
    string Error { get; }
    string this[string propertyName] { get; }
}
```

The Error property is used to indicate an error for the object as a whole, and the indexer is used to indicate errors at the individual property level. They both work the same: returning a non-null or non-empty string indicates a validation error. In addition, the string you return can be used to display the error to the user, as I'll show later.

When you're working with individual controls bound to individual properties on a data source object, the most important part of the interface is the indexer. The Error property is used only in scenarios such as when the object is displayed in a DataGrid or in a Binding-Group. The Error property is used to indicate an error at the row level, whereas the indexer is used to indicate an error at the cell level.

Implementing IDataErrorInfo has one big downside: the implementation of the indexer typically leads to a big switch-case statement, with one case for each property name in the object, and you have to switch and match based on strings and return strings to indicate an error. Furthermore, your implementation of IDataErrorInfo is not called until the property value has already been set on the object. If other objects have subscribed to INotifyPropertyChanged.PropertyChanged on your object, they will already have been notified of the change and could have started working based on data that your IDataErrorInfo implementation is about to declare invalid. If that could be a problem for your application, you'll need to throw exceptions from the property setters when you're unhappy with the value being set.

The good thing about IDataErrorInfo is that it makes it easy to address cross-coupled properties. For example, in addition to using the ValidationRule to validate the input format of the Inventory field, remember the requirement that the Inventory field must be filled in when the ActivityType is Install. The ValidationRule itself

has no access to the other properties on the data-bound object. It just gets passed a value that's being set for the property the Binding is hooked up to. To address this requirement, when the ActivityType property gets set you need to cause validation to occur on the Inventory property and return an invalid result when ActivityType is set to Install if the value of Inventory is empty.

To accomplish this, you need IDataErrorInfo so that you can inspect both the Inventory and ActivityType properties when evaluating Inventory, as shown here:

```
public string this[string propertyName] {
    get { return IsValid(propertyName); }
}
```

```
private string IsValid(string propertyName) {
    switch (propertyName) {
        ...
        case "Inventory":
            if (ActivityType != null &&
                ActivityType.Name == "Install" &&
                string.IsNullOrEmpty(Inventory))
                return "Inventory expended must be entered for installs";
            break;
    }
}
```

Additionally, you need to get the Inventory Binding to invoke validation when the ActivityType property changes. Normally, a Binding only queries the IDataErrorInfo implementation or calls ValidationRules if that property changed in the UI. In this case, I want to trigger the re-evaluation of the Binding validation even though the Inventory property has not changed, but the related ActivityType has.

There are two ways to get the Inventory Binding to refresh when the ActivityType property changes. The first and simplest way is to publish the PropertyChanged event for Inventory when you set the ActivityType:

```
ActivityType _ActivityType;
public ActivityType ActivityType {
    get { return _ActivityType; }
    set {
        if (value != _ActivityType) {
            _ActivityType = value;
            PropertyChanged(this,
                new PropertyChangedEventArgs("ActivityType"));
            PropertyChanged(this,
                new PropertyChangedEventArgs("Inventory"));
        }
    }
}
```

This causes the Binding to refresh and re-evaluate the validation of that Binding.

The second way is to hook the Binding.SourceUpdated attached event on the ActivityType ComboBox or one of its parent elements, and trigger a Binding refresh from the code-behind handler for that event:

```
<ComboBox Name="activityTypeIdComboBox"
    Binding.SourceUpdated="OnPropertySet"...

private void OnPropertySet(object sender,
    DataTransferEventArgs e) {

    if (activityTypeIdComboBox == e.TargetObject) {
        inventoryTextBox.GetBindingExpression(
            TextBox.TextProperty).UpdateSource();
    }
}
```



Calling `UpdateSource` on a `Binding` programmatically causes it to write the current value in the bound target element into the source property, triggering the validation chain as if the user had just edited the control.

## Using `BindingGroup` for Cross-Coupled Properties

The `BindingGroup` feature was added in the Microsoft .NET Framework 3.5 SP1. A `BindingGroup` is specifically designed to allow you to evaluate validation on a group of bindings all at once. For example, you could allow a user to fill in an entire form and wait until she pressed the Submit or Save button to evaluate the validation rules for the form, then present the validation errors all at once. In the sample application, I had the requirement that at least one Customer, Objective, or Reason had to be provided. A `BindingGroup` can be used to evaluate a subset of a form as well.

To use a `BindingGroup`, you need a set of controls with normal Bindings on them that share a common ancestor element. In the sample application, the Customer `ComboBox`, Objective `ComboBox` and Reason `TextBox` all live within the same `Grid` for layout. `BindingGroup` is a property on `FrameworkElement`. It has a `ValidationRules` collection property that you can populate with one or more `ValidationRule` objects. The following XAML shows the `BindingGroup` hookup for the sample application:

```
<Grid>...
<Grid.BindingGroup>
  <BindingGroup>
    <BindingGroup.ValidationRules>
      <local:CustomerObjectiveOrReasonValidationRule
        ValidationStep="UpdatedValue"
        ValidatesOnTargetUpdated="True"/>
    </BindingGroup.ValidationRules>
  </BindingGroup>
</Grid.BindingGroup>
</Grid>
```

In this example, I added an instance of the `CustomerObjectiveOrReasonValidationRule` to the collection. The `ValidationStep` property allows you to have some control over the value that's passed to the rule. `UpdatedValue` means to use the value that was written to the data source object after it is written. You can also choose values for `ValidationStep` that let you use the raw input from the user, the value after type and value conversion is applied, or the "committed" value, which means implementing the `IEditableObject` interface for transactional changes to the properties of your object.

The default way WPF displays validation errors is to draw a red border around the control.

The `ValidatesOnTargetUpdated` flag causes the rule to be evaluated each time the target property is set through the Bindings. This includes when it is set initially, so you have immediate validation error indications if the initial data is invalid, as well as each time the user changes the values in the controls that are part of the `BindingGroup`.

A `ValidationRule` that is hooked up to a `BindingGroup` works a little differently than a `ValidationRule` hooked up to a single Bind-

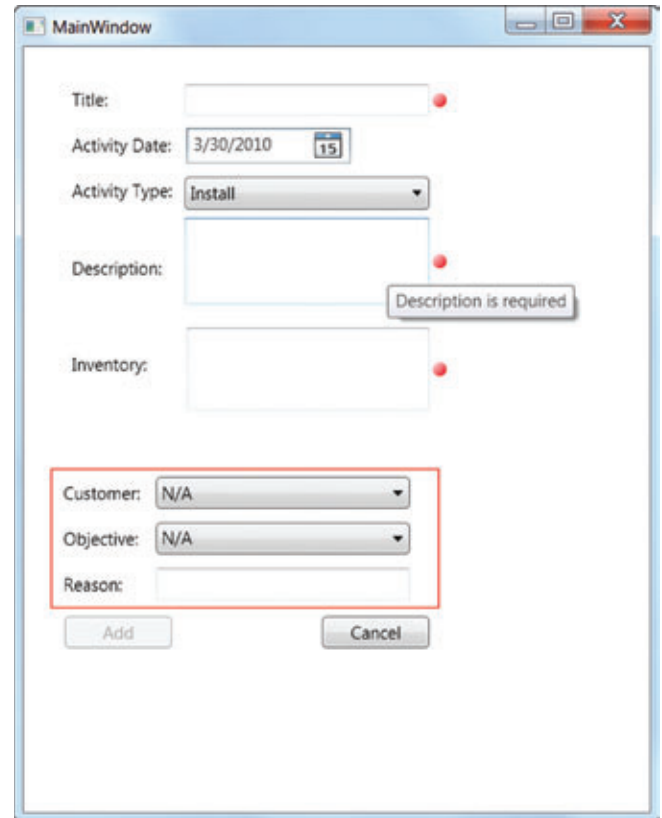


Figure 4 A Dialog Showing ToolTips and Invalid Data

ing. **Figure 7** shows the `ValidationRule` hooked up to the `BindingGroup` shown in the previous code sample.

In a `ValidationRule` hooked up to a single `Binding`, the value that's passed in is the single value from the data source property that's set as the Path of the Binding. In the case of a `BindingGroup`, the value that is passed to the `ValidationRule` is the `BindingGroup` itself. It contains an `Items` collection that is populated by the `DataContext` of the containing element, in this case the `Grid`.

For the sample application, I'm using the MVVM pattern, so the `DataContext` of the view is the `ViewModel` itself. The `Items` collection contains just a single reference to the `ViewModel`. From the `ViewModel`, I can get to the `Activity` property on it. The `Activity` class in this case has the validation method that determines whether at least one Customer, Objective, or Reason has been entered so I don't have to duplicate that logic in the `ValidationRule`.

As with other `ValidationRules` covered earlier, if you're happy with the values of the data passed in, you return a `ValidationResult.ValidResult`. If you're unhappy, you construct a new `ValidationResult` with a false valid flag and a string message indicating the problem, which can then be used for display purposes.

Setting the `ValidatesOnTargetUpdated` flag is not enough to get the `ValidationRules` to fire automatically, though. The `BindingGroup` was designed around the concept of explicitly triggering validation for an entire group of controls, typically through something like a Submit or Save button press on a form. In some scenarios, users don't want to be bothered with validation error indications until they consider the editing process complete, so the `BindingGroup` is designed with this approach in mind.



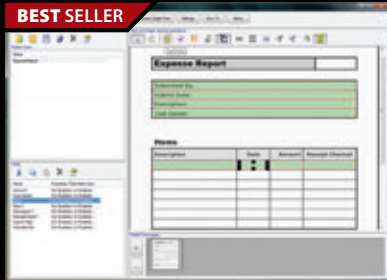
BEST SELLER

**FusionCharts** | from \$195.02

Interactive and animated charts for ASP and ASP.NET apps.

- Liven up your Web applications using animated Flash charts
- Create AJAX-enabled charts that can change at client-side without invoking server requests
- Export charts as images/PDF and data as CSV for use in reporting
- Also create gauges, financial charts, Gantt charts, funnel charts and over 550 maps
- Used by over 15,000 customers and 250,000 users in 110 countries

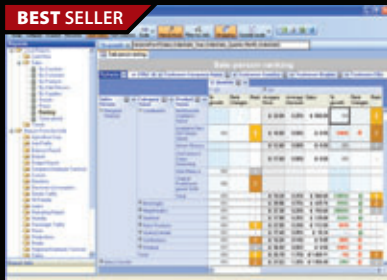
BEST SELLER

**LEADTOOLS Recognition SDK** | from \$3,595.50

Add robust 32/64 bit document imaging &amp; recognition functionality into your applications.

- Features accurate, high-speed multi-threaded OCR and forms recognition
- Supports text, OMR, image, and 1D/2D barcode fields
- Auto-registration and clean-up to improve recognition results
- Includes .NET, C/C++, WPF, WF, WCF and Silverlight interfaces
- Includes comprehensive confidence reports to assess performance

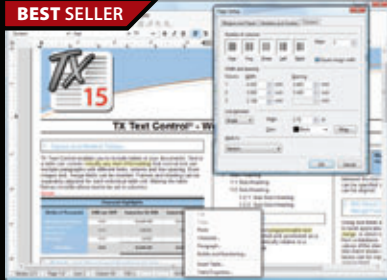
BEST SELLER

**ContourCube** | from \$900.00

OLAP component for interactive reporting and data analysis.

- Embed Business Intelligence functionality into database applications
- Zero report coding - design reports with drag and drop
- Self-service interactive reporting - get hundreds of reports by managing rows/columns
- Royalty free - only development licenses are needed
- Provides extremely fast processing of large data volumes

BEST SELLER

**TX Text Control .NET and .NET Server** | from \$499.59

Word processing components for Visual Studio .NET.

- Add professional word processing to your applications
- Royalty-free Windows Forms text box
- True WYSIWYG, nested tables, text frames, headers and footers, images, bullets, structured numbered lists, zoom, dialog boxes, section breaks, page columns
- Load, save and edit DOCX, DOC, PDF, PDF/A, RTF, HTML, TXT and XML



In the sample application, I want to provide immediate validation-error feedback to the user any time he changes something in the form. To do that with a `BindingGroup`, you have to hook the appropriate change event on the individual input controls that are part of the group, and have the event handler for those events trigger the evaluation of the `BindingGroup`. In the sample application, this means hooking the `ComboBox.SelectionChanged` event on the two `ComboBoxes` and the `TextBox.TextChanged` event on the `TextBox`. Those all can point to a single handling method in the code-behind:

```
private void OnCommitBindingGroup(
    object sender, EventArgs e) {

    CrossCoupledPropsGrid.BindingGroup.CommitEdit();
}
```

Note that for the validation display, the default red border will be displayed on the `FrameworkElement` that the `BindingGroup` resides on, such as the `Grid` in the sample application, as in **Figure 4**. You can also alter where the validation indication is displayed by using the `Validation.ValidationAdornerSite` and `Validation.ValidationAdornerSiteFor` attached properties. By default, the individual controls will also display red borders for their individual validation errors. In the sample application, I turn those borders off by setting the `ErrorTemplate` to null through `Styles`.

With `BindingGroup` in the .NET Framework 3.5 SP1, you may encounter problems with the proper display of validation errors on initial form load, even if you set the `ValidatesOnTargetUpdated` property on the `ValidationRule`. A workaround I found for this was to “jiggle” one of the bound properties in the `BindingGroup`. In the sample application, you could add and remove a space at the end of whatever text is initially presented in the `TextBox` in the `Loaded` event of the view like so:

```
string originalText = m_ProductTextBox.Text;
m_ProductTextBox.Text += " ";
m_ProductTextBox.Text = originalText;
```

This causes the `BindingGroup` `ValidationRules` to fire since one of the contained `Binding` properties has changed, causing the validation of each `Binding` to be called. This behavior is fixed in the .NET Framework 4.0, so there should be no need for the workaround to get initial display of validation errors—just set the `ValidatesOnTargetUpdated` property to true on the validation rules.

## Validation Error Display

As mentioned previously, the default way WPF displays validation errors is to draw a red border around the control. Often you’ll want to customize this to display errors in some other way. Moreover, the error message associated with the validation error is not displayed by default. A common requirement is to display the error message in a `ToolTip` only when the validation error exists. Customizing the validation error displays is fairly easy through a combination of `Styles` and a set of attached properties associated with validation.

To add a `ToolTip` that displays the error text is trivial. You just need to define a `Style` that applies to the input control that sets the `ToolTip` property on the control to the validation error text whenever there



Figure 5 A Validation Error

is a validation error. To support this, there are two attached properties you’ll need to employ: `Validation.HasError` and `Validation.Errors`. A `Style` targeting the `TextBox` type that sets the `ToolTip` is shown here:

```
<Style TargetType="TextBox">
    <Style.Triggers>
        <Trigger Property="Validation.HasError"
            Value="True">
            <Setter Property="ToolTip">
                <Setter.Value>
                    <Binding
                        Path="(Validation.Errors).CurrentItem.ErrorContent"
                        RelativeSource="{x:Static RelativeSource.Self}" />
                    </Setter.Value>
                </Setter>
            </Trigger>
        </Style.Triggers>
    </Style>
```

You can see that the `Style` just contains a property trigger for the `Validation.HasError` attached property. The `HasError` property will be set to true when a `Binding` updates its source object property and the validation mechanisms generate an error. That could come from an exception, `ValidationRule` or `IDataErrorInfo` call. The `Style` then uses the `Validation.Errors` attached property, which will contain a collection of error strings if a validation error exists. You can use the `CurrentItem` property on that collection type to just grab the first string in the collection. Or you could design something that data binds to the collection and displays the `ErrorContent` property for each item in a list-oriented control.

To change the default validation error display for a control to something other than the red border, you will need to set the `Validation.ErrorTemplate` attached property to a new template on the control you want to customize. In the sample application, instead of displaying a red border, a small red gradient circle is displayed to the right of each control with an error. To do that, you define a control template that will be used as the `ErrorTemplate`.

Figure 6 `ValidationRule` to Validate a String Array

```
public class InventoryValidationRule : ValidationRule {

    public override ValidationResult Validate(
        object value, CultureInfo cultureInfo) {

        if (InventoryPattern == null)
            return ValidationResult.ValidResult;

        if (!(value is string))
            return new ValidationResult(false,
                "Inventory should be a comma separated list of model numbers as a string");

        string[] pieces = value.ToString().Split(',');
        Regex m_RegEx = new Regex(InventoryPattern);

        foreach (string item in pieces) {
            Match match = m_RegEx.Match(item);
            if (match == null || match == Match.Empty)
                return new ValidationResult(
                    false, "Invalid input format");
        }

        return ValidationResult.ValidResult;
    }

    public string InventoryPattern { get; set; }
}
```

Figure 7 ValidationRule for a BindingGroup

```
public class CustomerObjectiveOrReasonValidationRule :  
    ValidationRule {  
  
    public override ValidationResult Validate(  
        object value, CultureInfo cultureInfo) {  
  
        BindingGroup bindingGroup = value as BindingGroup;  
        if (bindingGroup == null)  
            return new ValidationResult(false,  
                "CustomerObjectiveOrReasonValidationRule should only be used with  
a BindingGroup");  
  
        if (bindingGroup.Items.Count == 1) {  
            object item = bindingGroup.Items[0];  
            ActivityEditorViewModel viewModel =  
                item as ActivityEditorViewModel;  
            if (viewModel != null && viewModel.Activity != null &&  
                !viewModel.Activity.CustomerObjectiveOrReasonEntered())  
                return new ValidationResult(false,  
                    "You must enter one of Customer, Objective, or Reason to a valid entry");  
        }  
        return ValidationResult.ValidResult;  
    }  
}
```

```
<ControlTemplate x:Key="InputErrorTemplate">  
    <DockPanel>  
        <Ellipse DockPanel.Dock="Right" Margin="2,0"  
            ToolTip="Contains invalid data"  
            Width="10" Height="10">  
            <Ellipse.Fill>  
                <LinearGradientBrush>  
                    <GradientStop Color="#1FF111" Offset="0" />  
                    <GradientStop Color="#FFFF00" Offset="1" />  
                </LinearGradientBrush>  
            </Ellipse.Fill>  
        </Ellipse>  
        <AdornedElementPlaceholder />  
    </DockPanel>  
</ControlTemplate>
```

To hook up that control template to a control, you just need to set the `Validation.ErrorTemplate` property for the control, which you can again do through a `Style`:

```
<Style TargetType="TextBox">  
    <Setter Property="Validation.ErrorTemplate"  
        Value="{StaticResource InputErrorTemplate}" />  
    ...  
</Style>
```

## Wrap Up

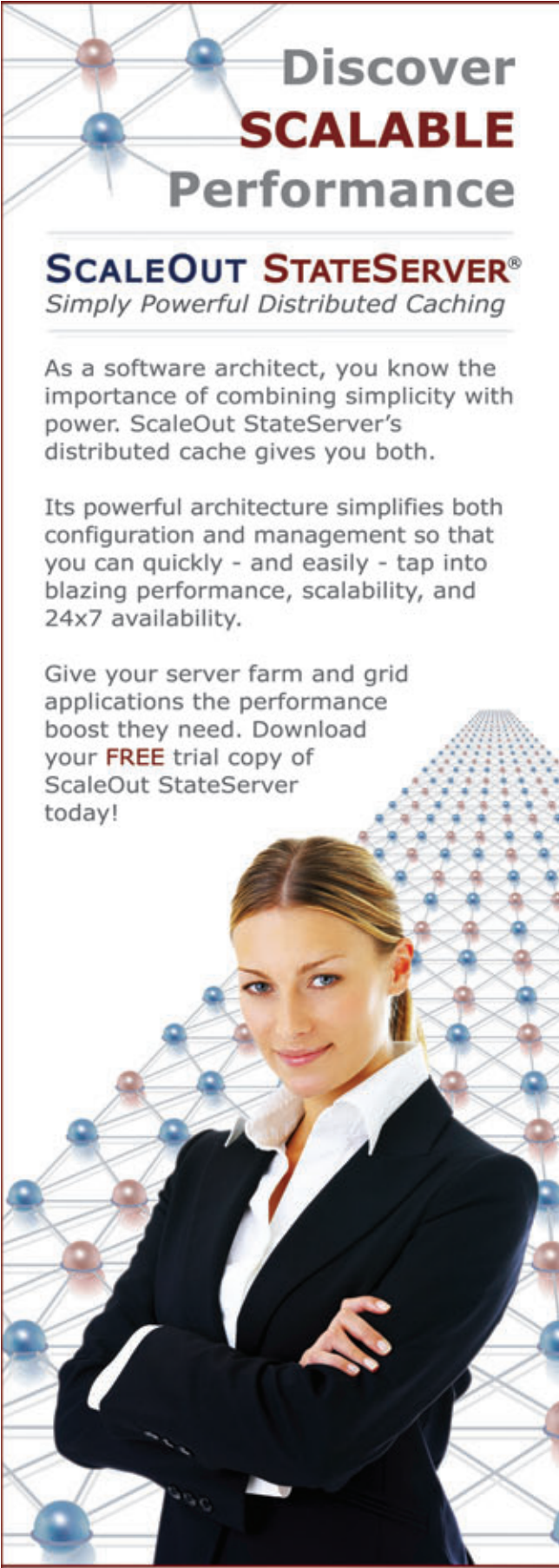
In this article, I've shown how you can use the three validation mechanisms of WPF data binding to address a number of business data validation scenarios. You saw how to use exceptions, `ValidationRules`, and the `IDataErrorInfo` interface to address single property validation, as well as properties whose validation rules depend on the current values of other properties on the control. You also saw how to use `BindingGroups` to evaluate several Bindings at once, and how to customize the display of errors beyond the defaults of WPF.

The sample application for this article has the full set of validation that satisfies the described business rules in a simple application that uses MVVM to hook up the view to the data supporting it. ■

**BRIAN NOYES** is chief architect of *IDesign* ([idesign.net](http://idesign.net)), a Microsoft regional director and Microsoft MVP. Noyes is an author and a frequent speaker at Microsoft Tech·Ed, DevConnections, DevTeach and other conferences worldwide. Contact him through his blog at [briannoyes.net](http://briannoyes.net).

**THANKS** to the following technical expert for reviewing this article:  
Sam Bent

[msdnmagazine.com](http://msdnmagazine.com)



**Discover**  
**SCALABLE**  
**Performance**

**SCALEOUT STATESERVER®**  
*Simply Powerful Distributed Caching*

As a software architect, you know the importance of combining simplicity with power. ScaleOut StateServer's distributed cache gives you both.

Its powerful architecture simplifies both configuration and management so that you can quickly - and easily - tap into blazing performance, scalability, and 24x7 availability.

Give your server farm and grid applications the performance boost they need. Download your **FREE** trial copy of ScaleOut StateServer today!

**SCALEOUT SOFTWARE**  
[www.scaleoutsoftware.com/eval](http://www.scaleoutsoftware.com/eval) | (503) 643-3422



# Building Rich Internet Apps with the Open Data Protocol

Shayne Burgess

At PDC09 the **Microsoft** WCF Data Services team (formerly known as the ADO.NET Data Services team) first unveiled OData, the Open Data Protocol. The announcement was in a keynote on the second day of the conference, but that wasn't where OData started. People familiar with ADO.NET Data Services have been using OData as the data transfer protocol for resource-based applications since ADO.NET Data Services became available in the Microsoft .NET Framework 3.5 SP1. In this article, I'll explain how developers of Rich Internet Applications (RIAs) can use OData, and I'll also show the benefits of doing so.

I'll start by answering the No. 1 question I've been asked since the unveiling of OData in November: What is it? In very simple terms, OData is a resource-based Web protocol for querying and

updating data. OData defines operations on resources using HTTP verbs (PUT, POST, UPDATE and DELETE), and it identifies those resources using a standard URI syntax. Data is transferred over HTTP using the AtomPub or JSON standards. For AtomPub, the OData protocol defines some conventions on the standard to support the exchange of query and schema information. Visit [odata.org](http://odata.org) for more information on OData.

## The OData Ecosystem

In this article, I'll introduce a few products, frameworks and Web Services that consume or produce OData feeds. The protocol defines the resources and methods that can be operated on and the operations (GET, PUT, POST, MERGE and DELETE, which correspond to read, create, replace, merge and delete) that can be performed on those resources.

In practice this means any client that can consume the OData protocol can operate over any of the producers. It's not necessary to learn the programming model of a service to program against the service; it's only necessary to choose the target language to program in.

If, for example, you're a Silverlight developer who learns the OData library for that platform, you can program against any OData feed. Beyond the OData library for Silverlight you'll find libraries for the Microsoft .NET Framework client, AJAX, Java, PHP and Objective-C, with more on the way. Also, Microsoft PowerPivot for Excel supports an OData feed as one of the options for data import to its in-memory analysis engine.

### This article discusses:

- The OData ecosystem
- What's new in WCF Data Services
- Using OData with SharePoint
- Consuming Open Government Data Initiative services
- Using OData with Silverlight
- Using OData with PowerPivot

### Technologies discussed:

OData, Silverlight, WCF Data Services, Entity Framework, SharePoint, Windows Azure Platform, PowerPivot

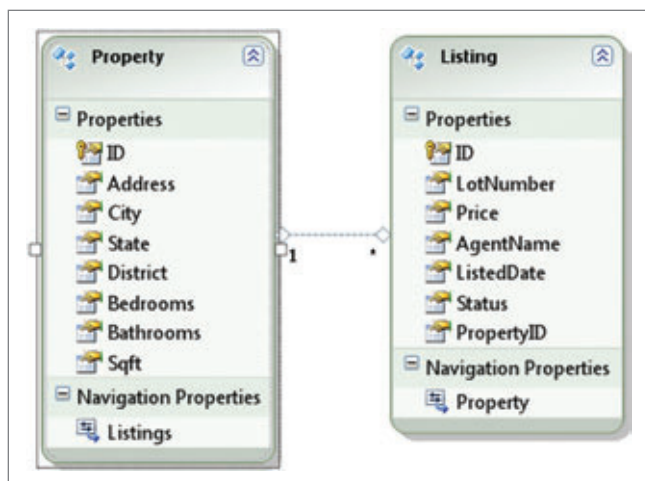


Figure 1 The Entity Framework Data Model for the Relational Data

And just as clients capable of consuming the OData protocol can operate over any of the producers, a service or application created using OData can be consumed by any OData-enabled client. After creating a Web service that exposes relational data as an OData endpoint (or exposes the data in a SharePoint site, tables in Windows Azure or what have you), you can easily build a rich desktop client in the .NET Framework or a rich AJAX-based Web site that consumes the same data.

The long-term goal for OData is to have an OData client library for every major technology, programming language and platform so that every client app can consume the wealth of OData feeds. Combined, the producers and consumers of OData create an OData “ecosystem.”

## What’s New in WCF Data Services?

WCF Data Services, a component of the .NET Framework, is a framework that offers a turnkey solution for creating OData Web services and includes a client library with which you can build clients that consume OData feeds. The WCF Data Services team recently released an update to the .NET Framework 3.5 SP1 that introduces a host of new features you’ll also find in the .NET Framework 4. This is the second version of the Data Services framework. Visit [blogs.msdn.com/astoriateam/archive/2010/01/27/data-services-update-for-net-3-5-sp1-available-for-download.aspx](http://blogs.msdn.com/astoriateam/archive/2010/01/27/data-services-update-for-net-3-5-sp1-available-for-download.aspx), where you’ll find a description and a link for downloading.

The WCF Data Services framework is not just a protocol for RIA applications. It was also designed for high-scale service developers and has many features that appeal to them, such as server paging limits, HTTP caching support, stateless services, streaming support and a pluggable provider model. Let’s look at the new features that are generally of most interest to RIA developers.

One of the top feature wishes customers expressed after the initial release was the ability to request the number of entities in a set. The new “count” feature addresses that need with two parts. First, it lets you request only the count—that is, the number of values a query would return. Second, it adds a query option that tells the service to include a count of the total number of entities in a set when the query result is a partial set (for example, when server paging is enabled).

To enhance the experience when binding data from an OData service, a new type, `DataServiceCollection`, has been added to the

WCF Data Services client library. It implements change tracking on the items it contains (through the use of the `INotifyPropertyChanged` and `INotifyCollectionChanged` interfaces). When it’s bound to a control—a `DataGrid` in Silverlight, for example—it will track the changes made to the objects and to the collection itself. This new collection greatly simplifies the process of creating OData clients with an interface component.

Another frequently requested feature was the ability to project a subset of the properties of an entity returned in a query result. LINQ support has been added for this through the LINQ Select statement. This has two benefits: It reduces the size of the HTTP responses to queries, and it reduces the memory footprint of the client-side objects. This can be especially useful when you’re developing a client application against a service you don’t own and in which each entity may have many properties of no interest to the client. Later in this article, I’ll demonstrate working with a large, publicly available service that has many entities with numerous properties on each entity. Projections will be useful in the example because it includes only a few needed properties on one entity.

To help you understand the value of the OData ecosystem, we’ll create a Web application that lets visitors browse the site of my fictional real estate company, Contoso Ltd., to see the listings of the properties it manages.

## Relational Data

The main data source for the Contoso.com Home Finder application is a SQL Server database that contains information about all of the properties the company is managing and all of the listings (current and previously sold) it has published for those properties.

Since the release of WCF Data Services and the ADO.NET Entity Framework in the .NET Framework 3.5 SP1, it has been easy to expose a relational database as an OData feed. All that’s needed is an Entity Framework model created over the relational data. An OData feed is HTTP-based, so you need a Web site or Web service to host the service.

To create the OData feed over the relational data, the first step is to create an ASP.NET Web application in Visual Studio 2010 to host the OData service. In Visual Studio, select **File | New | Project | ASP.NET Web Application**. This will create the skeleton of a Web service that can be used to host the OData feed.

Figure 2 Defining the WCF Data Service

```
// The ListingsEntities is the Entity Framework Context that the Service exposes
public class Listings : DataService<ListingsEntities>
{
    public static void InitializeService(DataServiceConfiguration config)
    {
        // These lines set the access rights to "Read Only" for both entity sets
        config.SetEntitySetAccessRule("Listings", EntitySetRights.AllRead);
        config.SetEntitySetAccessRule("Properties", EntitySetRights.AllRead);

        // There are currently no service operations in the service
        config.SetServiceOperationAccessRule("MyServiceOperation",
            ServiceOperationRights.All);

        config.DataServiceBehavior.MaxProtocolVersion =
            DataServiceProtocolVersion.V2;
    }
}
```



```

<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <service xmlns:base="http://host/Agents/_vti_bin/listdata.svc/" xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:app="http://www.w3.org/2007/app" xmlns="http://www.w3.org/2007/app">
- <workspace>
  <atom:title>Default</atom:title>
  + <collection href="Agents">
  + <collection href="Announcements">
  + <collection href="Attachments">
  + <collection href="Calendar">
  + <collection href="CalendarCategory">
  + <collection href="Links">
  + <collection href="MasterPageGallery">
  + <collection href="MasterPageGalleryCompatibleUIVersionS">
  + <collection href="MyDocuments">
  + <collection href="SharedDocuments">
  + <collection href="SiteAssets">
  + <collection href="SitePages">
  + <collection href="Tasks">
  + <collection href="TasksPriority">
  + <collection href="TasksStatus">
  + <collection href="TeamDiscussion">
</workspace>
</service>

```

Figure 3 Service Document for the SharePoint Site

After the Web service is created and configured, we'll create the Entity Framework data model that the OData feed will expose. Visual Studio makes this easy using the Add New Item wizard, which lets you auto-generate a model from an existing database. **Figure 1** shows a simple data model created using the Add New Item wizard over the SQL Server data containing the properties and listings managed by Contoso.

Now let's create a WCF Data Service that exposes this data model as an OData feed. Visual Studio also makes this simple with the WCF Data Service option in the Add New Item wizard. When you select this option, Visual Studio provides a code file (in this example the file is called Listings.svc.cs) that's used to configure the Data Service.

The code in **Figure 2** demonstrates how to define a WCF Data Service. The Listings class is the service class that's exposing the Data Service, and it implements the generic DataService<T>. The type used to define the DataService<T> in **Figure 2** is the Listings-Entities type, which is the Entity Framework context created in **Figure 1**. Because this class will accept an Entity Framework context, this is a quick and easy way to get a WCF Data Service that exposes relational data up and running. The DataService class

Figure 4 OData Query Options

\$top=n	Restricts the query to the first n entities.
\$skip=n	Skips the first n entities in the set.
\$inlinecount=allpages	Includes the count of all entities of the set in the result.
\$filter=<expression>	An expression can be supplied to restrict the results returned by the query (example: \$filter=Status eq 'Available' restricts the results to entities that have a Status property with the value "Available").
\$orderby=<expression>	Orders the results by a set of properties of the entity
\$select=<expression>	Specifies a subset of the properties of the entity to be returned.
\$format	Specifies the format of the feed to be returned (ATOM or JSON). This option is not supported in WCF Data Services.

isn't restricted to just working over Entity Framework contexts, however; the class will accept any collection of CLR objects that implements the IQueryable interface. In the .NET Framework 4, a new custom provider model for WCF Data Services has been added that allows a service to be created over almost any data source.

Let's look a little closer at what else the InitializeService method in **Figure 2** is doing. The method is calling the Set-EntitySetAccessRule for both of the entity sets the service will expose and setting the access rights to AllRead. This tells the service to make both entity sets fully readable but not allow any inserts,

updates or deletes. This is a great way to control access to the service. WCF Data Services also support methods called Query Interceptors that allow the service author to configure finer-grained access control for the service on a per-entity-set basis. Set the Listings.svc file as the project start page and run the project. A browser window will open and display the service document, as shown in **Figure 3**.

The Open Government Data Initiative is a service built on the Microsoft Windows Azure platform that makes it easier for government agencies to publish a wide variety of public data.

## OData URI Conventions

The service document lists the entity sets that are exposed by the service. Remember, you can access the resources in this service using the powerful URI syntax defined as an optional part of the OData protocol. Let's take a quick look at the URI syntax for this service. To access a feed for each entity set, you append the name of the entity set to the base URI for the service; for example, http://myhost/Listings.svc/Properties would address the set of entities in the Properties entity set.

It's also possible to address a particular entity individually using its key value; the URI http://myhost/Listings.svc/Properties(0) would address the property with ID = 0. You can address a relationship from this entity to another entity or set of entities by appending the name of the relationship to the end of the URI, so http://myhost/Listings.svc/Properties(0)/Listings would access the set of listings associated with the property entity with ID = 0. Using this syntax, it's possible to navigate through many levels of relationships.

The URI syntax also defines a number of query options that can be appended to a URI to modify the base query in some way, and

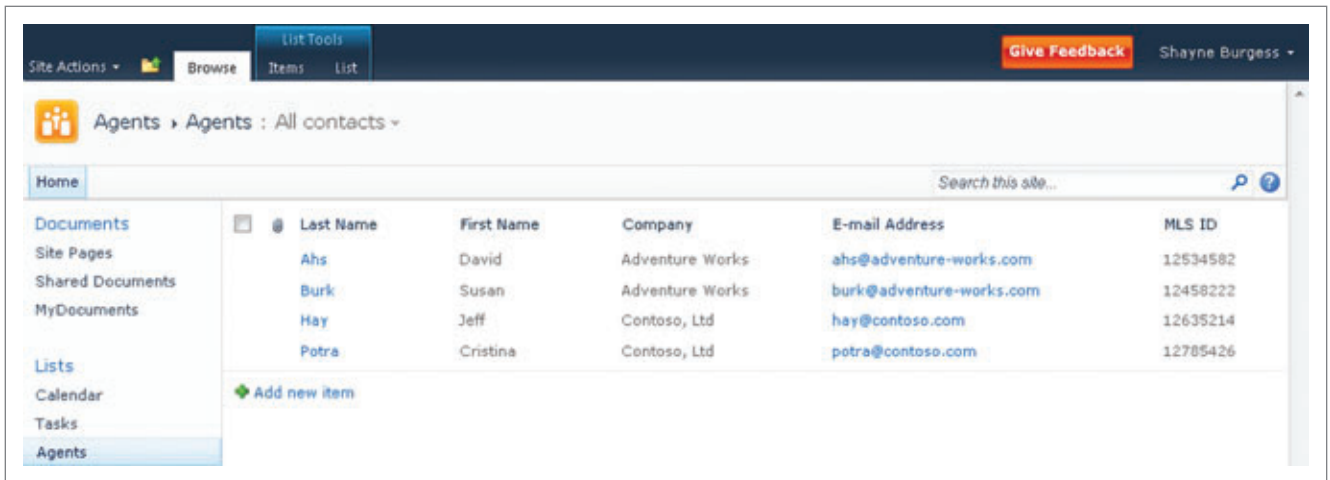


Figure 5 SharePoint Site for Agent Information

each query option is defined as a name/value pair. For example, by appending the query option \$top=10, you restrict the query to only the first 10 entries in the result. **Figure 4** lists all of the query options available in the URI syntax.

## Exposing Data from SharePoint

In the preceding section I showed you how to expose the data stored in my relational database, the property and listing information for the real estate Web site. Let's say I also have information about the real estate agents who are selling the properties, but that data is stored in a SharePoint site. Microsoft SharePoint 2010 has the ability to expose all lists and documents within those lists as an OData feed. This is great for the real estate site because it means the agent information that company employees have entered is available as an OData

feed that can be included in the listings application I'm building. The users who have processes using the SharePoint interface for entering and updating this data don't have to change their workflow to suit my application. The data entered into the company SharePoint site is available in real time to the Listings application that's being created.

**Figure 5** shows the simple SharePoint portal the real estate agents use to record and update their contact information.

When the ADO.NET Data Services Update for the .NET Framework 3.5 SP1 is installed on the SharePoint system, a new HTTP endpoint becomes available for each site that exposes the list data as an OData feed. Because an OData feed is accessed using HTTP, it can be examined by using just Internet Explorer. **Figure 6** shows the feed for the agents list in SharePoint.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <feed xml:base="http://myhost/Agents/_vti_bin/listdata.svc"
  xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Agents</title>
  <id>http://myhost/Agents/_vti_bin/listdata.svc/Agents</id>
  <updated>2010-02-09T02:26:19Z</updated>
  <link rel="self" title="Agents" href="Agents" />
  + <entry m:etag="W/"3">
  + <entry m:etag="W/"3">
  - <entry m:etag="W/"2">
    <id>http://myhost/Agents/_vti_bin/listdata.svc/Agents(3)</id>
    <title type="text">Ahs</title>
    <updated>2010-02-05T12:36:49-08:00</updated>
    - <author>
      <name />
    </author>
    <link rel="edit" title="AgentsItem" href="Agents(3)" />
    <link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Attachments"
      type="application/atom+xml" type="feed" title="Attachments" href="Agents(3)/Attachments" />
    <category term="Microsoft.SharePoint.DataService.AgentsItem"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
    - <content type="application/xml">
      - <m:properties>
        <d:ContentTypeID>0x010600B4A2E60B049C444A9BA8A452D487768</d:ContentTypeID>
        <d:LastName>Ahs</d:LastName>
        <d:FirstName>David</d:FirstName>
        <d:FullName>David Ahs</d:FullName>
        <d:EMailAddress>ahs@adventure-works.com</d:EMailAddress>
        <d:Company>Adventure Works</d:Company>
        <d:JobTitle>Agent</d:JobTitle>
        <d:MLSID>12534582</d:MLSID>
        <d:ID m:type="Edm.Int32">3</d:ID>
        <d:ContentType>Contact</d:ContentType>
        <d:Modified m:type="Edm.DateTime">2010-02-05T12:36:49</d:Modified>
        <d:Created m:type="Edm.DateTime">2010-02-02T16:49:21</d:Created>
        <d:CreatedByID m:type="Edm.Int32">20</d:CreatedByID>
        <d:ModifiedByID m:type="Edm.Int32">20</d:ModifiedByID>
        <d:Owshiddenversion m:type="Edm.Int32">2</d:Owshiddenversion>
        <d:Version>1.0</d:Version>
        <d:Path>/Agents/Lists/Agents</d:Path>
      </m:properties>
    </content>
  </entry>
```

Figure 6 Agents Feed from the SharePoint Agent Service

## Consuming Reference Data from OGDl

By default, an OData feed will return an ATOM representation for the feed, and when accessed from a Web browser the result will be an ATOM feed. If the accept header of the request is changed to "application/json," the result will be the same data as a JSON feed.

The feed in **Figure 6** starts with a <feed> element that represents a set of entities. Contained within each feed is a set of <entry> elements, each of which represents a single entity in the feed (the first three entry elements are collapsed to make the whole feed visible in one screen).

In this example, the entity has a concurrency token defined on it; as a result, each entity in the feed has an etag property on it. The etag is the token used by the data service to enforce a concurrency check when a change is made to the requested entity. Each entity, formatted using an <entry> tag, consists of a set of



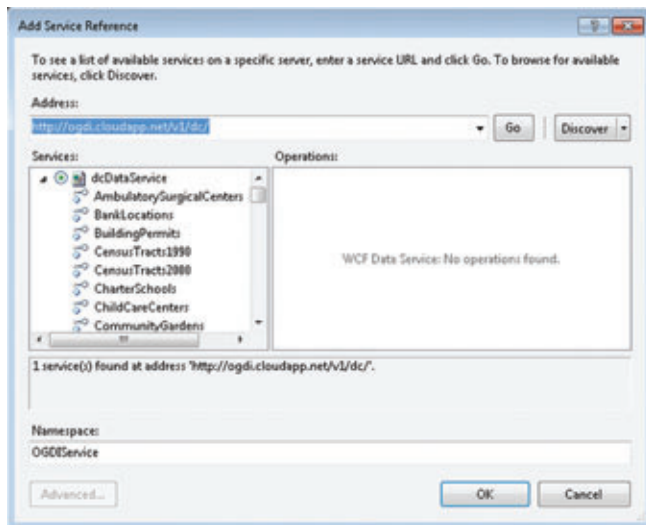


Figure 7 Add Service Reference for the OGDl Sample Service

links that contain both the link to be used when editing the entity and the entity's relationships. Each relationship link points either to another entity or to a set of entities (these are called reference and navigation properties, respectively). Each <entry> element also includes an <m:properties> element that contains the primitive and complex type properties for the entity; the property values consist of the name of the property on the entity and the value for that property.

The Open Government Data Initiative (OGDI) is a service built on the Microsoft Windows Azure platform that makes it easier for government agencies to publish a wide variety of public data. The OGDI project provides a starter kit that can be used by government agencies to expose their data. For example, the city of Edmonton has adopted the starter kit to expose its government data, and a service at [ogdisdk.cloudapp.net](http://ogdisdk.cloudapp.net) has a data set with a variety of data about the Washington, D.C., area. Another example is the Microsoft Codename "Dallas" project that aims to make it simple for anyone with a data set to expose the data as a service to the Web. This project is also built on the Windows Azure platform and exposes data using OData. These are examples of high-scale services that expose large reference data sets that can be further consumed by Web applications. As I will show, when these services expose their data using OData, it's simple to consume that data from a variety of applications.

As explained, the OGDI Web site features publicly available data about the Washington, D.C., area. Contoso's real estate application is used for browsing listings in that area, and it would be helpful for users to have available some of this reference data about the area around a particular property when viewing it. When I create the client for the sample application, I'll demonstrate how to include the OData feed from the OGDI Web site as one of the sources of data for the application.

## Other OData Producers

So far I've shown examples of consuming data from SQL Server, SharePoint and a generic OData service on the Web, but more options exist. The cloud-based Windows Azure platform has a table service that exposes data stored in Windows Azure tables, and the API for this is built using OData. As mentioned, the Microsoft Dallas project is a data

marketplace for finding and querying data exposed by the Dallas service, and this service exposes its data using the OData protocol. OData producers aren't just limited to Microsoft products either; IBM recently announced that its WebSphere eXtreme Scale 7.0 product now supports the OData protocol.

## Silverlight Client

Contoso's real estate finder application now has an ASP.NET Web service that exposes the relational data in SQL Server about the real estate listings and properties managed by the company; a SharePoint site that's being used to manage the data about company agents; and a government Web service that exposes data about the region around the properties the company is advertising. I want to put all of these sources together into one Silverlight application that can work with this data in a meaningful way.

In Silverlight 3, a WCF Data Services client library is included in the Silverlight SDK that makes it simple for Silverlight applications to communicate with a service that's OData-enabled. To do this, in Visual Studio from a Silverlight project, right-click the project and select Add Service Reference. This walks you through the process of creating a service reference. The main input to a service reference is the URI of the service that's being referenced from the Silverlight application. Figure 7 shows an example of adding a service reference to the OGDI sample service.

The service reference wizard creates a client-side context class that's used to interact with the data service. The client context

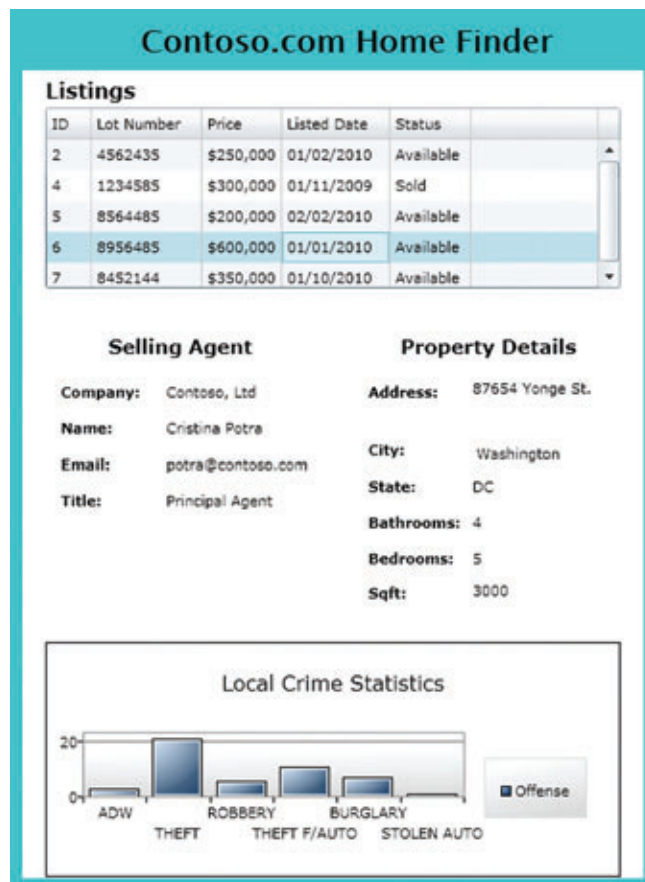


Figure 8 The Contoso Home Finder

# Speed • Reliability • Precision • Agility

*DynamicPDF...Proven .NET Components for Real-Time PDFs*



- Easy-to-use • Highly efficient
- Industry leading support • Huge feature set

**Try it FREE today!**

Experience our fully functional, never expiring evaluation and community editions.

## DynamicPDF Suite v6.0 for .NET

Our easy-to-use tools integrate with ASP.NET and ADO.NET allowing for the quick, real-time generation of PDF documents and reports.

For easy maintenance and deployment, our most popular tools are now available as a bundled suite.



Layout reports in DynamicPDF Designer with its Visual Studio look and feel.

## DynamicPDF Generator v6.0 for .NET

- Linearize/Fast Web View • JavaScript
- Encryption/Security • PDF/X-1a • PDF/A-1a/b
- Create Tagged PDFs • Interactive Form Fields
- Over 50 Ready-To-Use Page Elements Including 22 Bar Codes and Charting • Digital Signatures



## DynamicPDF Merger v6.0 for .NET

- Merge • Stamp • Append • Split • Password/Security
- Form-Fill • Outline and Annotation Preservation
- Place, Rotate, Scale and Clip Pages • Decryption



## DynamicPDF ReportWriter v6.0 for .NET

- GUI Report Designer • Event Driven • Recursive Sub-Reports
- Use PDF Templates • Automatic Pagination • Placeholders
- Record Splitting and Expansion • Column Support
- Full DynamicPDF Merger and Generator Integration



Also, check out our newest product, **DynamicPDF PrintManager for .NET**.

**ceTesoftware**  
INFINITE POSSIBILITIES

ceTe Software has been delivering quality software applications and components to our customers for over 10 years. Our DynamicPDF product line has proven our commitment to delivering innovative software components and our ability to respond to the changing needs of software developers. We back our products with a first class support team trained to provide timely, accurate and thorough responses to any support needs.

[www.cete.com](http://www.cete.com)  
[info@cete.com](mailto:info@cete.com)

800.631.5006  
+1 410.772.8620



Figure 9 Creating the Client Proxy Contexts

```
private void getListings()
{
    DataServiceCollection<Listing> listings = new
        DataServiceCollection<Listing>();

    listingsGrid.ItemsSource = listings;

    var query = from listing in
        listingsService.Listings.Expand("Property")
        select listing;
    listings.LoadAsync(query);
}
```

Figure 10 Executing an Asynchronous Query

```
private void GetAgentData(string agentName)
{
    var query = agentsService.Agents.Where(a => a.FullName == agentName) as
        DataServiceQuery;

    query.BeginExecute(
        AgentQueryCallback, query);
}

private void AgentQueryCallback(IAsyncResult result)
{
    Dispatcher.BeginInvoke(() =>
    {
        var queryResult = result.AsyncState as DataServiceQuery<AgentsItem>;
        if (queryResult != null)
        {
            var agents = queryResult.EndExecute(result);
            this.grdAgent.DataContext = agents.First();
        }
    });
}
```

abstracts the details of working with HTTP and URIs away from the client programming model and allows the client developer to think only about C# classes and XAML. The client context also includes a LINQ provider implementation and, as a result, LINQ queries on the proxy are supported. The Add Service Reference wizard will also generate a set of client proxy classes that mirror the types that are exposed by the referenced service. After creating the OGD I service reference, I will also create a service reference to both the SharePoint and Listings services I created. This code shows how to create the contexts that are used to interact with the three OData services:

```
// OGD I service context
OGDIService.dcDataService OGDIService =
    new dcDataService(new Uri(OGDIServiceURI));

// SharePoint service context
AgentsReference.AgentsDataContext agentsService =
    new AgentsReference.AgentsDataContext(new Uri(sharepointServiceURI));

// Listings Service context
ListingReference.ListingsEntities listingsService =
    new ListingReference.ListingsEntities(new Uri(listingsServiceURI));
```

Figure 8 shows the outline of the Silverlight real estate Home Finder application. The application will be hosted in SharePoint so that it's easily available for my existing users who are used to working in the SharePoint environment.

Figure 9 contains the code for querying the listings service and binding the result to the grid at the top of the Home Finder Silverlight application.

The code in Figure 9 creates a DataServiceCollection that's a tracked collection and binds the collection to the ItemsSource

property of the main listings grid. Because this collection implements change tracking, any changes made to the items in the grid will automatically be reflected on the entities in the listings collection. The changes in the grid can be persisted to the service by calling the BeginSaveChanges method on the context for the listings service.

In Silverlight, all network calls are made asynchronously, so executing any operations against a service using the WCF Data Services client library involves making the initial call to the operation and then writing a separate callback method that's passed to the method to handle the result of the asynchronous call. To improve this asynchronous experience, a method was added to the DataServiceCollection class, LoadAsync, which does all the work of handling the asynchronous callback function and loading the results into the collection.

In the Figure 9 code, the collection is bound to a grid before the LoadAsync call is made, and the values won't be loaded into the collection until after the asynchronous call completes. The collection will raise collection-changed events when the results are returned from the service, and the grid will catch those events and display the results when the asynchronous call completes.

When a listing is selected from the data grid, the SharePoint site needs to be queried to get the information about the agent managing that listing. In this application architecture, a second query is needed because the data sources for the listing type and the agent type are separate and there's no explicit relationship between the two (if you are a person who thinks in terms of models, this example involves two completely separate models, and the relationship between the models is an artificial one created and enforced by the client).

Figure 10 shows how to query the SharePoint service for the agent entity, given the name of the agent. A similar code sequence is used to query the OGD I data for the neighborhood statistics in the chart at the bottom of the Home Finder page. The code up to this point demonstrates only the query capabilities of the Silverlight client, but the client isn't limited to queries only; it has rich capabilities in writing back changes to the service from the client.

## OData in PowerPivot

PowerPivot is a new in-memory business intelligence tool that's delivered as an add-in for Microsoft Excel 2010—visit [powerpivot.com](http://powerpivot.com) for more information. The tool provides support for importing large data sets from a data source and doing complex data analysis and reporting. PowerPivot can import data from a number of different data sources, including directly from an OData feed. PowerPivot's From Data Feeds option (shown in Figure 11), accepts an OData service endpoint as the location of the feed to import.

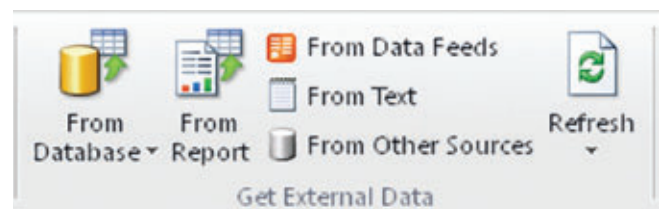
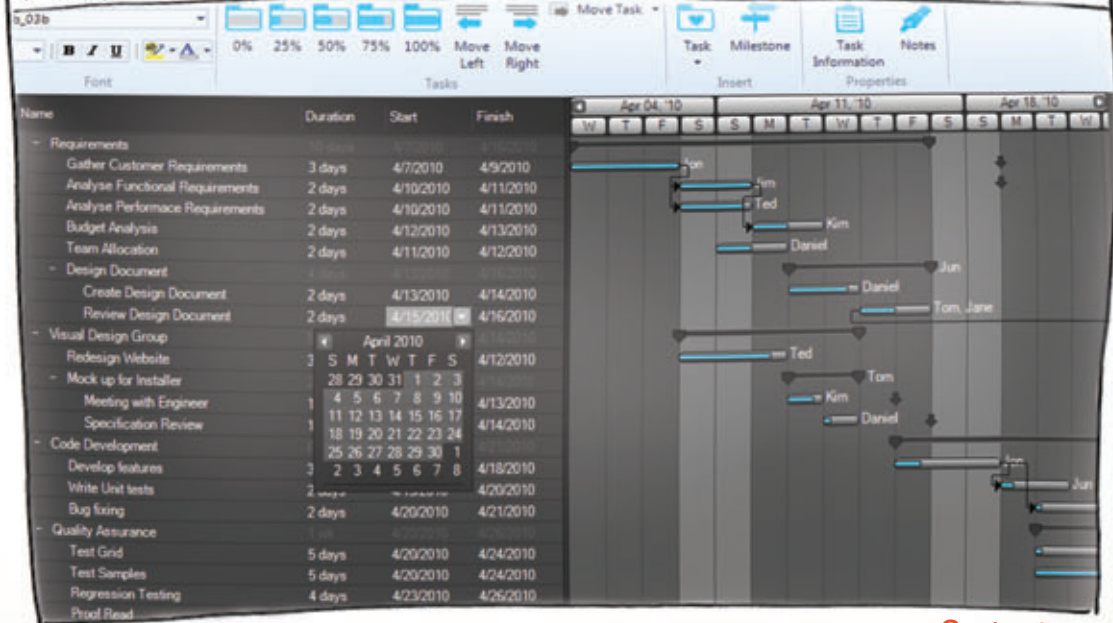


Figure 11 PowerPivot Imports from an OData Feed

# INNOVATION TO SPARK KILLER APPS

KILLER APPS. No Excuses.

3:53Q



Gantt Chart

You have the vision, but time, budget and staff constraints prevent you from seeing it through. With rich user interface controls like Gantt Charts that Infragistics **NetAdvantage® for .NET** adds to your Visual Studio 2010 toolbox, you can go to market faster with extreme functionality, complete usability and the “Wow-factor!” Go to [infragistics.com/spark](http://infragistics.com/spark) now to get innovative controls for creating Killer Apps.

**Infragistics**  
KILLER APPS. No Excuses.

Infragistics Sales 800 231 8588  
Infragistics Europe Sales +44 (0) 800 298 9055  
Infragistics India +91-80-6785-1111  
[twitter.com/infragistics](https://twitter.com/infragistics)



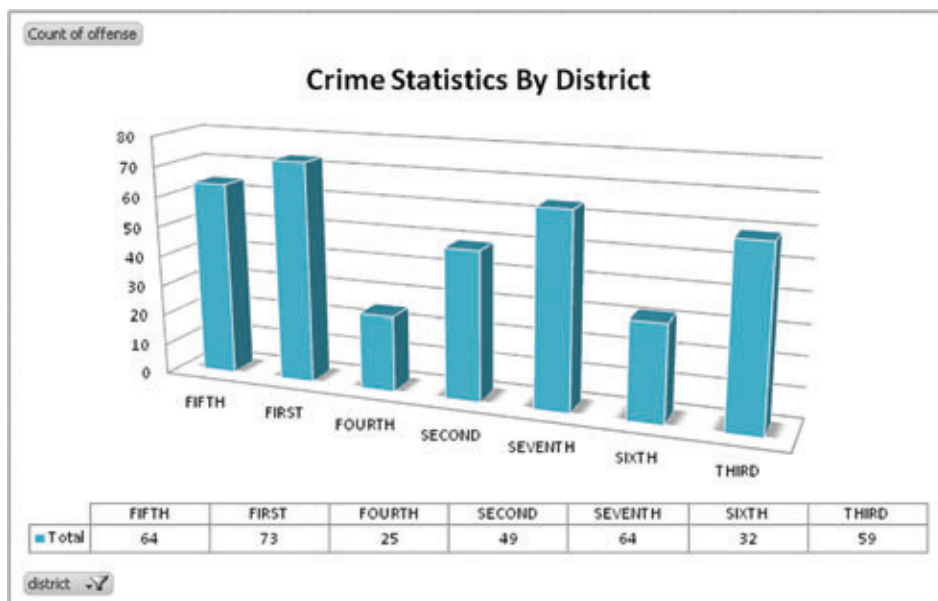


Figure 12 PowerPivot Chart from OData Feed

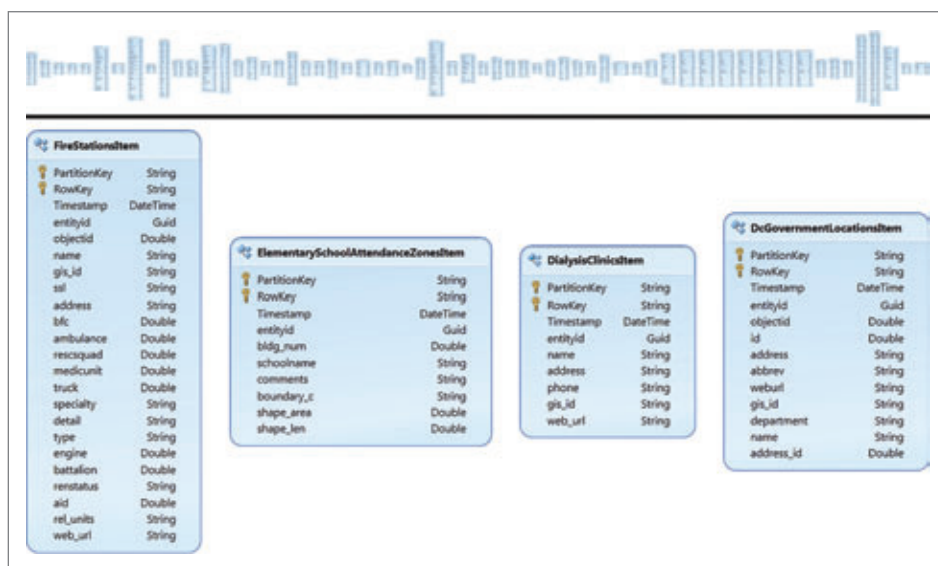


Figure 13 Open Data Visualizer Views of the OGD I Sample Service

Figure 12 shows a chart made from the summary of crime statistics in the OGD I's Washington, D.C., data feed.

The chart in Figure 12, made using the same data set as the real estate application in the previous example, shows a summary of all data for each district. I encourage you to download PowerPivot for Excel 2010 and import data from the OGD I site at [ogdi.cloudapp.net/v1/dc](http://ogdi.cloudapp.net/v1/dc), and see for yourself how quickly you'll be doing rich data analysis over this data.

## The Open Data Protocol Visualizer

The OGD I data service is essentially a "black box" to an external developer who creates an application that consumes the data exposed by the service. Thankfully, the OGD I service exposes its data using the OData protocol, so there's no need to know anything about the internal details of the service to interact with it. The

programming model for the service is the OData protocol. The service endpoint describes the shape of the data and, as I showed you in the previous section, that's all you need to interact with the service. However, it's often useful to view the shape of the data in the service and get a better understanding of the relationships between parts of the service. The Open Data Protocol Visualizer was created for just this purpose. It's available from the Tools | Extension Manager menu item in Visual Studio 2010. Figure 13 shows two views from the visualizer that display the structure of the OGD I service.

The top view in Figure 13 shows the entire service; the bottom view is zoomed in to show only four of the boxes in the display. The visualizer represents entity sets as boxes, and relationships between entities as lines connecting the boxes. It's clear from the top view in Figure 13 that the OGD I service is entirely flat and doesn't contain any relationships at all, as there are no connecting lines between any boxes. This is only a characteristic of the OGD I service and isn't typical of most OData services. The bottom view shows a close-up of four of the entity sets in the service. Just from examining the view, you can determine that the service exposes data about fire stations, elementary school attendance, dialysis clinics and government locations, as well as the properties and keys for each of those types.

## Learning More

This article introduces the Open Data Protocol and the ecosystem that has been built around it, including the WCF Data Services Framework. For more information, visit the Open Data Protocol Web site at [odata.org](http://odata.org). To learn more about WCF Data Services, see [msdn.microsoft.com/data/bb931106](http://msdn.microsoft.com/data/bb931106) or the WCF Data Services blog at [blogs.msdn.com/astoriateam](http://blogs.msdn.com/astoriateam).

**SHAYNE BURGESS** is a program manager in the Data and Modeling Group at Microsoft, working specifically on WCF Data Services and the Open Data Protocol. Burgess regularly blogs on the WCF Data Services team blog at [blogs.msdn.com/astoriateam](http://blogs.msdn.com/astoriateam).

**THANKS** to the following technical experts for reviewing this article:  
Elisa Flasko and Mike Flasko

# VSP<sup>2</sup>

## VISUAL STUDIO PARTNER PROFILE

# A Visual Studio 2010 Q&A with Daniel Jebaraj of Syncfusion

**Daniel Jebaraj**  
Vice President  
Syncfusion



*Syncfusion provides a broad range of enterprise-class .NET software components that help developers deliver business innovation in their applications.*

### Q How do developers typically use Syncfusion controls?

**A** Our customers range from small ISVs to global IT consultancies to Fortune 100 companies. Whether they're working in Windows Forms, ASP.NET, ASP.NET MVC, WPF, or Silverlight, they use our controls to deliver the elegant user interfaces and sophisticated reports that business users need, in the familiar formats they prefer.

### Q When did you first decide to integrate with Visual Studio 2010?

**A** Syncfusion made the decision as soon as the first CTP was available. Visual Studio 2010 was highly anticipated by the .NET developer community and we were eager to support it. The 2010 Volume 2 release of our Essential Studio suite sim-shipped with VS2010 in April and fully supports the new IDE.

### Q How do you help light up the new functionality of Visual Studio 2010?

**A** Our controls for WPF and Silverlight offer tight integration with the IDE enabling a drag and drop experience for our customers. The VS2010 IDE also offers tight integration for the development and deployment of ASP.NET MVC applications. We offer a wide set of components for the ASP.NET MVC environment.

### Q What sets Syncfusion apart from other component vendors?

**A** As developers ourselves, each decision we make about how our products are built, delivered, and supported is based on our common viewpoint and we uncompromisingly strive for excellence in order to offer the very best value to our customers.

- **Built with the future in mind:** Syncfusion provides comprehensive support for the latest technologies, including .NET 4, WPF, Silverlight, MVC 2, as well as Windows Forms and SharePoint.
- **Full source code available:** You can use our components in debug mode at the flick of a switch, which is invaluable in improving overall productivity and application quality. With a Source license, you can step directly into our code from within the IDE, just as you would your own code.
- **First-rate support:** Online, forum-based, and 24x5 phone support at no extra cost ensure that you'll get the help you need, when you need it—even while you're evaluating our products with a free 30-day trial.
- **Straightforward, flexible licensing:** Our components are licensed on a simple per-developer basis with no royalties, run-time, or server-deployment fees. Each licensed developer may install on multiple personal machines and both Binary and Source licenses are available.

For more detailed information about Syncfusion controls and a free 30-day trial, visit [www.syncfusion.com](http://www.syncfusion.com).



For more information please visit  
[www.syncfusion.com](http://www.syncfusion.com)

**Syncfusion**  
Deliver innovation with ease





## Generating Graphs with WPF

Generating a graph from a set of test-related data is a common software-development task. In my experience, the most common approach is to import data into an Excel spreadsheet, then produce the graph manually using the Excel built-in graphing features. This works well in most situations, but if the underlying data changes frequently, creating graphs by hand can quickly become tedious. In this month's column, I'll show you how to automate the process using Windows Presentation Foundation (WPF) technology. To see where I'm heading, look at **Figure 1**. The graph shows a count of open versus closed bugs by date, and it was generated on the fly using a short WPF program that reads data from a simple text file.

The open bugs, represented by red circles on the blue line, increase rapidly near the beginning of the development effort, then trail off over time—information that might be useful when estimating a zero-bug bounce date. The closed bugs (the triangular markers on the green line) steadily increase.

But while the information may be useful, in production environments development resources are often limited, and manually generating such a graph might not be worth the effort. But using the technique I'll explain, creating graphs like this is quick and easy.

In the following sections, I'll present and describe in detail the C# code that generated the graph in **Figure 1**. This column assumes you have intermediate-level knowledge of C# coding and a very basic familiarity with WPF. But even if you're new to both, I think you'll be able to follow the discussion without too much difficulty. I'm confident you'll find the technique an interesting and useful addition to your skill set.

### Setting up the Project

I started by launching Visual Studio 2008 and creating a new C# project using the WPF Application template. I selected the .NET Framework 3.5 library from the drop-down control in the upper right-hand area of the New Project dialog box. I named my project BugGraph. Although you can programmatically generate graphs using WPF primitives, I used the convenient `DynamicDataDisplay` library developed by a Microsoft Research lab.



Figure 1 Programmatically Generated Bug-Count Graph

You can download the library for free from the CodePlex open source hosting site at [codeplex.com/dynamicdatadisply](http://codeplex.com/dynamicdatadisply). I saved my copy in the root directory of my BugGraph project, then added a reference to the DLL in my project by right-clicking on the project name, selecting the Add Reference option and pointing to the DLL file in my root directory.

Next, I created my source data. In a production environment, your data could be located in an Excel spreadsheet, a SQL database or an XML file. For simplicity, I used a simple text file. In the Visual Studio Solution Explorer window, I right-clicked on my project name and selected Add | New Item from the context menu. I then chose the Text File item, renamed the file to BugInfo.txt and clicked the Add button. Here's the dummy data:

```
01/15/2010:0:0
02/15/2010:12:5
03/15/2010:60:10
04/15/2010:88:20
05/15/2010:75:50
06/15/2010:50:70
07/15/2010:40:85
08/15/2010:25:95
09/15/2010:18:98
10/15/2010:10:99
```

The first colon-delimited field in each line holds a date, the second contains the number of open bugs on the associated date and the

Code download available at [code.msdn.microsoft.com/mag201006TestRun](http://code.msdn.microsoft.com/mag201006TestRun).

# Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at [www.alexcorp.com](http://www.alexcorp.com)

Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

## Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD



### New in Team 2.11

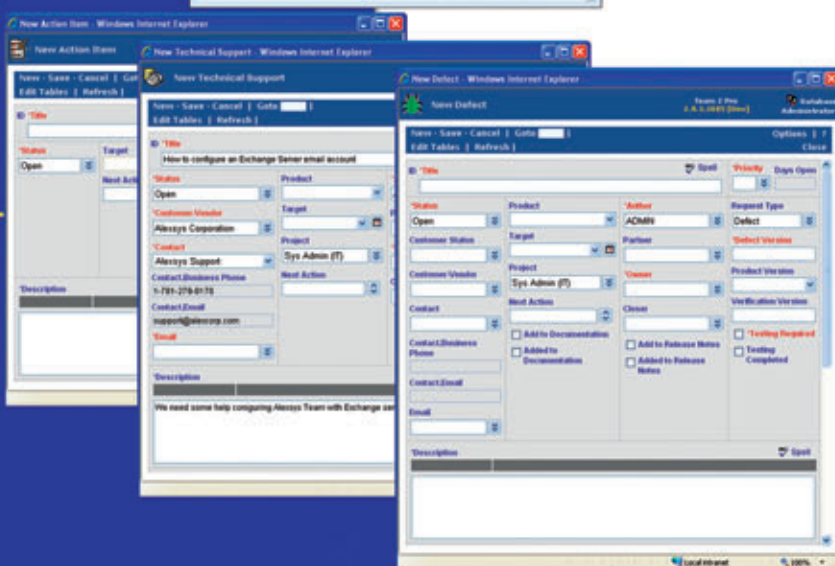
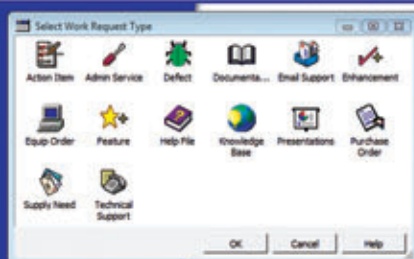
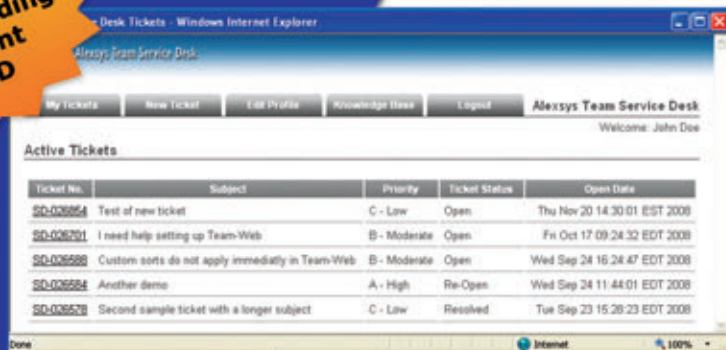
- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment

### Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

### Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration



Free Trial and Single User FreePack™ available at [www.alexcorp.com](http://www.alexcorp.com). FreePack™ includes a free single user Team Pro and Team-Web license. Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers. Team 2 works with Windows 7/2008/2003/Vista/XP. Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.



## Figure 2 Adding the Key Plotting Object

```
<d3:ChartPlotter Name="plotter" Margin="10,10,20,10">
  <d3:ChartPlotter.HorizontalAxis>
    <d3:HorizontalDateTimeAxis Name="dateAxis"/>
  </d3:ChartPlotter.HorizontalAxis>
  <d3:ChartPlotter.VerticalAxis>
    <d3:VerticalIntegerAxis Name="countAxis"/>
  </d3:ChartPlotter.VerticalAxis>

  <d3:Header FontFamily="Arial" Content="Bug Information"/>
  <d3:VerticalAxisTitle FontFamily="Arial" Content="Count"/>
  <d3:HorizontalAxisTitle FontFamily="Arial" Content="Date"/>
</d3:ChartPlotter>
```

third field shows the number of closed bugs. As you'll see shortly, the `DynamicDataDisplay` library can deal with most types of data.

Next I double-clicked on the file `Window1.xaml` to load the UI definitions for the project. I added a reference to the graphing library DLL and slightly modified the default `Height`, `Width` and `Background` attributes of the WPF display area, as follows:

```
xmlns:d3="http://research.microsoft.com/DynamicDataDisplay/1.0"
Title="Window1" WindowState="Normal" Height="500" Width="800"
Background="Wheat">
```

After that, I added the key plotting object, shown in **Figure 2**.

The ChartPlotter element is the main display object. In the definition for it, I added declarations for a horizontal date axis and a vertical integer axis. The default axis type for the DynamicDataDisplay library is a number with a decimal, that is type double in C# terms; no explicit axis declaration is necessary for that type. I also added a header title declaration and axis title declarations. **Figure 3** shows my design so far.

## Going to the Source

Once I'd configured the static aspects of my project, I was ready to add the code that would read the source data and programmatically gener-

ate my graph. I double-clicked on Window1.xaml.cs in the Solution Explorer window to load the C# file into the code editor. **Figure 4** lists the entire source code for the program that generated the graph in **Figure 1**.

I deleted the unnecessary using namespace statements (such as `System.Windows.Shapes`), which were generated by the Visual Studio template. Then I added using statements to three namespaces from the `DynamicDataDisplay` library so I wouldn't have to fully qualify their names. Next, in the `Window1` constructor I added an event for the main program-defined routine:

```
Loaded += new RoutedEventHandler(Window1_Loaded);
```

Here's how I began the main routine:

```
private void Window1_Loaded(object sender, RoutedEventArgs e)
{
    List<BugInfo> bugInfoList = LoadBugInfo("..\\..\\BugInfo.txt");
    ...
}
```

When working with the DynamicDataDisplay library, organizing the display data into a set of one-dimensional arrays is often convenient.

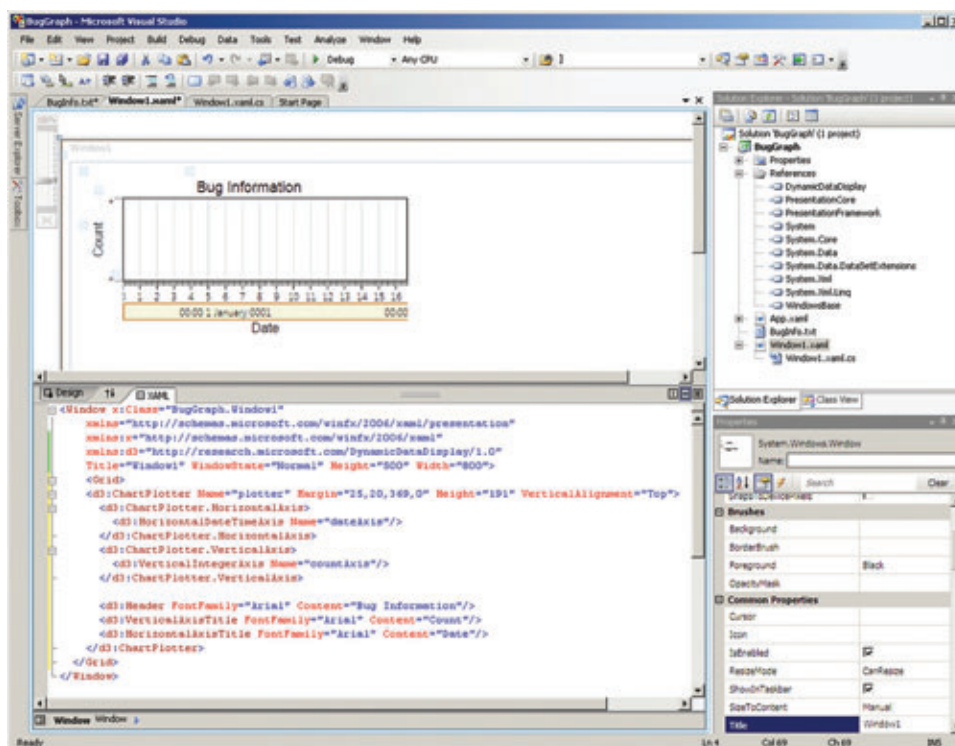
I declared a generic list object, `bugInfoList`, and populated the list with the dummy data in the file `BugInfo.txt` by using a program-defined helper method named `LoadBugInfo`. To organize my bug information, I declared a tiny helper class—`BugInfo`—as

Figure 5 shows.

I declared the three data fields as type public for simplicity, rather than as type private combined with get and set Properties. Because BugInfo is just data, I could've used a C# struct instead of a class. The LoadBugInfo method opens the BugInfo.txt file and iterates through it, parsing each field, then instantiates a BugInfo object and stores each BugInfo object into a result List, as shown in **Figure 6**.

Rather than reading and processing each line of the data file, I could have read all the lines into a string array using the `File.ReadAllLines` method. Notice that both to keep the size of my code small and for clarity, I omitted the normal error-checking you'd perform in a production environment.

Next I declared and assigned values to three arrays, as you can see in **Figure 7**.



### Figure 3 BugGraph Program Design

Figure 4 Source Code for the BugGraph Project

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Media; // Pen

using System.IO;
using Microsoft.Research.DynamicDataDisplay; // Core functionality
using Microsoft.Research.DynamicDataDisplay.DataSources; //
EnumerableDataSource
using Microsoft.Research.DynamicDataDisplay.PointMarkers; //
CirclePointMarker

namespace BugGraph
{
    public partial class Window1 : Window
    {
        public Window1()
        {
            InitializeComponent();
            Loaded += new RoutedEventHandler(Window1_Loaded);
        }

        private void Window1_Loaded(object sender, RoutedEventArgs e)
        {
            List<BugInfo> bugInfoList = LoadBugInfo(".....\\BugInfo.txt");

            DateTime[] dates = new DateTime[bugInfoList.Count];
            int[] numberOpen = new int[bugInfoList.Count];
            int[] numberClosed = new int[bugInfoList.Count];

            for (int i = 0; i < bugInfoList.Count; ++i)
            {
                dates[i] = bugInfoList[i].date;
                numberOpen[i] = bugInfoList[i].numberOpen;
                numberClosed[i] = bugInfoList[i].numberClosed;
            }

            var datesDataSource = new EnumerableDataSource<DateTime>(dates);
            datesDataSource.SetXMapping(x => dateAxis.ConvertToDouble(x));

            var numberOpenDataSource = new EnumerableDataSource<int>(numberOpen);
            numberOpenDataSource.SetYMapping(y => y);

            var numberClosedDataSource = new EnumerableDataSource<int>(numberClosed);
            numberClosedDataSource.SetYMapping(y => y);

            CompositeDataSource compositeDataSource1 = new
            CompositeDataSource(datesDataSource, numberOpenDataSource);
            CompositeDataSource compositeDataSource2 = new
            CompositeDataSource(datesDataSource, numberClosedDataSource);

            plotter.AddLineGraph(compositeDataSource1,
                new Pen(Brushes.Blue, 2),
                new CirclePointMarker { Size = 10.0, Fill = Brushes.Red },
                new PenDescription("Number bugs open"));

            plotter.AddLineGraph(compositeDataSource2,
                new Pen(Brushes.Green, 2),
                new TrianglePointMarker { Size = 10.0,
                    Pen = new Pen(Brushes.Black, 2.0),
                    Fill = Brushes.GreenYellow },
                new PenDescription("Number bugs closed"));

            plotter.Viewport.FitToView();
        } // Window1_Loaded()

        private static List<BugInfo> LoadBugInfo(string fileName)
        {
            var result = new List<BugInfo>();
            FileStream fs = new FileStream(fileName, FileMode.Open);
            StreamReader sr = new StreamReader(fs);

            string line = "";
            while ((line = sr.ReadLine()) != null)
            {
                string[] pieces = line.Split(':');
                DateTime d = DateTime.Parse(pieces[0]);
                int numopen = int.Parse(pieces[1]);
                int numclosed = int.Parse(pieces[2]);
                BugInfo bi = new BugInfo(d, numopen, numclosed);
                result.Add(bi);
            }
            sr.Close();
            fs.Close();
            return result;
        }

    } // class Window1

    public class BugInfo {
        public DateTime date;
        public int numberOpen;
        public int numberClosed;

        public BugInfo(DateTime date, int numberOpen, int numberClosed) {
            this.date = date;
            this.numberOpen = numberOpen;
            this.numberClosed = numberClosed;
        }
    }

} // ns
```

When working with the `DynamicDataDisplay` library, organizing the display data into a set of one-dimensional arrays is often convenient. As an alternative to my program design, which read data into a list object and then transferred the list data into arrays, I could have read data directly into arrays.

Next I converted my data arrays into special `EnumerableDataSource` types:

```
var datesDataSource = new EnumerableDataSource<DateTime>(dates);
datesDataSource.SetXMapping(x => dateAxis.ConvertToDouble(x));

var numberOpenDataSource = new EnumerableDataSource<int>(numberOpen);
numberOpenDataSource.SetYMapping(y => y);

var numberClosedDataSource = new EnumerableDataSource<int>(numberClosed);
numberClosedDataSource.SetYMapping(y => y);
...
```

For the `DynamicDataDisplay` library, all data to be graphed must be in a uniform format. I just passed the three arrays of data to the generic `EnumerableDataSource` constructor. Additionally, the library must be told which axis, *x* or *y*, is associated with each data source. The `SetXMapping` and `SetYMapping` methods

accept method delegates as arguments. Rather than define explicit delegates, I used lambda expressions to create anonymous methods. The `DynamicDataDisplay` library's fundamental-axis data type is double. The `SetXMapping` and `SetYMapping` methods map my particular data type to type double.

On the *x*-axis, I used the `ConvertToDouble` method to explicitly convert `DateTime` data into type double. On the *y*-axis, I simply wrote `y => y` (read as “*y* goes to *y*”) to implicitly convert the input `int y` to the output double `y`. I could have been explicit with my type mapping by writing `SetYMapping(y => Convert.ToDouble(y))`. My choices of *x* and *y* for the lambda expressions’ parameters were arbitrary—I could have used any parameter names.

The next step was to combine the *x*-axis and *y*-axis data sources:

```
CompositeDataSource compositeDataSource1 = new
    CompositeDataSource(datesDataSource, numberOpenDataSource);

CompositeDataSource compositeDataSource2 = new
    CompositeDataSource(datesDataSource, numberClosedDataSource);
...
```



Figure 5 The Helper Class BugInfo

```
public class BugInfo {
    public DateTime date;
    public int numberOpen;
    public int numberClosed;

    public BugInfo(DateTime date, int numberOpen, int numberClosed) {
        this.date = date;
        this.numberOpen = numberOpen;
        this.numberClosed = numberClosed;
    }
}
```

The screenshot in **Figure 1** shows two data series—the number of open bugs and the number of closed bugs—plotted on the same graph. Each composite data source defines a data series, so here I needed two individual data sources—one for the number of open bugs and one for the number of closed bugs. With the data all prepared, a single statement actually plotted the data points:

```
plotter.AddLineGraph(compositeDataSource1,
    new Pen(Brushes.Blue, 2),
    new CirclePointMarker { Size = 10.0, Fill = Brushes.Red },
    new PenDescription("Number bugs open"));
```

...

The `AddLineGraph` method accepts a `CompositeDataSource`, which defines the data to be plotted, along with information about exactly how to plot it. Here I instructed the plotter object named *plotter* (defined in the `Window1.xaml` file) to do the following: draw a graph using a blue line of thickness 2, place circular markers of size 10 that have red borders and red fill, and add the series title *Number bugs open*. Neat! As one of many alternatives, I could have used

```
plotter.AddLineGraph(compositeDataSource1, Colors.Red, 1, "Number Open")
```

The `AddLineGraph` method accepts a `CompositeDataSource`, which defines the data to be plotted, along with information about exactly how to plot it.

to draw a thin red line with no markers. Or I could have created a dashed line instead of a solid line:

```
Pen dashedPen = new Pen(Brushes.Magenta, 3);
dashedPen.DashStyle = DashStyles.DashDot;
plotter.AddLineGraph(compositeDataSource1, dashedPen,
    new PenDescription("Open bugs"));
```

My program finished by plotting the second data series:

```
...
plotter.AddLineGraph(compositeDataSource2,
    new Pen(Brushes.Green, 2),
    new TrianglePointMarker { Size = 10.0,
        Pen = new Pen(Brushes.Black, 2.0),
        Fill = Brushes.GreenYellow },
    new PenDescription("Number bugs closed"));

plotter.Viewport.FitToView();

} // Window1_Loaded()
```

Figure 6 The `LoadBugInfo` Method

```
private static List<BugInfo> LoadBugInfo(string fileName)
{
    var result = new List<BugInfo>();
    FileStream fs = new FileStream(fileName, FileMode.Open);
    StreamReader sr = new StreamReader(fs);

    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
        string[] pieces = line.Split(':');
        DateTime d = DateTime.Parse(pieces[0]);
        int numopen = int.Parse(pieces[1]);
        int numclosed = int.Parse(pieces[2]);
        BugInfo bi = new BugInfo(d, numopen, numclosed);
        result.Add(bi);
    }
    sr.Close();
    fs.Close();
    return result;
}
```

Here I instructed the plotter to use a green line with triangular markers that have a black border and a green-yellow fill. The `FitToView` method scales the graph to the size of the WPF window.

After instructing Visual Studio to build the `BugGraph` project, I got a `BugGraph.exe` executable, which can be launched manually or programmatically at any time. I can update the underlying data by simply editing the `BugInfo.txt` file. Because the entire system is based on .NET Framework code, I can easily integrate graphing capability into any WPF project without having to deal with cross-technology issues. And there's a Silverlight version of the `DynamicDataDisplay` library so I can add programmatic graphing to Web applications, too.

## A Scatter Plot

The technique I presented in the previous section can be applied to any kind of data, not just test-related data. Let's take a brief look at another simple but rather impressive example. The screenshot in **Figure 8** shows 13,509 U.S. cities. You can probably identify where Florida, Texas, Southern California and the Great Lakes are. I obtained the data for the scatter plot from a library of data intended for use with the traveling salesman problem ([iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95](http://iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95)), one of the most famous and widely studied topics in computer science. The file I used, `usa13509.tsp.gz`, looks like:

```
NAME : usa13509
(other header information)
1 245552.778 817827.778
2 247133.333 810905.556
3 247205.556 810188.889
...

13507 489663.889 972433.333
13508 489938.889 1227458.333
13509 490000.000 1222636.111
```

Figure 7 Building Arrays

```
DateTime[] dates = new DateTime[bugInfoList.Count];
int[] numberOpen = new int[bugInfoList.Count];
int[] numberClosed = new int[bugInfoList.Count];

for (int i = 0; i < bugInfoList.Count; ++i)
{
    dates[i] = bugInfoList[i].date;
    numberOpen[i] = bugInfoList[i].numberOpen;
    numberClosed[i] = bugInfoList[i].numberClosed;
}

...
```

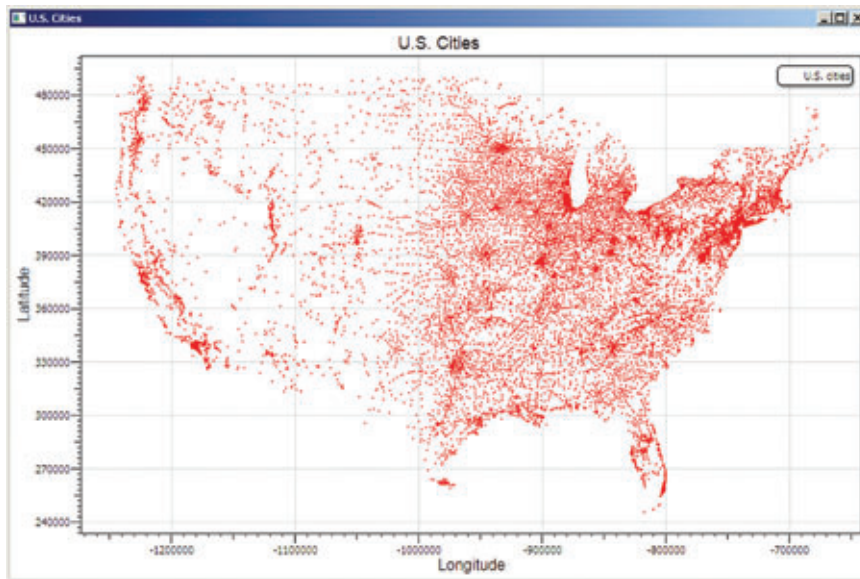


Figure 8 Scatter Plot Example

The first field is a 1-based index ID. The second and third fields represent coordinates derived from the latitude and longitude of U.S. cities with populations of 500 or greater. I created a new WPF application as described in the previous section, added a text-file item to the project and copied the city data to the file. I commented out the header lines of the data file by prepending double-slash (//) characters to those lines.

To create the scatter plot shown in **Figure 8**, I only needed to make minor changes to the example presented in the previous section. I modified the MapInfo class members as follows:

```
public int id;
public double lat;
public double lon;
```

**Figure 9** shows the key processing loop in the revised Load-MapInfo method.

I had the code check to see if the current line begins with my program-defined comment tokens, and if so, skip over it. Notice that I multiplied the longitude-derived field by -1.0 because longitudes go from east to west (or right to left) along the *x*-axis. Without the -1.0 factor, my map would be a mirror image of the correct orientation.

When I populated my raw data arrays, all I had to do was ensure I correctly associated latitude and longitude to the *y*-axis and the *x*-axis, respectively:

Figure 9 Loop for Scatter Plot

```
while ((line = sr.ReadLine()) != null)
{
    if (line.StartsWith("//"))
        continue;
    else {
        string[] pieces = line.Split(' ');
        int id = int.Parse(pieces[0]);
        double lat = double.Parse(pieces[1]);
        double lon = -1.0 * double.Parse(pieces[2]);
        MapInfo mi = new MapInfo(id, lat, lon);
        result.Add(mi);
    }
}
```

```
for (int i = 0; i < mapInfoList.Count; ++i)
{
    ids[i] = mapInfoList[i].id;
    xs[i] = mapInfoList[i].lon;
    ys[i] = mapInfoList[i].lat;
}
```

If I had reversed the order of the associations, the resulting map would have been tilted on its edge. When I plotted my data, I needed only one small tweak to make a scatter plot instead of a line graph:

```
plotter.AddLineGraph(compositeDataSource,
    new Pen(Brushes.White, 0),
    new CirclePointMarker { Size = 2.0, Fill = Brushes.Red },
    new PenDescription("U.S. cities"));
```

By passing a 0 value to the Pen constructor, I specified a 0-width line, which effectively removed the line and created a scatter plot rather than a line graph. The resulting graph is pretty cool, and the program that generated the graph took only a few minutes to write. Believe me, I've tried many other approaches to plotting geographic data, and

using WPF with the DynamicDataDisplay library is among the best solutions I've found.

I've tried many other approaches  
to plotting geographic data,  
and using WPF with the  
DynamicDataDisplay library  
is one of the best  
solutions I've found.

## Graphing Made Easy

The techniques I've presented here can be used to programmatically generate graphs. The key to the technique is the DynamicDataDisplay library from Microsoft Research. When used as a standalone technique to generate graphs in a software production environment, the approach is most useful if the underlying data changes frequently. When used in an application as an integrated technique to generate graphs, the approach is most useful with WPF or Silverlight applications. And as those two technologies evolve, I'm sure we'll see more great visual-display libraries based on them. ■

**DR. JAMES McCaffrey** works for Volt Information Sciences Inc. where he manages technical training for software engineers working at the Microsoft Redmond, Wash., campus. He has worked on several Microsoft products, including Internet Explorer and MSN Search. McCaffrey is the author of ".NET Test Automation Recipes: A Problem-Solution Approach" (Apress, 2006). He can be reached at [jammc@microsoft.com](mailto:jammc@microsoft.com).

**THANKS** to the following technical experts for reviewing this article:  
Paul Newson, Paul Koch and Anne Loomis; all of Microsoft Research



# Multi-Targeting Visual Basic Applications in Visual Studio 2010

Prior to Visual Studio 2008, writing applications that targeted different versions of the Microsoft .NET Framework required installation of different versions of the Visual Studio development environment. Each installation of Visual Studio offered a different developer experience and used significant disk space. Moreover, the project file format changed between each version of Visual Studio. As a result, you could end up with multiple versions of a project or solution while developing a component for use in projects targeting different .NET Framework versions.

Visual Studio 2008 was the first version to fully support multi-targeting within a single IDE, which allowed developers to write applications targeting different versions of the .NET Framework (2.0, 3.0 and 3.5) using a single Visual Studio installation. The result? A single, consistent developer experience with reduced disk-space requirements.

Multi-targeting in Visual Studio 2008 worked because each of the available frameworks used the same underlying CLR 2.0. Furthermore, each version of the framework built upon the .NET Framework 2.0 foundation, providing additional functionality through the use of referenced assemblies. Ultimately, all used the .NET Framework 3.5 command-line Visual Basic compiler (vbc.exe).

In this article I discuss the 3.5 and 4 compilers, referring to the compilers installed as part of the respective .NET Framework 3.5 and 4 installations. The 3.5 compiler is the version shipped with Visual Studio 2008 and Visual Basic 9, while the 4 compiler is the version shipped with Visual Studio 2010 and Visual Basic 10.

So let's take a look at how multi-targeting works in Visual Studio today, and how you should approach multi-targeting in your projects.

## Multi-Targeting in Visual Studio

In Visual Studio 2008, changing the desired target framework was as simple as selecting the target from a drop-down list in the project properties, as shown in **Figure 1**. This added or removed specific references required for each framework version and made changing frameworks painless.

For command-line compilation it was simply a matter of changing the reference assemblies used.

Some big changes came with Visual Studio 2010, however. The new .NET Framework 4 brings a new version of the CLR. This means the approach taken in Visual Studio 2008 is not practical in Visual Studio 2010. As a result, Visual Studio 2010 uses the version 4 compiler for all multi-targeting, even when targeting previous .NET Framework versions. This allows many of the newer language

features to be used for down-targeting and provides a much-simplified development experience.

However, one downside of being able to use Visual Studio 2010 features for down-level targets is that source files may not be design-time compatible if used with earlier versions of Visual Studio. This may be an issue if you're sharing source code for projects built using different versions of Visual Studio and targeting different .NET Framework versions.

Visual Studio 2010 uses  
the version 4 compiler for  
all multi-targeting.

If you keep the projects within Visual Studio 2010 for all design-time work, you'll have a better experience. You'll be able to generate assemblies targeting the .NET Framework 2.0 upward using only Visual Studio 2010 and the .NET Framework 3.5 SP1.

## Design-Time Compatibility

Now let's take a look at an example of a design-time compatibility gotcha. The code in **Figure 2** uses both implicit line continuation and the auto-implemented property feature, both introduced in Visual Studio 2010. This code can be compiled to target any framework from 2.0 onward when compiled using Visual Studio 2010. So the generated assembly is runtime-compatible.

However, take this same source code file and try compiling using either the 3.5 or 2.0 versions of the compiler—you'll generate the errors shown in **Figure 3**.

This occurs because the earlier versions of the compiler know nothing about these features and treat this as invalid code. So the source files are not design-time compatible. To make this design-time compatible, you'd have to use only features available in the 3.5 compiler.

Design-time compatibility has implications for Web projects as well. For many Web projects, compilation takes place on the server, and the server, of course, compiles the pages using the target framework compiler installed on the server. So if you have a Web page written in Visual Basic targeting the 3.5 compiler, the page would be compiled on the server using the 3.5 version of the Visual Basic Compiler (vbc.exe). Any use of new Visual Studio 2010 language features would fail as the 3.5 compiler knows nothing about them.

# VSLive!

Presents:

## SharePoint Live

**August 3-6, 2010**  
Redmond, WA  
Microsoft Campus

**4 FULL DAYS OF EXCLUSIVE  
SHAREPOINT SESSIONS!**  
Maximize the Development Capability of SharePoint 2010!

### CONFERENCE DAY 1—TUESDAY, AUGUST 3

9:45–11:00 AM	Whats New in the SharePoint 2010 Development Platform
11:15 AM–12:30 PM	Introduction to SharePoint Development with Visual Studio 2010
1:45–3:00 PM	Silverlight Development with SharePoint 2010
3:45–5:00 PM	Feature Upgrade Enhancements in SharePoint 2010

### CONFERENCE DAY 2—WEDNESDAY, AUGUST 4

9:45–11:00 AM	Creating SharePoint 2010 Workflows with SharePoint Designer 2010
11:15–12:30 PM	Creating Workflow Templates with Visual Studio 2010
1:30–2:45 PM	Integrating SharePoint 2010 and Azure: Evolving Towards the Cloud
3:00–4:15 PM	Client-Side Development for SharePoint 2010 Using JavaScript and jQuery
5:00–6:15 PM	PowerPivot and Excel Services

### CONFERENCE DAY 3—THURSDAY, AUGUST 5

8:30–9:45 AM	SharePoint 2010 Support for Social Computing and Communities	
10:00–11:15 AM	Developing Social Computing Solutions	SharePoint 2010 WCM for Developers with Visual Studio 2010
11:30 AM–12:45 PM	SharePoint Developers Deep Dive into Claim-Based Security	Understanding How the Sandbox Works
1:45–3:00 PM	Creating No-Code SharePoint Applications with SharePoint Designer	Best Practices for Upgrading Web Parts
3:15–4:30 PM	Creating Extensions for the Visual Studio 2010 SharePoint Tools	Application Lifecycle Management for Developers in SharePoint 2010

### POST-CONFERENCE WORKSHOP—FRIDAY, AUGUST 6 (SEPERATE ENTRY FEE REQUIRED)

9:00 AM–5:00 PM	SharePoint 2010 for ASP.NET Developers
-----------------	--



REGISTER BEFORE JUNE 30TH AT

**VSLIVE.COM/REDMOND**

**AND SAVE \$200!**  
USE PROMO CODE: **SPL1**

Platinum Sponsors:

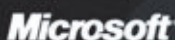


VERSANT

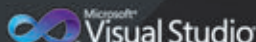
Gold Sponsor:



Supported by:



Visual Studio  
MAGAZINE





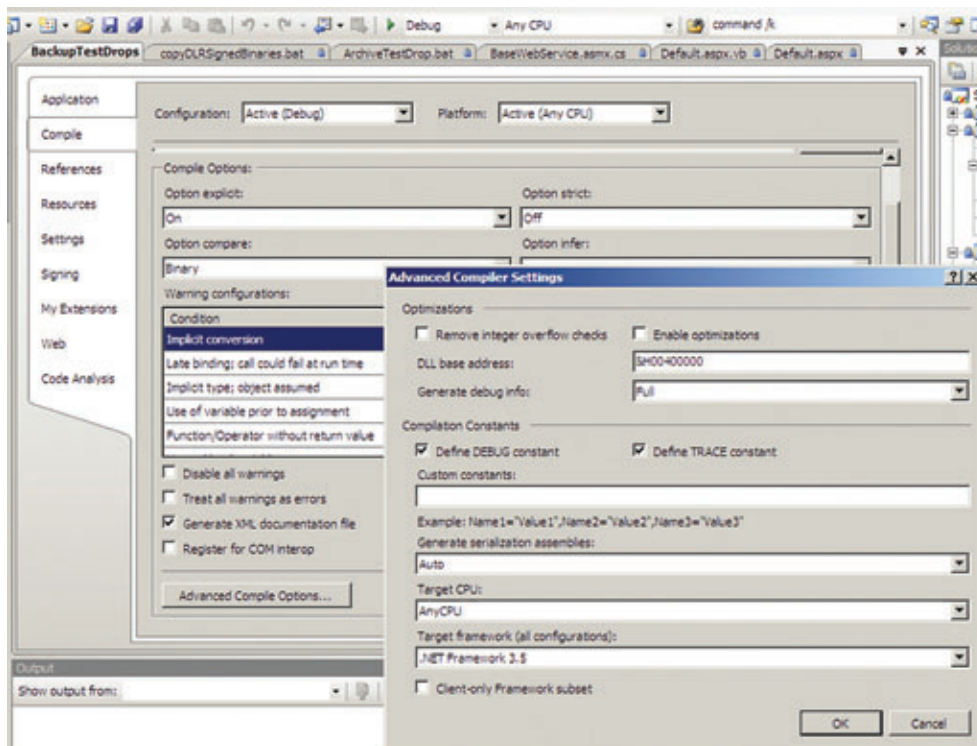


Figure 1 Changing the Desired Target Framework in Visual Studio 2008

For code developed in Visual Studio 2010, which uses the version 4 compiler, you need a way to identify this requirement at compile time to prevent unexpected errors when the Web page is deployed to the server. You can do this with the `/langversion` switch, which is used when developing Web projects to generate errors about newer language syntax features that won't compile on an earlier framework's compiler. When building ASP.NET project types, this switch is used internally to generate errors if your code uses new Visual Studio 2010 features but you are targeting earlier versions of the framework.

Although the `/langversion` switch is not used by default for any of the other project types, it can be useful if you want to verify that your source code is design-time compatible with previous versions of Visual Studio.

## Multi-Targeting in the Visual Studio 2010 IDE

The multi-targeting user experience in the Visual Studio 2010 IDE is almost identical to that of Visual Studio 2008. It is still controlled from within the project properties, but in a default installation you may not see the earlier target frameworks. To cut down on install size, the Visual Studio team decided not to ship the 3.5 framework in the default installation of Visual Studio 2010. This change means that you would not see these framework options appearing in the Target

framework drop-down or New Project dialog box.

To add these additional frameworks, you need to install the .NET Framework 3.5 SP1. You can do this right from the IDE. At the top of the New Project dialog box, you'll see a drop-down menu for choosing the target framework. If only the .NET Framework 4 is installed, the menu contains a link to download more. If you install any others, however, you'll see only the .NET Framework 3.5 SP1 on the drop-down menu because Visual Studio only recognizes installation of the .NET Framework 3.5 SP1 here.

Another change has to do with client profiles. These were introduced in the .NET Framework 3.5 SP1 and they let applications use a lightweight version of the framework, which can improve

deployments by not requiring the inclusion of server-side pieces of the framework, like ASP.NET. These profiles are available for both 3.5 and 4 framework targets.

As a result, the default profile for various project types has changed. The client project types—Windows, Console, Office and Windows Presentation Foundation (WPF) applications—will default to the client profile. However, for Web applications the default profile will be “full” because of references to libraries that are not deployed with the client profile, such as System.Web.

Class libraries also default to the full profile, but can be easily changed back to a client profile if you're depending only on references deployed with the client profile. If your class library is set to full profile and is then used in a project with a client profile, it will still work as long as the library doesn't depend upon references that are not part of the client framework assemblies.

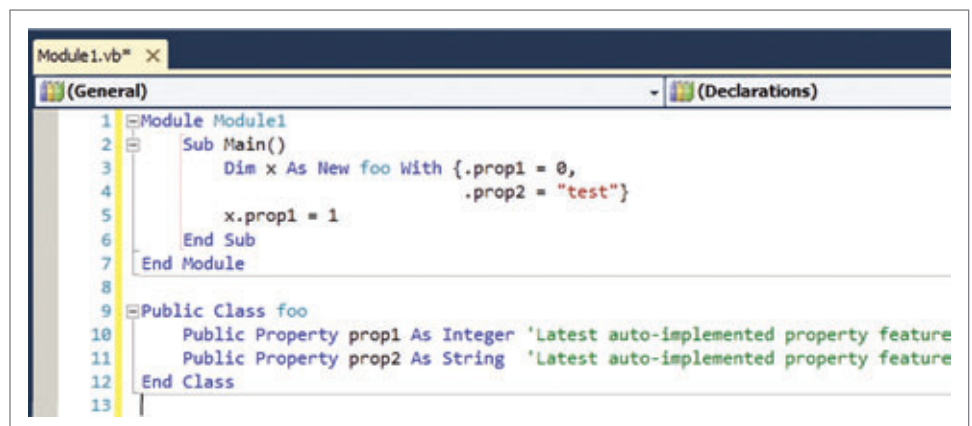


Figure 2 Using New Language Features That Will Work in Down-Level Targets

By default, none of the references added to the class library project type require a full profile. However, because they're deployed with an application, the application deployment profile is the important setting to ensure full application functionality. If your library depends on references outside of the client scope, both the library and the application using it need to employ the full profile.

## Multi-Targeting Using the Command-Line Compiler

The version 4 compiler has a number of command-line switches, none of which, unfortunately, controls the target framework so it's important to understand a little about each of the switches and how they work.

If the .NET Framework 4 is installed, it is possible to build applications using `vbc.exe` that target earlier versions of the framework without having Visual Studio installed on the build machine. Build scripts that call the command-line compiler directly are often used in larger development environments. If you're targeting earlier versions from the command line, this requires files installed with the previous framework version you're targeting, so the best plan is to have both .NET Framework 3.5 SP1 and .NET Framework 4 installed on the machine.

With this in mind, some of the potential switches for multi-targeting are detailed in **Figure 4**.

This table provides a quick description of each switch, but in order to actually create a down-targeted compilation, you'll need to use a combination—there's no single multi-target switch. For a version 3.5 target, the most important switch is `sdopath`, which you

can use to specify the 2.0 version of `mscorlib`. Then ensure your references point to the correct versions of `System.dll`, `System.core.dll` and any other prior target framework assemblies. (These can be found in the `%programfiles%\Reference Assemblies\Microsoft\Framework\v3.5` folder.)

The idea of design-time compatibility leads to an interesting gotcha with Web projects.

You need to specify the `noconfig` switch to avoid using the default switches in the version 4 of `vbc.rsp`, which contains the default settings for compilation. Without adding this critical switch, the compiler would add those default 4 references, imports and so on.

Multi-targeted command-line compilation is best demonstrated by an example. Here I'm simply compiling a simple source file, `test.vb`, to target the .NET Framework 3.5:

```
vbc.exe /noconfig /sdopath:D:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
/r:"D:\Program Files\Reference
Assemblies\Microsoft\Framework\v3.5\System.Core.dll" d:\school\test.vb
/out:\school\test.exe
```

When you understand the switches to compile a 3.5 assembly, targeting 2.0 instead simply involves removing some of the references,

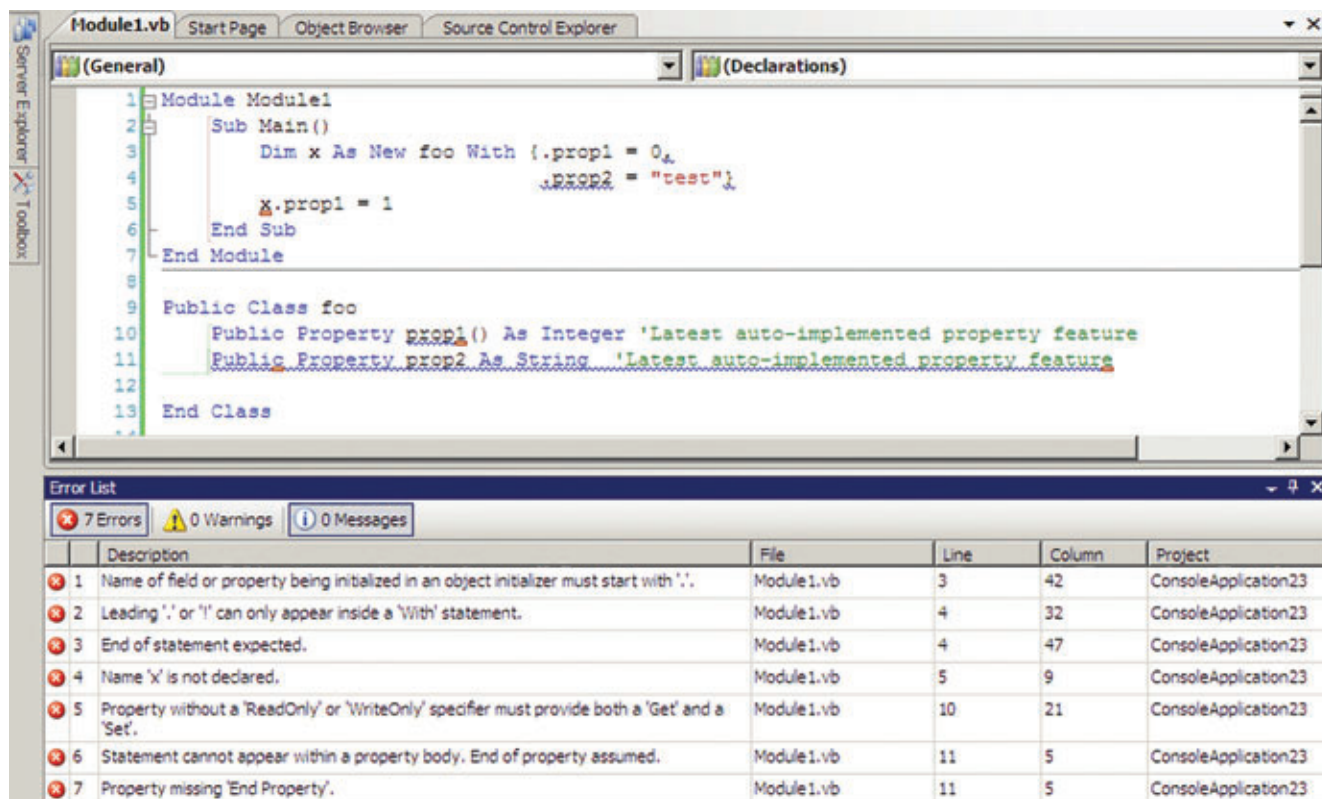


Figure 3 Source Code from Visual Studio 2010 Is Not Design-Time Compatible with Visual Studio 2008



Figure 4 Command-Line Build Switches to Control Multi-Targeting

Switch	Description
langversion	Provides errors for source code using features that don't meet the specific language version. (9.0 relates to targets up to the .NET Framework 3.5; 10 relates to .NET Framework 4 targets.) This does not actually determine the target framework or CLR being used, but it allows Web projects to identify Visual Studio 2010 features used in down-target scenarios.
vbruntime	Although there's a different version of Microsoft.VisualBasic for the .NET Framework 4, simply trying to specify the 2.0 version of Microsoft.VisualBasic.dll doesn't work and results in an assembly that's dependent on the version 4 NetFX.
nostdlib	Prevents the standard reference to system.dll from being added to the assembly. Although it is possible to use this option along with a reference to the version of system.dll in the 2.0 framework, the result is still a version 4 assembly.
sdkpath	A key option that specifies which version of MSCorLib.dll and Microsoft.VisualBasic.dll to use if the vbruntime switch does not specify which will be used. However, this is not an explicit reference you'll typically see in the list of references. Instead, the compiler includes this in its standard references. Adding it is part of the solution for multi-targeting when you want the version 2.0 MSCorLib, not the version 4.
noconfig	Causes the compiler to avoid adding the default references, imports and switches contained in the vbc.rsp file, which would otherwise be used.

such as system.core.dll, that are required for the 3.5 framework. The sdkpath and noconfig switches remain the same:

```
vbc.exe /noconfig /sdkpath:D:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
d:\school\test.vb /out:\school\test.exe
```

Once you have the compiled binary, you can employ a tool such as the MSIL Disassembler (Ildasm.exe) or .NET Reflector to look at the versions of the references being used. This lets you determine whether an executable will run on a desired target framework.

Client-Profile and Mixed-Target Solutions

Earlier in the article I mentioned that client profiles were the default for most project types. When such applications are deployed, a smaller-footprint framework installation occurs. As part of this deployment you can observe that the command-line compiler is deployed. This is the same version of the compiler that's deployed with the full framework, but there's a potential gotcha when trying to compile a simple application on a client profile.

Using this command on a client framework machine would fail because the client frameworks do not ship with a vbc.rsp that contains the default references:

```
vbc.exe test.vb /out: test.exe
```

You'd have to either specify all the references and imports statements that would normally be contained in the vbc.rsp file or create your own.

The bare minimum switches required to compile a Visual Basic application on a client framework installation are:

```
/r:System.dll
/imports:System
/imports:Microsoft.VisualBasic
```

By including these switches you can compile a basic Visual Basic application. However, you should compile applications on a machine that has the full framework installed.

Mixed-target solutions—class libraries built with the .NET Framework 3.5 used with a client application targeting the .NET framework 4—are supported, but with some caveats. Within the Visual Studio 2010 IDE, if you're using project references and are accustomed to the experience they provide,

you can still get this experience if the target frameworks in the project references use the same version of MSCorlib. **Figure 5** shows the MSCorlib versions and supported framework versions.

So if you're using a class library targeting MSCorlib 2.0 on the .NET Framework 3.5 application, you are still able to use project references. Similarly, a .NET Framework 4 full-profile class library referenced by a .NET Framework 4 client-profile Windows application can have a project reference to the library.

However, if you use a project-to-project reference where a different version of MSCorlib is used, the project reference will be converted into a file reference. This means you'll need to manually rebuild the solution when you correct errors. The experience will be familiar if you've worked with solutions that have multiple referenced projects written in both C# and Visual Basic. You lose some of the convenient features available with project references, such as renaming across projects and automatic background compilation.

The IDE shields you somewhat from what happens behind the scenes during compilation.

The IDE shields you somewhat from what happens behind the scenes during compilation, but not everyone builds from the IDE. The file reference will automatically change back to a project reference if the target frameworks are changed so that both use frameworks with a common version of MSCorlib—so it's not a true file reference.

What happens if you use a down-target (3.5) class library within a version 4 application? The class library will actually run against the .NET Framework 4. Considerable testing has occurred to ensure that this scenario works at run time with few problems. However, trying to use a 4.0 framework class library on a 3.5 framework application is not supported and will result in a compile-time error if building with either Visual Studio or MSBuild. To use 4.0 framework class libraries in a 3.5 framework-targeted application, however, you would need to down-target the class library to the .NET Framework 3.5.

Keep in mind, though, that with the ability to use the Visual Studio 2010 language features on down-target scenarios, targeting the

Figure 5 MSCorlib and Framework Version Compatibility

MSCorlib version	Supported Frameworks	Profiles
2.0	2.0, 3.0, 3.5	Client and Full Profiles
4.0	4	Client and Full Profiles

class library to the .NET Framework 3.5 should not be a big issue. **Figure 6** summarizes the new features you can expect to work in down-target projects.

## PIAs and Interop

Using the .NET Framework to program against the Microsoft Office object model requires the use of Primary Interop Assemblies (PIAs), which must be deployed to the user's machine. These assemblies are often very large and deploying them can be a nuisance.

The new type-embedding feature allows these applications to be deployed without requiring PIAs on the user's machine. It does this by generating embedded interop types that perform the interop calls to the COM library directly. These types are annotated by the compiler in such a way that the CLR treats all embedded interop type instances as equivalent. The compiler will not copy every type in the PIA into your assembly, just the ones you actually use. For more information, see "Type Equivalence and Embedded Interop Types" in the MSDN library ([msdn.microsoft.com/library/dd997297\(VS.100\)](http://msdn.microsoft.com/library/dd997297(VS.100))).

The functionality of this feature is not supported for down-target scenarios below the .NET Framework 4. Using the Visual Studio IDE, this would not be immediately obvious as setting the reference embedded interop property to true in a down-target scenario results in normal references being used. The user experience for the feature from the IDE is that the assembly will continue to build, but would revert back to the behavior of standard references, which would require PIAs to be deployed.

From the command line, references are normally added using the /reference switch. For embedding, the /link switch is used instead. Attempting to use the /link switch for down-target scenarios results in compile errors.

Here's an example of a command line embedding types from the Word interop assembly:

```
D:\Windows\Microsoft.NET\Framework\v4.0.30128\Vbc.exe /
imports:Microsoft.VisualBasic,System /link:"D:\Program Files\Microsoft
Visual Studio 10.0\Visual Studio Tools for Office\PIA\Office14\
Microsoft.Office.Interop.Word.dll" /reference:"D:\Program Files\
Reference Assemblies\Microsoft\Framework\NETFramework\v4.0\Profile\
Client\System.Core.dll","D:\Program Files\Reference Assemblies\
Microsoft\Framework\NETFramework\v4.0\Profile\Client\System.dll" /
out:ConsoleApplication16.exe /target:exe Module1.vb
```

This behavior is important because, by default, COM references added to a project in Visual Studio 2010 set the Embed Interop property to true. So changing the target framework should not result in additional errors, but should provide the benefit of embedded interop types where possible.

**Figure 6 New Visual Studio Features in Down-Target Scenarios**

Language Feature	Works in Down-Target Scenarios
Collection initializers	Yes
Array initializers	Yes
Auto-implemented properties	Yes
Implicit line continuation	Yes
Statement lambdas	Yes
No PIA	No
Dynamic interop	No
Co/contravariance	Partially

Another new feature in Visual Studio 2010 that is not supported for down-target scenarios is dynamic interop because, prior to Visual Studio 2010, the Dynamic Language Runtime (DLR) didn't exist.

## Other Issues

Covariance and contravariance are supported for use with user-defined interfaces. However, Base Class Library (BCL) interfaces for down-level targets are not changed and therefore using the feature with these base classes is not supported. For more information on covariance and contravariance, see the Basic Instincts column in the March 2010 issue of *MSDN Magazine* ([msdn.microsoft.com/magazine/ee336029](http://msdn.microsoft.com/magazine/ee336029)).

If you have a project or solution created in a previous version of Visual Studio that you open in Visual Studio 2010, you'll see the standard upgrade dialog box. Visual Studio will make the necessary changes to the project or solution files to work with Visual Studio 2010. However, there are two different actions involved in upgrading your files that could affect multi-targeting.

Covariance and contravariance are supported for use with user-defined interfaces.

If you have the .NET Framework 3.5 SP1 installed, the upgrade dialog box will allow the project or solution files to be upgraded to Visual Studio 2010, but the target frameworks specified for the project will remain untouched. So if you're upgrading a .NET Framework 3.5-targeted application, after upgrade it should still target the 3.5 framework.

If you do not have the .NET Framework 3.5 SP1 installed, you can't build multi-targets correctly because you need the 2.0 version of MSCorlib and the reference assemblies. The dialog will provide the option to change the target to a version 4 framework or to not upgrade the project. Your best course in this case is to cancel the upgrade, install the .NET Framework 3.5 SP1, then run through the process to upgrade the project again.

By understanding a little more of the implementation details of Visual Basic multi-targeting in Visual Studio 2010, you should be able to write code that produces assemblies that can be deployed on prior versions of the framework using the IDE or command line, but still take advantage of some of the new Visual Studio 2010 features. Although multi-targeting has some caveats, you retain the ability to develop and deploy applications that can't immediately be upgraded to use the .NET Framework 4. ■

**ADRIAN SPOTTY BOWLES** has developed using every version of Visual Basic and managed to find his way to Redmond, Wash., where he works on the Visual Basic product team as a software design engineer tester focused on the Visual Basic compiler. He is still passionate about Visual Basic and can often be found answering questions in the MSDN Visual Basic forums. You can reach Bowles at [Abowles@microsoft.com](mailto:Abowles@microsoft.com).

**THANKS** to the following technical experts for reviewing this article: Kevin Halverson and Beth Massi





## Going NoSQL with MongoDB, Part 2

In my previous article, MongoDB's basics took front and center: getting it installed, running and inserting and finding data. However, I covered only the basics—the data objects used were simple name/value pairs. That made sense, because MongoDB's “sweet spot” includes unstructured and relatively simple data structures. But surely this database can store more than just simple name/value pairs.

In this article, we'll use a slightly different method to investigate MongoDB (or any technology). The procedure, called an exploration test, will help us find a possible bug in the server and, along the way, highlight one of the common issues object-oriented developers will run into when using MongoDB.

### In Our Last Episode ...

First we'll make sure we're all on the same page, and we'll also cover some slightly new ground. Let's look at MongoDB in a bit more structured fashion than we did in the previous article ([msdn.microsoft.com/magazine/ee310029](http://msdn.microsoft.com/magazine/ee310029)). Rather than just create a simple application and hack on it, let's kill two birds with one stone and create exploration tests—code segments that look like unit tests but that explore functionality rather than try to verify it.

Writing exploration tests serves several different purposes when investigating a new technology. One, they help discover whether the technology under investigation is inherently testable (with the assumption that if it's hard to exploration-test, it's going to be hard to unit-test—a huge red flag). Two, they serve as a sort of regression when a new version of the technology under investigation comes out, because they give a heads-up if old functionality no longer works. And three, since tests should be relatively small and granular, exploration tests inherently make learning a technology easier by creating new “what-if” cases that build on previous cases.

Writing exploration tests serves several different purposes when investigating a new technology.

But unlike unit tests, exploration tests aren't continuously developed alongside the application, so once you consider the technology learned, set the tests aside. Don't discard them, however—they can also help separate bugs in application code from those in the

Figure 1 Partial Code for Test Initializer and Cleanup

```
namespace MongoDB_Explore
{
    [TestClass]
    public class UnitTest1
    {
        private static Process serverProcess;

        [ClassInitialize]
        public static void MyClassInitialize(TestContext testContext)
        {
            DirectoryInfo projectRoot =
                new DirectoryInfo(testContext.TestDir).Parent.Parent;
            var mongodbbindir =
                projectRoot.Parent.GetDirectories("mongodb-bin")[0];
            var mongod =
                mongodbbindir.GetFiles("mongod.exe")[0];

            var psi = new ProcessStartInfo
            {
                FileName = mongod.FullName,
                Arguments = "--config mongo.config",
                WorkingDirectory = mongodbbindir.FullName
            };

            serverProcess = Process.Start(psi);
        }

        [ClassCleanup]
        public static void MyClassCleanup()
        {
            serverProcess.CloseMainWindow();
            serverProcess.WaitForExit(5 * 1000);
            if (!serverProcess.HasExited)
                serverProcess.Kill();
        }
    }
    ...
}
```

library or framework. The tests do so by providing a lightweight, application-neutral environment for experimentation without the overhead of the application.

With that in mind, let's create MongoDB-Explore, a Visual C# test project. Add MongoDB.Driver.dll to the list of assembly references and build to make sure everything is good to go. (Building should pick up the one TestMethod that's generated as part of the project template. It will pass by default, so everything should be good, which means that if the project fails to build, something's screwed up in the environment. Checking assumptions is always a good thing.)

As tempting as it would be to jump into writing code right away, though, a problem surfaces pretty quickly: MongoDB needs the external server process (mongod.exe) to be running before client code can connect against it and do anything useful. While

Code download available at [code.msdn.microsoft.com/mag201006WorkProg](http://code.msdn.microsoft.com/mag201006WorkProg).

# PURE

## VISUAL STUDIO AND .NET

**GET TIPS**  
**GET CODE**  
**GET THE BEST**  
**HOW-TO ARTICLES**  
**ON THE NET**

Visit [VisualStudioMagazine.com](http://VisualStudioMagazine.com)



 Microsoft Visual Studio **Microsoft**

Visual Studio  
MAGAZINE



Figure 2 TestMethod to Make Sure the Server Can Be Found and a Connection Made

```
[TestMethod]
public void ConnectInsertAndRemove()
{
    Mongo db = new Mongo();
    db.Connect();

    Document ted = new Document();
    ted["firstname"] = "Ted";
    ted["lastname"] = "Neward";
    ted["age"] = 39;
    ted["birthday"] = new DateTime(1971, 2, 7);
    db["exploretests"]["readwrites"].Insert(ted);
    Assert.IsNotNull(ted["_id"]);

    Document result =
        db["exploretests"]["readwrites"].FindOne(
            new Document().Append("lastname", "Neward"));
    Assert.AreEqual(ted["firstname"], result["firstname"]);
    Assert.AreEqual(ted["lastname"], result["lastname"]);
    Assert.AreEqual(ted["age"], result["age"]);
    Assert.AreEqual(ted["birthday"], result["birthday"]);

    db.Disconnect();
}
```

it's tempting to simply say “Fine, fine, let's start it and get back to writing code,” there's a corollary problem. It's an almost sure bet that at some point, 15 weeks later when looking back at this code, some poor developer (you, me or a teammate) will try to run these tests, see them all fail and lose two or three days trying to figure out what's going on before she thinks to look to see if the server's running.

Lesson: Try to capture all the dependencies in the tests somehow. The issue will arise again during unit-testing, anyway. At that point we'll need to start from a clean server, make some changes and then undo them all. That's easiest to accomplish by simply stopping and starting the server, so solving it now saves time later.

Apparently, sending a DateTime into the MongoDB database without a time doesn't round-trip quite correctly.

This idea of running something before testing (or after, or both) isn't a new one, and Microsoft Test and Lab Manager projects can have both per-test and per-test-suite initializers and cleanup methods. These are adorned by the custom attributes ClassInitialize and ClassCleanup for per-test-suite bookkeeping and TestInitialize and TestCleanup for per-test bookkeeping. (See “Working with Unit Tests” at [msdn.microsoft.com/library/ms182515\(v=VS.80\)](http://msdn.microsoft.com/library/ms182515(v=VS.80)) for more details.) Thus, a per-test-suite initializer will launch the mongod.exe process, and the per-test-suite cleanup will shut the process down, as shown in Figure 1.

The first time this runs, a dialog box will pop up informing the user that the process is starting. Clicking OK will make the dialog go away ... until the next time the test is run. Once that dialog gets too annoying, find the radio box that says, “Never show this dialog box again” and check it to make the message goes away for good. If

firewall software is running, such as Windows Firewall, the dialog will likely make an appearance here also, because the server wants to open a port to receive client connections. Apply the same treatment and everything should run silently. Put a breakpoint on the first line of the cleanup code to verify the server is running, if desired.

Whether this is a bug in the MongoDB server is a value judgment and not something to be explored right now.

Once the server is running, tests can start firing—except another problem surfaces: Each test wants to work with its own fresh database, but it's helpful for the database to have some pre-existing data to make testing of certain things (queries, for example) easier. It would be nice if each test could have its own fresh set of pre-existing data. That will be the role of the TestInitializer- and TestCleanup-adorned methods.

But before we get to that, let's look at this quick TestMethod, which tries to ensure that the server can be found, a connection made, and an object inserted, found and removed, to bring the exploration tests up to speed with what we covered in the previous article (see Figure 2).

If this code runs, it trips an assertion and the test fails. In particular, the last assertion around “birthday” is fired. So apparently, sending a DateTime into the MongoDB database without a time doesn't round-trip quite correctly. The data type goes in as a date with an associated time of midnight but comes back as a date with an associated time of 8 a.m., which breaks the AreEqual assertion at the end of the test.

This highlights the usefulness of the exploration test—without it (as is the case, for example, with the code from the previous article), this little MongoDB characteristic might have gone unnoticed until weeks or months into the project. Whether this is a bug in the MongoDB server is a value judgment and not something

Figure 3 A Simple Object Collection

```
[TestMethod]
public void StoreAndCountFamily()
{
    Mongo db = new Mongo();
    db.Connect();

    var peter = new Document();
    peter["firstname"] = "Peter";
    peter["lastname"] = "Griffin";

    var lois = new Document();
    lois["firstname"] = "Lois";
    lois["lastname"] = "Griffin";

    var cast = new[] {peter, lois};
    db["exploretests"]["familyguy"].Insert(cast);
    Assert.IsNotNull(peter["_id"]);
    Assert.IsNotNull(lois["_id"]);

    db.Disconnect();
}
```

to be explored right now. The point is, the exploration test put the technology under the microscope, helping isolate this “interesting” behavior. That lets developers looking to use the technology make their own decisions as to whether this is a breaking change. Forewarned is forearmed.

Fixing the code so the test passes, by the way, requires the `DateTime` that comes back from the database to be converted to local time. I brought this up in an online forum, and according to the response from the `MongoDB.Driver` author, Sam Corder, “All dates going in are converted to UTC but left as UTC coming back out.” So you must either convert the `DateTime` to UTC time before storing it via `DateTime.ToUniversalTime`, or else convert any `DateTime` retrieved from the database to the local time zone via the `DateTime.ToLocalTime`, by using the following sample code:

```
Assert.AreEqual(ted["birthday"],
    ((DateTime)result["birthday"]).ToLocalTime());
```

This in itself highlights one of the great advantages of community efforts—typically the principals involved are only an e-mail away.

## Adding Complexity

Developers looking to use MongoDB need to understand that, contrary to initial appearances, it isn’t an object database—that is, it can’t handle arbitrarily complex object graphs without help. There are a few conventions that deal with ways to provide that help, but thus far doing so remains on the developer’s shoulders.

For example, consider **Figure 3**, a simple collection of objects designed to reflect the storage of a number of documents describing a well-known family. So far so good. In fact, while it’s at it, the test really should query the database for those objects inserted, as shown in **Figure 4**, just to make sure they’re retrievable. And ... the test passes. Awesome.

Developers looking to use  
MongoDB need to understand  
that, contrary to initial  
appearances, it isn’t an  
object database.

Actually, that might not be entirely true—readers following along at home and typing in the code might find that the test doesn’t pass after all, as it claims that the expected count of objects isn’t matching 2. This is because, as databases are expected to do, this one retains state across invocations, and because the test code isn’t explicitly removing those objects, they remain across tests.

This highlights another feature of the document-oriented database: Duplicates are fully expected and allowed. That’s why each document, once inserted, is tagged with the implicit `_id` attribute and given a unique identifier to be stored within it, which becomes, in effect, the document’s primary key.

So, if the tests are going to pass, the database needs to be cleared before each test runs. While it’s pretty easy to just delete the files

Figure 4 Querying the Database for Objects

```
[TestMethod]
public void StoreAndCountFamily()
{
    Mongo db = new Mongo();
    db.Connect();

    var peter = new Document();
    peter["firstname"] = "Peter";
    peter["lastname"] = "Griffin";

    var lois = new Document();
    lois["firstname"] = "Lois";
    lois["lastname"] = "Griffin";

    var cast = new[] {peter, lois};
    db["exploretests"]["familyguy"].Insert(cast);
    Assert.IsNotNull(peter["_id"]);
    Assert.IsNotNull(lois["_id"]);

    ICursor griffins =
        db["exploretests"]["familyguy"].Find(
            new Document().Append("lastname", "Griffin"));
    int count = 0;
    foreach (var d in griffins.Documents) count++;
    Assert.AreEqual(2, count);

    db.Disconnect();
}
```

in the directory where MongoDB stores them, again, having this done automatically as part of the test suite is vastly preferable. Each test can do so manually after completion, which could get to be a bit tedious over time. Or the test code can take advantage of the `TestInitialize` and `TestCleanup` feature of Microsoft Test and Lab Manager to capture the common code (and why not include the database connect and disconnect logic), as shown in **Figure 5**.

Though the last line of the `CleanDatabase` method is unnecessary because the next test will overwrite the field reference with a new Mongo object, sometimes it’s best to make it clear that the reference is no longer good. *Caveat emptor*. The important thing is that the test-dirtied database is dropped, emptying the files MongoDB uses to store the data and leaving everything fresh and sparkly clean for the next test.

As things stand, however, the family model is incomplete—the two people referenced are a couple, and given that, they should have a reference to each other as spouses, as shown here:

```
peter["spouse"] = lois;
lois["spouse"] = peter;
```

Figure 5 Taking Advantage of `TestInitialize` and `TestCleanup`

```
private Mongo db;

[TestInitialize]
public void DatabaseConnect()
{
    db = new Mongo();
    db.Connect();
}

[TestCleanup]
public void CleanDatabase()
{
    db["exploretests"].Metadata.DropDatabase();

    db.Disconnect();
    db = null;
}
```

Running this in the test, however, produces a `StackOverflowException`—the MongoDB driver serializer doesn't natively understand the notion of circular references and naively follows the references around *ad infinitum*. Oops. Not good.

Fixing this requires you to choose one of two options. With one, the spouse field can be populated with the other document's `_id` field (once that document has been inserted) and updated, as shown in **Figure 6**.

There's a drawback to the approach, though: It requires that the documents be inserted into the database and their `_id` values (which are `Oid` instances, in the MongoDB.Driver parlance) be copied into the spouse fields of each object as appropriate. Then each document is again updated. Although trips to the MongoDB database are fast in comparison to those with a traditional RDBMS update, this method is still somewhat wasteful.

A second approach is to pre-generate the `Oid` values for each document, populate the spouse fields, and then send the whole batch to the database, as shown in **Figure 7**.

This approach requires only the Insert method, because now the `Oid` values are known ahead of time. Note, by the way, that the `ToString` calls on the assertion test are deliberate—this way, the documents are converted to strings before being compared.

What's really important to notice about the code in **Figure 7**, though, is that de-referencing the document referenced via the `Oid` can be relatively difficult and tedious because the document-oriented style assumes that documents are more or less stand-alone or hierarchical entities, not a graph of objects. (Note that the .NET driver provides `DBRef`, which provides a slightly richer way of referencing/dereferencing another document, but it's still not going to make this

**Figure 6 Overcoming the Circular References Problem**

```
[TestMethod]
public void StoreAndCountFamily()
{
    var peter = new Document();
    peter["firstname"] = "Peter";
    peter["lastname"] = "Griffin";

    var lois = new Document();
    lois["firstname"] = "Lois";
    lois["lastname"] = "Griffin";

    var cast = new[] { peter, lois };
    var fg = db["exploretests"]["familyguy"];
    fg.Insert(cast);
    Assert.IsNotNull(peter["_id"]);
    Assert.IsNotNull(lois["_id"]);

    peter["spouse"] = lois["_id"];
    fg.Update(peter);
    lois["spouse"] = peter["_id"];
    fg.Update(lois);

    Assert.AreEqual(peter["spouse"], lois["_id"]);
    TestContext.WriteLine("peter: {0}", peter.ToString());
    TestContext.WriteLine("lois: {0}", lois.ToString());
    Assert.AreEqual(
        fg.FindOne(new Document().Append("_id",
            peter["spouse"])).ToString(),
        lois.ToString());

    ICursor griffins =
        fg.Find(new Document().Append("lastname", "Griffin"));
    int count = 0;
    foreach (var d in griffins.Documents) count++;
    Assert.AreEqual(2, count);
}
```

**Figure 7 A Better Way to Solve the Circular References Problem**

```
[TestMethod]
public void StoreAndCountFamilyWithOid()
{
    var peter = new Document();
    peter["firstname"] = "Peter";
    peter["lastname"] = "Griffin";
    peter["_id"] = Oid.NewOid();

    var lois = new Document();
    lois["firstname"] = "Lois";
    lois["lastname"] = "Griffin";
    lois["_id"] = Oid.NewOid();

    peter["spouse"] = lois["_id"];
    lois["spouse"] = peter["_id"];

    var cast = new[] { peter, lois };
    var fg = db["exploretests"]["familyguy"];
    fg.Insert(cast);

    Assert.AreEqual(peter["spouse"], lois["_id"]);
    Assert.AreEqual(
        fg.FindOne(new Document().Append("_id",
            peter["spouse"])).ToString(),
        lois.ToString());

    Assert.AreEqual(2,
        fg.Count(new Document().Append("lastname", "Griffin")));
}
```

into an object-graph-friendly system.) Thus, while it's certainly *possible* to take a rich object model and store it into a MongoDB database, it's not *recommended*. Stick to storing tightly clustered groups of data, using Word or Excel documents as a guiding metaphor. If something can be thought of as a large document or spreadsheet, then it's probably a good fit for MongoDB or some other document-oriented database.

While it's certainly *possible* to take a rich object model and store it into a MongoDB database, it's not *recommended*.

## More to Explore

We've finished our investigation of MongoDB, but before we wrap up, there are a few more things to explore, including carrying out predicate queries, aggregates, LINQ support and some production administration notes. We'll tackle that next month. (That article is going to be a pretty busy piece!) In the meantime, explore the MongoDB system, and be sure to drop me an e-mail with suggestions for future columns. ■

**TED NEWARD** is a principal with Neward & Associates, an independent firm specializing in enterprise .NET Framework and Java platform systems. He has written more than 100 articles, is a C# MVP, INETA speaker and the author or coauthor of a dozen books, including the forthcoming "Professional F# 2.0" (Wrox). He consults and mentors regularly. Reach him at [ted@tedneward.com](mailto:ted@tedneward.com) and read his blog at [blogs.tedneward.com](http://blogs.tedneward.com).

**THANKS** to the following technical expert for reviewing this article:  
Sam Corder





# The Ins and Outs of ItemsControl

If someone were to ask me what single class most epitomizes the power and flexibility of Windows Presentation Foundation (WPF) and Silverlight, I'd first say that it's a stupid question and then, without a moment's hesitation, respond "DataTemplate."

A DataTemplate is basically a visual tree of elements and controls. Programmers use DataTemplates to give a visual appearance to non-visual data objects. Properties of elements in the visual tree are linked to properties of the data objects through bindings. Although the DataTemplate is most commonly used to define the appearance of objects in an ItemsControl or a ListBox (one of the classes that derive from ItemsControl), you can also use a DataTemplate to define the appearance of an object set to the Content property of a ContentControl or a ContentControl derivative, such as a button.

Creating a DataTemplate—or any other type of Framework-Template derivative such as ControlTemplate or HierarchicalDataTemplate—is one of the few Silverlight programming tasks that can't be done in code. You need to use XAML. It was once possible to create WPF templates entirely in code using Framework-ElementFactory, but I think I was the only person to actually publish examples (in chapters 11, 13 and 16 of my book, "Applications = Code + Markup" [Microsoft Press, 2006]) and the technique has now been deprecated.

What I want to show you in this article is a variation of drag-and-drop: The user simply moves an item from one ItemsControl to another. But my major objective is to implement this whole process with an entirely fluid look and feel that seems natural, and where nothing suddenly jerks or disappears. Of course, "the natural look" is often painstakingly achieved, and any program that strives for fluidity needs to avoid revealing all the awkward machinations just underneath the surface.

I'll be using a combination of techniques I showed in last month's column ("Thinking Outside the Grid," [msdn.microsoft.com/magazine/ff646962](http://msdn.microsoft.com/magazine/ff646962)) as well a DataTemplate that's shared among two ItemsControls and a ContentControl—a concept essential to this whole program.

## Program Layout

The downloadable code that accompanies this article contains a single Silverlight project named ItemsControlTransitions, which you can run from my Web site at [charlespetzold.com/silverlight/ItemsControlTransitions2](http://charlespetzold.com/silverlight/ItemsControlTransitions2). (I'll explain the "2" at the end of this URL

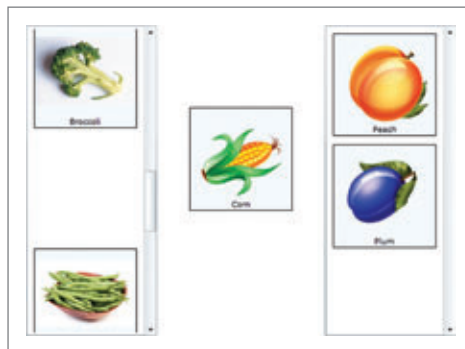


Figure 1 The ItemsControlTransitions Display

later.) You can use the same concepts presented in this Silverlight program in a WPF program.

The program displays two ItemsControls, both contained in ScrollViews. You can visualize the ItemsControl at the left as a "market" selling produce. The one at the right is your "basket." You use the mouse to pick produce items from the market and move them into the basket. **Figure 1** shows the Corn item in transition from market to basket.

Although the Corn item has been moved out of the market, notice the gap in the ItemsControl that continues to indicate the source of the item. If the user releases the mouse button before the dragged item has been positioned over the basket ItemsControl, the program animates the item back into the market. Only when the item is dropped into the basket does that gap close, again with an animation. Depending on where the item is dropped, an animated gap opens to receive the item, and the item is animated into position.

Once an item has been moved from the market, it no longer exists there, but that's a program detail that could easily be changed. No facility exists to remove an item from the basket and move it back into the market, but that feature or something similar could be added fairly easily as well.

**Figure 2** shows the bulk of the XAML file responsible for the basic layout. (Missing are two storyboards with seven animations that I'll describe later.)

The Resources section contains a DataTemplate for displaying the produce items, and a reference to this resource is set to the ItemsTemplate property of the two ItemsControls.

In addition, a Canvas covers the entire area occupied by the program. You'll recall from last month's column how you can use a Canvas to host items that need to "float" over the rest of the UI. The only child of this Canvas is a ContentControl, with its ContentTemplate also set to that DataTemplate. But the Visibility property is set to Collapsed so this ContentControl is initially invisible.

Controls that derive from ContentControl are common in WPF and Silverlight applications, but it's not often you see a ContentControl itself. It turns out to be extremely handy if all you want is to display an object using a DataTemplate. Visually, it's very much like a single item in an ItemsControl.

Code download available at [code.msdn.microsoft.com/mag201006UIF](http://code.msdn.microsoft.com/mag201006UIF).

Figure 2 Partial XAML File Responsible for Basic Layout

```
<UserControl x:Class="ItemsControlTransitions.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Name="this">
  <UserControl.Resources>
    <DataTemplate x:Key="produceDataTemplate">
      <Border Width="144"
        Height="144"
        BorderBrush="Black"
        BorderThickness="1"
        Background="AliceBlue"
        Margin="6">
        <Grid>
          <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
          </Grid.RowDefinitions>
          <Image Grid.Row="0"
            Source="{Binding Photo}" />
          <TextBlock Grid.Row="1"
            Text="{Binding Name}"
            HorizontalAlignment="Center" />
        </Grid>
      </Border>
    </DataTemplate>
    ...
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="White">
    <ScrollView HorizontalAlignment="Left"
      Margin="48">
      <ItemsControl Name="market"
        ItemTemplate="{StaticResource produceDataTemplate}"
        Width="156"
        MouseLeftButtonDown="OnMarketItemsControlMouseLeftButtonDown" />
    </ScrollView>
    <ScrollView HorizontalAlignment="Right"
      Margin="48">
      <ItemsControl Name="basket"
        ItemTemplate="{StaticResource produceDataTemplate}"
        Width="156" />
    </ScrollView>
    <Canvas Name="dragCanvas">
      <ContentControl Name="dragControl"
        ContentTemplate="{StaticResource produceDataTemplate}"
        Visibility="Collapsed" />
    </Canvas>
  </Grid>
</UserControl>
```

The program begins by loading in a little XML database of some produce—using the same files from the `ItemsControlPopouts` project in last month's column—and then filling up the market `ItemsControl` with objects of type `ProduceItem`. This class has `Name` and `Photo` properties that the `DataTemplate` references to display each item.

## Pulling Items from ItemsControl

The `ItemsControl` for the market has a handler set for `MouseLeftButtonDown`. On receipt of this event, the program needs to dislodge an item from the four-wall confines of the `ItemsControl` and allow it to track the mouse. But the item can't actually be removed from the `ItemsControl` or the gap will automatically close up.

As I demonstrated in last month's column, you can access the `ItemContainerGenerator` property of an `ItemsControl` to get a class that can associate each item in an `ItemsControl` with the visual tree that's been generated to display that particular item. This visual tree has a root element of type `ContentPresenter`.

My first impulse was to apply a `TranslateTransform` to the `RenderTransform` property of the `ContentPresenter`, to allow it to float outside the `ItemsControl`. I know from experience, however, that this doesn't work at all. The problem isn't the `ItemsControl` itself; the problem is the `ScrollView`, which of necessity clips its children to its interior. (More about the rationale behind this clipping shortly.)

Instead, the program copies the clicked `ProduceItem` in the `ItemsControl` to the `ContentControl`, and positions the `ContentControl` precisely over the `ContentPresenter` of the clicked item. (The program can obtain the location of the `ContentPresenter` relative to the `Canvas` using the always handy `TransformToVisual` method.) You'll recall that the XAML file sets the `Visibility` property of the `ContentControl` to `Collapsed`, but now the program toggles that property to `Visible`.

At the same time, the `ContentPresenter` in the `ItemsControl` is made invisible. In WPF, you can do this simply by setting the `Visibility` property to `Hidden`, which makes the item invisible but otherwise causes the element's size to be observed for layout purposes. The `Visibility` property in Silverlight doesn't have a `Hidden` option, and if you set the `Visibility` property of the `ContentPresenter` to `Collapsed`, the gap will close up. Instead, you can mimic a `Visibility` setting of `Hidden` by simply setting the `Opacity` property to zero. The element is still there, but it's invisible. As you experiment with the program, you'll discover that the transition from the item in the `ItemsControl` to the draggable `ContentControl` is imperceptible.

At this point, the `ContentPresenter` in the `ItemsControl` displays nothing but an empty hole, and the `ContentControl` displaying the item can now be dragged around the screen with the mouse.

## The Item Drop

Back when I was writing books about the Win16 and Win32 APIs, I spent whole chapters showing how to use scroll bars to display more text in a window than can fit there. Today we simply use a `ScrollView`, and everyone is much happier—me most of all.

Despite its essential role in WPF and Silverlight layout, the `ScrollView` can be a little tricky to use at times. It has some peculiarities that can be a little puzzling, and this program reveals one of them. See if you can anticipate the problem.

We left the user moving a produce item around the screen with the mouse. If the user drops the produce item somewhere over the `ItemsControl` representing the basket, it becomes part of that collection. (More on this process shortly.) Otherwise, the program animates the item back to its origin using two animations in the `returnToOriginStoryboard` in `MainPage.xaml`. At the conclusion of the animation, the `Opacity` property of the `ContentPresenter` is set to one, the `Visibility` property of the `ContentControl` is set to `Collapsed`, and the drag event is concluded with everything back to normal.

To determine if the produce item is being dropped on the `ItemsControl`, the program calculates a `Rect` object representing the location and size of the dragged `ContentControl` and another `Rect` object representing the location and size of the `ItemsControl`. In both cases, the program uses the `TransformToVisual` method to obtain the location of the upper-left corner of the control—the point (0, 0)—relative to the page, and the `ActualWidth` and `ActualHeight` properties to obtain the control's size. The `Rect` structure's `Intersect`

method then calculates an intersection of the two rectangles, which will be non-empty if there's some overlap.

This works fine except when the ItemsControl has more items than can fit in the vertical space allowed for it. The ScrollViewer then kicks into action by making its vertical scrollbar visible so you can scroll through the items. However, the ItemsControl inside the ScrollViewer actually believes itself to be larger than what you're seeing; in a very real sense, the ScrollViewer is providing only a viewable window (called a "viewport") on the ItemsControl. The location and size information you obtain for that ItemsControl always indicates the full size (called the "extent" size) and not the viewport size.

This is why ScrollViewer needs to clip its child. If you've been working with Silverlight for a while, you might be particularly accustomed to a certain laxity regarding clipping of children. You can almost always use RenderTransform to escape from a parent's boundaries. However, ScrollViewer definitely needs to clip or it simply can't work right.

This means that you can't use the apparent dimensions of the ItemsControl to determine a valid drop, because in some cases the ItemsControl extends above and below the ScrollViewer. For that reason, my program determines a valid drop rectangle based on horizontal dimensions of the ItemsControl—because it wants to exclude the area occupied by the scrollbar—but the vertical dimensions of the ScrollViewer.

When the ContentControl is dropped on the ItemsControl, it could be overlapping two existing items, or just one if it's being dropped on the top or bottom of the stack of items, or none at all. I wanted to insert the new item in the spot closest to where it's dropped, which required enumerating through the items in the ItemsControl (and their associated ContentPresenter objects) and determining a good index to insert the new item. (The GetBasketDestinationIndex method is responsible for determining this index.) After the item is inserted, the ContentPresenter associated with that new item is given an initial height of zero and an opacity of zero, so it isn't initially visible.

Following this insertion, the program initiates the storyboard called transferToBasketStoryboard with five animations: one to decrease the height of the invisible ContentPresenter in the ItemsControl for the market; another to increase the height of the invisible ContentPresenter newly created in the basket ItemsControl; and two more to animate the Canvas.Left and Canvas.Top attached properties to slide the ContentControl into place. (I'll discuss the fifth animation shortly.) **Figure 3** shows the gap widening as the ContentControl approaches its destination.

When the animation ends, the new ContentPresenter is given an opacity of one and the ContentControl is given a visibility of Collapsed, and now we're back to just dealing with two normal ItemsControls inside ScrollViewers.

## The Top and Bottom Problem

Earlier in this article I gave you the URL [charlespetzold.com/silverlight/ItemsControlTransitions2](http://charlespetzold.com/silverlight/ItemsControlTransitions2) to try out the program. An earlier version of

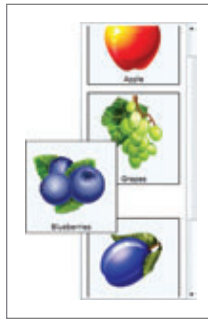


Figure 3 The Animation to Move a New Item into Place

the program can be run from [charlespetzold.com/silverlight/ItemsControlTransitions](http://charlespetzold.com/silverlight/ItemsControlTransitions), without the "2" on the end. Using this earlier version, move several produce items over to the basket—enough to make the vertical scrollbar appear. Now drag another one over and position it straddling the bottom of the ScrollViewer. When you release the mouse button, the ContentControl moves down toward an area of the ItemsControl that's invisible, and then suddenly disappears. The item has been correctly inserted (as you can verify by scrolling down), but not very elegantly.

Now scroll the ScrollViewer so the top item is only partially visible. Move another item from the basket and position it so it will be inserted before that item.

The new item slides into the ItemsControl, but it's not entirely visible. It's not quite as bad as the problem at the bottom of the ItemsControl, but it still needs some help.

The fix? Some way to programmatically scroll the ScrollViewer is required. The amount of vertical scrolling currently in effect for a ScrollViewer is provided through the VerticalOffset property. This number is a positive offset from the top of the entire ItemsControl to the location in the control that's displayed at the top of the ScrollViewer.

Wouldn't it be nice to simply animate that VerticalOffset property? Unfortunately, only the get accessor is public. Fortunately, it's possible to programmatically scroll the ScrollViewer, but you need to call a method named ScrollToVerticalOffset.

To accomplish this little scrolling job through the Silverlight animation facility, I defined a dependency property named Scroll in MainPage itself. In the XAML file, I gave the page a name of "this," and defined a fifth animation in transferToBasketStoryboard to target this property:

```
<DoubleAnimation x:Name="scrollItemsControlAnima"
    Storyboard.TargetName="this"
    Storyboard.TargetProperty="Scroll" />
```

The OnMouseLeftButtonUp override calculates the From and To values of this animation. (You can compare the effect of this additional animation by commenting out the block of code beginning with the comment "Calculate ScrollViewer scrolling animation.") As this Scroll property is animated, its property-changed handler calls the ScrollToVerticalOffset method of the ScrollViewer with the animated value.

## Toward a Fluid UI

Many, many years ago, computers were much slower than they are now, and nothing that happened on the screen was ever very startling. Today, programs can implement UIs that entirely change their appearances in the blink of an eye. But that's unsatisfactory as well. Often we can't even see what's going on, so now we find it necessary to deliberately slow down the UI and make transitions more fluid and natural. Silverlight 4 introduced some "fluid UI" features that I'm eager to discuss, but even in Silverlight 3 it's possible to begin the journey in that direction. ■

---

**CHARLES PETZOLD** is a longtime contributing editor to MSDN Magazine. He is currently writing "Programming Windows Phone 7 Series," which will be published as a free downloadable e-book in the fall of 2010. A preview edition is currently available through his Web site [charlespetzold.com](http://charlespetzold.com).





# Chainsaw Development

It takes Microsoft three versions to get a product right, Windows itself being the classic example. Visual Studio 2010 and the Microsoft .NET Framework 4 represent the third release of Windows Presentation Foundation (WPF) and its tools. Right on schedule, my clients are telling me, “Yeah, looks about ready now, help us learn it, then we’ll try a pilot project.” But newcomers to WPF are often distracted by its glitz: They forget that their ultimate goal is making their users happier and more productive, not titillating their own vanity by cramming flashy gizmos into a program for the sheer pain of it. Above all, they forget that their program is just one of many that users switch among, all day every day, and that commonality among UIs—in other words, most Windows programs working more or less like each other—is the key to their users’ satisfaction and hence to their programs’ success.

Few people under age 35 remember DOS programs, when UIs had no commonality whatsoever. For example, most DOS programs had no menus, requiring snap-on keyboard templates to remind users of commands. (OK, I guess that’s some commonality.) A few DOS programs contained menus but didn’t show them until the user pressed a specific key, and naturally every program used a different key and showed the menu in a different place. Microsoft Word used ESC and the menu appeared below the document; Lotus 1-2-3 used the forward slash ‘/’ and the menu appeared above the document; Farsight (another spreadsheet) used F3. Every user had to (*gak!*) read the manual (remember those?) to even start poking at a new app, and then had to switch mental command sets every time he switched applications.

The biggest growth driver of the Windows user platform, besides Solitaire, is the standardized UI that its API encourages. The primary control structure is a menu at the top of a program’s window. Keyboard shortcuts are listed on the menu items as a teaching aid, toolbars provide graphic shortcuts and so on. These standards, like tool tips and right-click

context menus, have evolved over time, and they continue to evolve today (the Office Ribbon control, for example). No one ever reads a manual. Users expect a new Windows program to instantly explain itself through its UI, and will dump any that don’t.

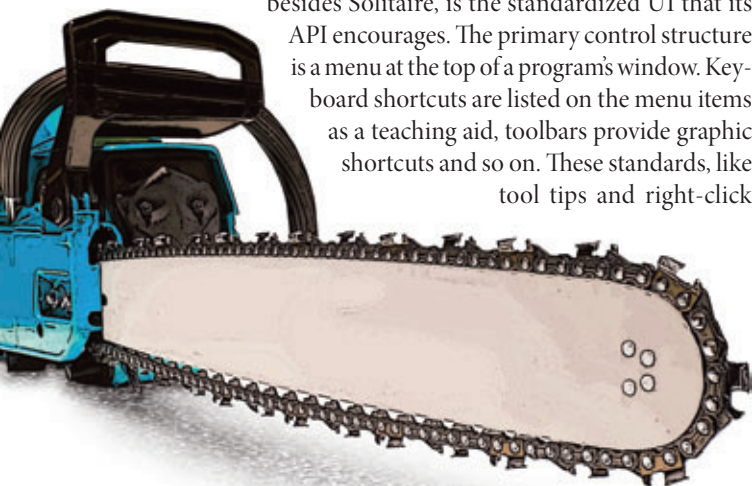
## WPF is much more powerful than Windows Forms.

We don’t have these standards for the new features of WPF yet, and that’s a real problem. For example, many articles explain how to program animation in WPF. But besides my paper, “Using WPF for Good and Not Evil” ([rollthunder.com/SoftwareThatDoesntSuck/WpfForGoodAndNotEvil.htm](http://rollthunder.com/SoftwareThatDoesntSuck/WpfForGoodAndNotEvil.htm)), I see no discussions in the Windows community of what information an animation communicates to a user, what effects an animation therefore has on the user’s productivity and satisfaction, or any sort of guidelines on where animation should be used and where it shouldn’t. That’s why, whenever I teach a class on WPF, I always insist on devoting at least a day to UI design, teaching my clients not just to write WPF code, but to start from the user’s needs and work inward, rather than starting from the toolkit and working outward.

WPF is much more powerful than Windows Forms, as a chainsaw is more powerful than a handsaw. I see great exultation over that power, and the exalters are absolutely correct about its magnitude. But I see zero discussion of the careful thought needed to safely and productively manage that power to make users happier—which is our ultimate goal.

This needs to change, and it needs to change now. After four-plus years of experimentation, we ought to have some notion of which usage patterns in WPF make users happier and which accomplish the opposite. With WPF poised to become the mainstream of Windows desktop development, I call on Microsoft to publish UI design guidelines for it; not how to program this or that feature, but when and where and why to use it. A company that manufactures a chainsaw incurs a duty to teach its customers which end of it to hold. ■

**DAVID S. PLATT** teaches Programming .NET at Harvard University Extension School and at companies all over the world. He is 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](http://rollthunder.com).



# Why switch to ActiveReports products



## from Our Competition?

# ACTIVEREPORTS 6



- Unmatched customization, performance and quality
- Rich collection of report designing and rendering components
- Medium Trust support in ASP.NET environment
- Wide range of cross-platform export and preview formats
- Powerful multilanguage PDF reporting
- End User Report Designer with flexible API and full customization
- Royalty-free licensing for Web and Windows applications

- ✓ Three powerful products to meet all reporting and BI needs.
- ✓ Flexible pricing options. Get one or get all.
- ✓ Simple developer based licensing. Site license options.
- ✓ Zero run time fees. Royalty-free deployment.

**GrapeCity PowerTools**  
1 919-460-4551 / 1 800-645-5913

**Find out why.**  
[www.GCPowerTools.com/Switch](http://www.GCPowerTools.com/Switch)



Spreadsheets



Reporting



Analysis

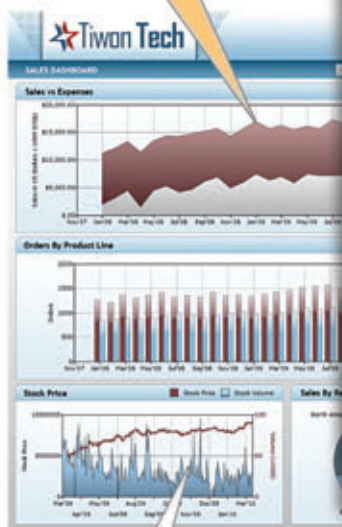




# The Dashboard Platform for Developers like you

Build more **powerful** and **effective**  
dashboards, **faster**.

Support For Numerous  
Data Sources



Rapid Dashboard  
Development

Leverages The Latest  
Silverlight Technology

Full Scripting  
Capabilities With  
DundasScript™

More Data  
Visualization Options

OLAP Support

Extensible And  
Customizable With  
An Open API



Dundas Dashboard is a ground-breaking, extensible solution that uses a revolutionary approach to dashboard creation, providing you with a unified view of key metrics and a new level of strategic insight and decision-making.



Powered by  
Microsoft Silverlight



[www.dundas.com/dashboard](http://www.dundas.com/dashboard)

(416) 467-5100 • (800) 463-1492

Silverlight is a trademark of Microsoft Corporation in the United States and/or other countries.