THE MICROSOFT JOURNAL FOR DEVELOPERS

DECEMBER 2010 VOL 25 NO 12

msdn magazine

MOBILE APPS

Microsoft

DESIGN DEVELOP EXPERIENCE

Quince

Using Quince™, you and your team can collaborate on the user interface using wireframes, designs and examples.

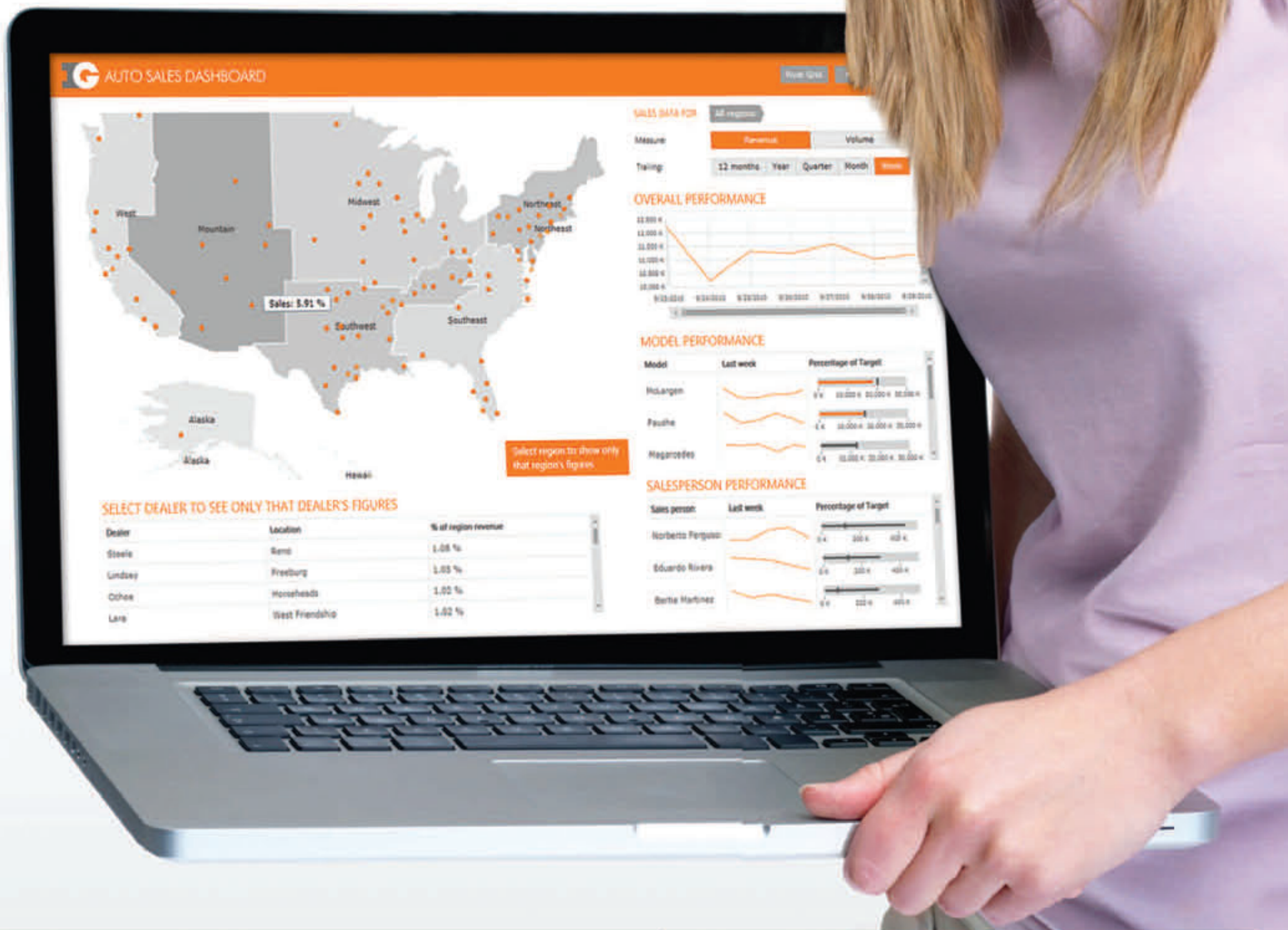
Net**Advantage**®
for Silverlight Data Visualization
for WPF Data Visualization

Then use NetAdvantage® UI controls, like the map control used here, to bring the application to life quickly & easily.

NetAdvantage® ULTIMATE

for ASP.NET, Windows Forms, WPF, Silverlight, WPF Data Visualization, Silverlight Data Visualization

From start to finish, Infragistics gives you the tools to create impressive user experiences that'll make end users happy!



SEE HOW WE USE THE TOOLS TO CREATE THIS KILLER APP AT **INFRAGISTICS.COM/IMPRESS**

Infragistics®

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @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

1-800-IT-FINDS • www.dtSearch.com

The Smart Choice for Text Retrieval® since 1991

msdn magazine

DECEMBER 2010 VOLUME 25 NUMBER 12

LUCINDA ROWLEY Director
DIEGO DAGUM Editorial Director/mmeditor@microsoft.com
KERI GRASSL Site Manager

KEITH WARD Editor in Chief/mmeditor@microsoft.com
TERRENCE DORSEY Technical Editor
DAVID RAMEL Features Editor
WENDY GONCHAR Managing Editor
KATRINA CARRASCO 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

RedmondMediaGroup

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

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

1105 MEDIA INC

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 US funds are: U.S. $35.00, International $60.00. Annual digital subscription rates payable in U.S. funds are: U.S. $25.00, International $25.00. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: *MSDN Magazine*, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or 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

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

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

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

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

Microsoft

BPA WORLDWIDE

Printed in the USA

Your best source for software development tools!

programmer's paradise®

LEADTOOLS Document Imaging v17.0:

by LEAD Technologies

LEADTOOLS Document Imaging has every component you need to develop powerful image-enabled business applications including specialized bi-tonal image display and processing, document clean up, annotation, high-speed scanning, advanced compression (CCITT G3/G4, JBIG2, MRC, ABC) and more.

- Multi-threaded OCR/ICR/OMR/MICR/Barcodes (1D/2D)
- Forms recognition/processing
- PDF and PDF/A
- Win32/x64 binaries for C/C++, .NET, Silverlight, WPF, WCF, & WF

Paradise #
L05 03301A01

$2,007.99

programmers.com/LEAD

STOP OVERBUYING SOFTWARE TODAY!

Eliminate Wasteful Software License Spend:

- Control your software licensing costs
- Stop paying for licenses you're not using
- Reduce your license spend by $300+ per desktop user

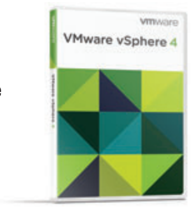
FREE 30-DAY PROOF OF CONCEPT

Learn more:
programmers.com/eliminate-wasteful-license-spend

VMware vSphere Essentials Kit Bundle

vSphere Essentials provides an all-in-one solution for small offices to virtualize three physical servers for consolidating and managing applications to reduce hardware and operating costs with a low up-front investment. vSphere Essentials includes:

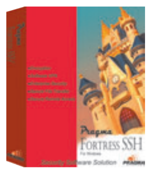
- VMware ESXi and VMware ESX (deployment-time choice)
- VMware vStorage VMFS
- Four-way virtual SMP
- VMware vCenter Server Agent
- VMware vStorage APIs/VCB
- VMware vCenter Update Manager
- VMware vCenter Server for Essentials



for 3 hosts
Paradise #
V55 85101C02

$446.99

programmers.com/vSphere



"Pragma SSH for Windows" Best SSH/SFTP/SCP Servers and Clients for Windows

by Pragma Systems

Get all in one easy to use high performance package. FIPS Certified and Certified 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
- Hyper-V and PowerShell support
- Runs in Windows 2008R2/2008/2003/7/Vista/XP/2000

Certified for Windows 7/2008R2

Paradise #
P35 04201A01

$550.99

programmers.com/pragma

InstallShield Professional for Windows 2011

by Flexera Software

NEW RELEASE!

Microsoft's Installation Solution of Choice for Visual Studio 2010 applications.

If your software targets Windows®, InstallShield® is your solution. It makes it easy to author high-quality reliable Windows Installer (MSI) and InstallScript installations and App-V™ virtual packages for Windows platforms, including Windows 7. InstallShield, the industry standard for MSI installations, also supports the latest Microsoft technologies including Visual Studio 2010, .NET Framework 4.0, IIS7.0, SQL Server 2008 SP1, and Windows Server 2008 R2 and Windows Installer 5, keeping your customers happy and your support costs down.

FLEXERA SOFTWARE™ InstallShield

Upg from any Active IS Pro + IS Pro Silver Mtn
Paradise #
I21H02401B01

$1,384.99

programmers.com/flexera

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

ActiveReports 6

by GrapeCity

The de facto standard reporting tool for Microsoft Visual Studio.NET

- Fast and Flexible reporting engine
- Flexible event-driven API to completely control the rendering of reports
- Wide range of Export and Preview formats including Windows Forms Viewer, Web Viewer, Adobe Flash and PDF
- XCopy deployment
- Royalty-Free Licensing for Web and Windows applications

NEW VERSION 6!

Professional Ed.
Paradise #
D03 04301A01

$1,310.99

programmers.com/grapecity

Microsoft Visual Studio Professional 2010

by Microsoft

Microsoft Visual Studio 2010 Professional with MSDN Essentials Subscription is an integrated environment that simplifies creating, debugging and deploying applications. Unleash your creativity and bring your vision to life with powerful design surfaces and innovative collaboration methods for developers and designers. Work within a personalized environment, targeting a growing number of platforms, including Microsoft SharePoint and cloud applications and accelerate the coding process by using your existing skills. Integrated support for Test-First Development and new debugging tools let you find and fix bugs quickly and easily to ensure high quality solutions.

with MSDN
Paradise #
M47 40201A02

$1,060.99

programmers.com/microsoft

TX Text Control 15.1

Word Processing Components

New Service Pack!

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

Professional Edition
Paradise #
T79 02101A02

$1,220.99

Download a demo today.

programmers.com/theimagingsource

FREE WEBINAR SERIES: MAXIMIZING DATA QUALITY FOR VALUE AND ROI

Data is a company's greatest asset. Enterprises that can harness the power of their data will be strategically positioned for the next business evolution. But too often businesses get bogged down in defining a Data Management process, awaiting some "magic bullet", while the scope of their task grows larger and their data quality erodes. Regardless of how your eventual data management solution is implemented, there are processes that need to occur now to facilitate that process. In this new series, with a mixture of slides, demonstrations and Q&A sessions, we will discuss how to use your existing Data Modeling assets to build the foundations of strong data quality.

REGISTER TODAY! programmers.com/CA

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

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

$3,931.99

programmers.com/ca

NEW! Intel® Parallel Studio 2011

by Intel

NEW RELEASE!

A comprehensive, all-in-one toolkit for Microsoft Visual Studio® C/C++ developers, Intel® Parallel Studio 2011 simplifies the analysis, compiling, debugging, error-checking, and tuning of your serial and threaded apps.

With Intel Parallel Studio, get everything you need to optimize legacy serial code, exploit multicore, and scale for manycore.

Single User DVD
Paradise #
I23 63101E03

$753.99

programmers.com/intel

866-719-1528 **programmersparadise.com**

Prices subject to change. Not responsible for typographical errors.

Windows Phone 7 App-roval

A quick glance at this magazine's cover reveals that our theme is Windows Phone 7 application development. In this space a few months ago, I listed some factors that will be important in selling your app (msdn.microsoft.com/magazine/gg232771), and in this issue, David Platt's Don't Get Me Started column (p. 92) has more good advice.

But I think it's time to hear from a developer who's actually built an app for Windows Phone 7. Bob Baker wrote his first PC statistics application in 1978. Since then, he's worked in a lot of different areas of development, including some "Silverlight insider work," as he calls it, back in the day. Currently, he's working as a contractor at a Fortune 50 company. Suffice it to say that he'd be recognized as an expert witness if this were a court case.

Baker's also a musician who plays bass and guitar. It's only natural, then, that his first Windows Phone 7 app is a metronome: he calls it "MetroGnome," as it's gnome-themed. It's his first-ever mobile app on any device, so Baker didn't know exactly what to expect.

Overall, Baker says that developing the app wasn't particularly difficult because he has a deep Silverlight background. The first version of MetroGnome, in fact, took only about 12 hours of development time. "It's all .NET," Baker says. "The general accessibility of writing something on this platform is miles ahead of Windows Mobile 6 and 5 ... That model of building something with the whole infrastructure is in place." One thing Baker says he would like to have, though, is OS access.

Baker also pointed out a few other areas he'd like to see Microsoft improve upon. And it starts with better documentation. "The documents are pretty thin. Most of what I've learned is the result of pinging [Microsoft support], finding blog posts, sample code," Baker says. He adds: "I wish [Microsoft had] spent more time tying together app submission guidelines and UI guidelines. There aren't a lot of helpful samples, at all," from Microsoft.

The next set of challenges for Baker came when he submitted his app for Microsoft approval. He says he submitted MetroGnome for approval on Oct. 18. After nearly a week of waiting, he found out his app failed the approval process. The problem was that he didn't know why the app failed. The application-publishing site—or "App Hub," as it's called—didn't tell him why the app failed. A PDF was supposed to give him information on why the failure occurred, but the document was blank, Baker says.

"Meanwhile, I didn't know if it was something wrong with my code—or something else," he explains. Baker e-mailed support, and eventually got a return PDF that said there was a screenshot failure. The explanation for the failure, according to Baker, was that he didn't take a screenshot of the running application. There was no other explanation—and the one he received was an unhelpful one at that, as he included screenshots with his submission.

Ultimately, the issue turned out to be a quirk in how a screenshot is captured. "I spent four to six hours banging my head against the App Hub, trying to get it to do what I wanted to do—checking forums, sending e-mail to support, etc.," he says. It turns out that doing a "region capture" screenshot results in the inclusion of a border from the phone emulator he used. Those added pixels meant that Baker's screenshot was too small, causing the failure. "There's a whole bunch of people who ran into the same thing," he says. (As one developer said [geekswithblogs.net/dlussier/archive/2010/10/27/142465.aspx]: "The screenshot can be taken from an emulator, but can't show the emulator.") After resubmitting the app with the new "window capture" screenshots, the app sailed through the approval process in less than a day.

In all, Baker says he'd give the current Microsoft Windows Phone 7 submission/approval process a "B-minus ... There are lots of bugs in the App Hub system. I hope they get it fixed." He also says that he realizes this is brand-new territory for Microsoft, and is overall happy with the process. In fact, he enjoys mobile development enough now that he's working on his next Windows Phone 7 app: a tuner for musical instruments.

What have your efforts to build a Windows Phone 7 app been like? Let me know at mmeditor@microsoft.com.

Keith Ward

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

OnTime Now!

Cloud-based Scrum Tool for Developers

In just seconds, get your team started with the world's most advanced agile project management solution. See why *OnTime Now* is so popular with agile dev teams!

$79 / month for 5 users • Free 30-day Team Trial

axosoft

Software for Software Development

Subversion for Visual Studio

It's every developer's dream come true when it comes to using *Subversion* with *Microsoft Visual Studio*: fast, keyboard-friendly and stays out of the way!

Just $49 / user

Local Scrum Tool for Developers

Ship software on time with the locally-installed, award-winning project management tool for agile development teams. *MS-SQL* server back-end.

Starts @ $395 for teams of 5 • Free for 1 user

www.axosoft.com

(800) 653-0024 • (480) 362-1900

TERRENCE DORSEY

Windows Azure Development Resources

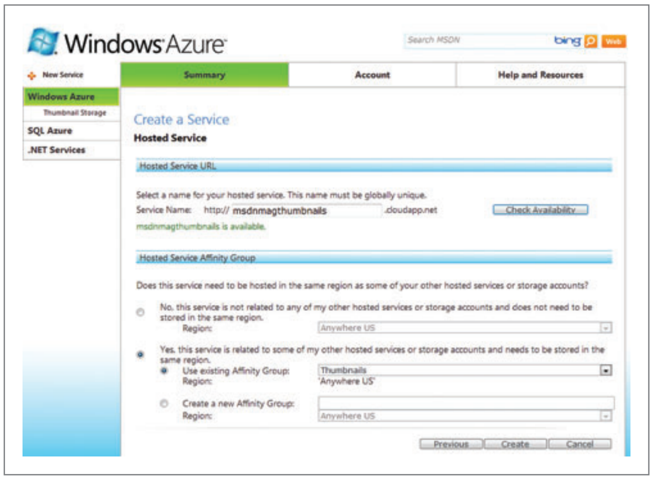
As you've probably read elsewhere in *MSDN Magazine*, the Windows Azure platform is Microsoft's stack of cloud computing resources that range from coding, testing and deploying Visual Studio and Windows Azure AppFabric to Windows Azure itself and the SQL Azure storage services. Here's a collection of tools and information that will get you writing apps for Windows Azure today.

Getting Started

When you're ready to start developing for the Windows Azure platform, your first stop should be the Windows Azure Developer Center on MSDN (**msdn.microsoft.com/windowsazure**). Here you'll find information about the entire platform along with links to documentation, tools, support forums and community blog posts.

Next, head over to the Windows Azure portal (**windows.azure.com**) and set up your account. This gives you access to Windows Azure, SQL Azure for storage and Windows Azure AppFabric (**Figure 1**). You'll need a Windows Live ID to sign up. If you don't have one already, there's a link on the sign-in page.

As we go to press, Microsoft is offering an introductory special that lets you try out many features of the Windows Azure platform at no charge. See **microsoft.com/windowsazure/offers/** for details.



Create a Service

Hosted Service

Hosted Service URL

Select a name for your hosted service. This name must be globally unique.

Service Name: http:// msdnmagthumbnails .cloudapp.net

msdnmagthumbnails is available.

Hosted Service Affinity Group

Does this service need to be hosted in the same region as some of your other hosted services or storage accounts?

No, this service is not related to any of my other hosted services or storage accounts and does not need to be stored in the same region.
Region: Anywhere US

Yes, this service is related to some of my other hosted services or storage accounts and needs to be stored in the same region.
Use existing Affinity Group: Thumbnails
Region: 'Anywhere US'
Create a new Affinity Group:
Region: Anywhere US

Figure 1 **Running a Service on Windows Azure**

Developer Tools

Before you can start slinging code, you'll need to get your development environment set up. While you could probably build your Windows Azure app with Notepad and an Internet connection, it's going to be a lot more productive—and enjoyable—to use tools optimized for the job.

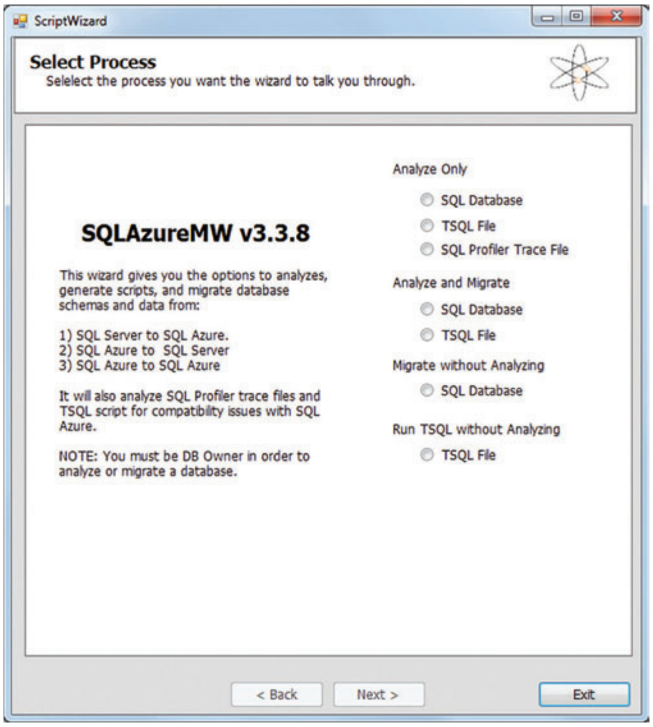
If you don't have Visual Studio 2010, you can enjoy (most of) the benefits of a Windows Azure-optimized development environment with Visual Web Developer 2010 Express (**asp.net/vwd**). You can get it via the Web Platform Installer (**microsoft.com/express/web**), which can also install SQL Server 2008 Express Edition, IIS, and extensions for Silverlight and ASP.NET development.

If you're already using Visual Studio, simply download and install the Windows Azure Tools for Microsoft Visual Studio (**bit.ly/aAsgjt**). These tools support both Visual Studio 2008 and Visual Studio 2010 and contain templates and tools specifically for Windows Azure development. Windows Azure Tools includes the Windows Azure SDK.

Moving Data from SQL Server

If you're migrating an existing Web application to Windows Azure, you'll need some way to migrate the app's data as well. For apps that employ SQL Server 2005 or SQL Server 2008 as a data store, the SQL Azure Migration Wizard (**sqlazuremw.codeplex.com**) makes this transition a lot easier (**Figure 2**). The wizard not only transfers the actual data, but also helps you identify and correct possible compatibility issues before they become a problem for your app.

To get a handle on how to use the SQL Server Migration Wizard, along with a lot of other helpful information about moving existing apps to Windows Azure, see "Tips for Migrating Your Applications



Select Process

Selelect the process you want the wizard to talk you through.

SQLAzureMW v3.3.8

This wizard gives you the options to analyzes, generate scripts, and migrate database schemas and data from:

1) SQL Server to SQL Azure.
2) SQL Azure to SQL Server
3) SQL Azure to SQL Azure

It will also analyze SQL Profiler trace files and TSQL script for compatibility issues with SQL Azure.

NOTE: You must be DB Owner in order to analyze or migrate a database.

Analyze Only
- SQL Database
- TSQL File
- SQL Profiler Trace File

Analyze and Migrate
- SQL Database
- TSQL File

Migrate without Analyzing
- SQL Database

Run TSQL without Analyzing
- TSQL File

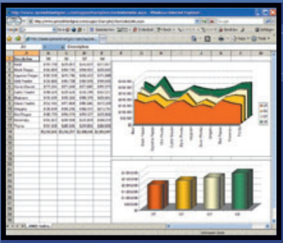
Figure 2 **SQL Azure Migration Wizard**

20 Minutes to 4 Seconds...

SpreadsheetGear for .NET reduced the time to generate a critical Excel Report "from 20 minutes to 4 seconds" making his team "look like miracle workers."

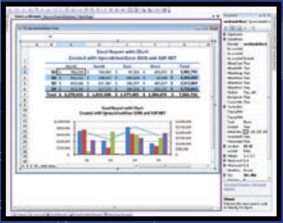
Luke Melia, Software Development Manager at Oxygen Media in New York

SpreadsheetGear 2010



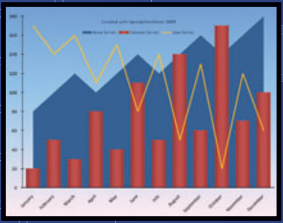
ASP.NET Excel Reporting

Easily create richly formatted Excel reports without Excel using the new generation of spreadsheet technology built from the ground up for scalability and reliability.



Excel Compatible Windows Forms Control

Add powerful Excel compatible viewing, editing, formatting, calculating, charting and printing to your Windows Forms applications with the easy to use WorkbookView control.



Create Dashboards from Excel Charts and Ranges

You and your users can design dashboards, reports, charts, and models in Excel rather than hard to learn developer tools and you can easily deploy them with one line of code.

Download the FREE fully functional 30-Day evaluation of SpreadsheetGear 2010 today at

www.SpreadsheetGear.com.

SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com to the Cloud" in the August 2010 issue of *MSDN Magazine* (**msdn.microsoft.com/magazine/ff872379**).

Security Best Practices

You need to take security into consideration with any widely available application, and cloud apps are about as widely available as they come. The Microsoft patterns & practices team launched a Windows Azure Security Guidance project in 2009 to identify best practices for building distributed applications on the Windows Azure platform. Their findings have been compiled into a handy PDF that covers checklists, threats and countermeasures, and detailed guidance for implementing authentication and secure communications (**bit.ly/aHQseJ**). The PDF is a must-read for anyone building software for the cloud.

You need to take security into consideration with any widely available application, and cloud apps are about as widely available as they come.

PHP Development on Windows Azure

Dating from before even the days of classic ASP, PHP continues to be a keystone of Web application development. With that huge base of existing Web apps in mind, Microsoft created a number of tools that bring support for PHP to the Windows Azure platform. These tools smooth the way for migrating older PHP apps to Windows Azure, as well as enabling experienced PHP developers to leverage their expertise in the Microsoft cloud.

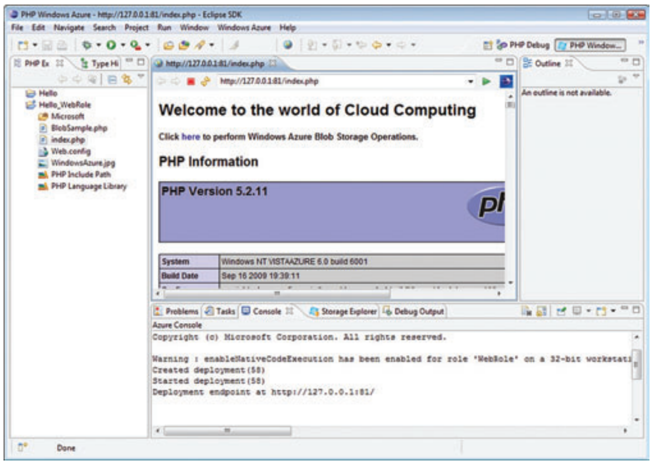
There are four tools for PHP developers:

- Windows Azure Companion helps you install and configure the PHP runtime, extensions and applications on Windows Azure.
- Windows Azure Tools for Eclipse for PHP is an Eclipse plug-in that optimizes the open source IDE for developing applications for Windows Azure (**Figure 3**).
- Windows Azure Command-Line Tools for PHP provides a simple interface for packaging and deploying PHP applications on Windows Azure.
- Windows Azure SDK for PHP provides an API for leveraging Windows Azure data services from any PHP application.

You'll find more information about the tools and links to the downloads on the Windows Azure Team Blog at **bit.ly/ajMt9g.**

Windows Azure Toolkit for Facebook

Building applications for Facebook is a sure-fire way to reach tens of millions of potential customers. And if your app takes off, Windows Azure provides a platform that lets you scale easily as demand grows. The Windows Azure Toolkit for Facebook (**azuretoolkit.codeplex.com**) gives you a head start in building your own highly scalable Facebook app. Coming up with the next FarmVille is still up to you, though!



Welcome to the world of Cloud Computing

Click here to perform Windows Azure Blob Storage Operations.

PHP Information

PHP Version 5.2.11

System	Windows NT VISTAAZURE 6.0 build 6001
Build Date	Sep 16 2009 19:39:11

```
Copyright (c) Microsoft Corporation. All rights reserved.

Warning : enableNativeCodeExecution has been enabled for role 'WebRole' on a 32-bit workstati
Created deployment(58)
Started deployment(58)
Deployment endpoint at http://127.0.0.1:81/
```

Figure 3 **Windows Azure Tools for Eclipse**

Windows Azure SDK for Java

PHP developers aren't the only ones getting some native tools for Windows Azure. Now Java developers can also work in their language of choice and get seamless access to Windows Azure services and storage. The Windows Azure SDK for Java (**windowsazure4j.org**) includes support for Create/Read/Update/Delete operations on Windows Azure Table Storage, Blobs and Queues. You also get classes for HTTP transport, authorization, RESTful communication, error management and logging.

Setting up Your System

Here are a few useful blog posts from the Windows Azure developer community that walk you through the process of setting up a development environment and starting your first cloud apps:

Mahesh Mitkari
Configuring a Windows Azure Development Machine
blog.cognitioninfotech.com/2009/08/configuring-windows-azure-development.html

Jeff Widmer
Getting Started with Windows Azure: Part 1 - Setting up Your Development Environment
weblogs.asp.net/jeffwids/archive/2010/03/02/getting-started-with-windows-azure-part-1-setting-up-your-development-environment.aspx

David Sayed
Hosting Videos on Windows Azure
blogs.msdn.com/b/david_sayed/archive/2010/01/07/hosting-videos-on-windows-azure.aspx

Josh Holmes
Easy Setup for PHP on Azure Development
joshholmes.com/blog/2010/04/13/easy-setup-for-php-on-azure-development/

Visual Studio Magazine
Cloud Development in Visual Studio 2010
visualstudiomagazine.com/articles/2010/04/01/using-visual-studio-2010.aspx

TERRENCE DORSEY *is the technical editor of* MSDN Magazine. *You can read his blog at terrencedorsey.com or follow him on Twitter: @tpdorsey.*

Blazing-Fast **Grid Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Members Choice Winner The Code Project 2010
Awards 2009-10 ComponentSource #1 PRODUCT
Awards 2009-10 ComponentSource #1 PUBLISHER
Readers' Choice 2009 asp.netPRO Winner
Visual Studio Readers Choice 2009
tech·ed North America 2009 Attendees' Pick
Best of tech·ed North America 2009

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/GRIDS**
or **CALL US (818) 844-3383**

Devexpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

DINO ESPOSITO

Aspect-Oriented Programming, Interception and Unity 2.0

There's no doubt that object orientation is a mainstream programming paradigm, one that excels when it comes to breaking a system down into components and describing processes through components. The object-oriented (OO) paradigm also excels when you deal with the business-specific concerns of a component. However, the OO paradigm isn't as effective when it comes to dealing with cross-cutting concerns. In general, a cross-cutting concern is a concern that affects multiple components in a system.

To maximize the reuse of complex business logic code, you typically tend to design a hierarchy of classes around the core and primary business functions of the system. But what about other non-business-specific concerns that cross-cut through the hierarchy of classes? Where would you fit features such as caching, security and logging? Most likely, they end up being repeated in every affected object.

Not being a specific responsibility of a given component or family of components, a cross-cutting concern is an aspect of the system that must be dealt with at a different logical level, a level beyond application classes. For this reason, a distinct programming paradigm was defined years ago: aspect-oriented programming (AOP). Incidentally, the concept of AOP was developed at Xerox PARC laboratories in the 1990s. The team also developed the first (and still most popular) AOP language: AspectJ.

Even though almost everyone agrees on the benefits of AOP, it's still not widely implemented. In my opinion, the main reason for such a limited adoption is essentially the lack of proper tools. I'm pretty sure the day that AOP is (even only partially) supported natively by the Microsoft .NET Framework will represent a watershed in the history of AOP. Today, you can only do AOP in .NET using ad hoc frameworks.

The most powerful tool for AOP in .NET is PostSharp, which you can find at sharpcrafters.com. PostSharp offers a full AOP framework where you can experience all the key features of the theory of AOP. However, it should be noted that many dependency injection (DI) frameworks include some AOP capabilities.

For example, you find AOP capabilities in Spring.NET, Castle Windsor and—of course—Microsoft Unity. For relatively simple scenarios, such as tracing, caching and decoration of components in the application tier, the capabilities of DI frameworks usually do the trick. However, it's difficult to go with DI frameworks when it comes to domain objects and UI objects. A cross-cutting concern can certainly be seen as an external dependency, and DI techniques certainly allow you to inject external dependencies in a class.

The point is that DI will likely require ad hoc up-front design or a bit of refactoring. In other words, if you're using a DI framework already, then it's easy to bring in some AOP features. Conversely, if your system is DI-free, bringing in a DI framework may require quite a bit of work. This may not be always possible in a large project or during the update of a legacy system. With a classic AOP approach, instead, you wrap up any cross-cutting concerns in a new component called an aspect. In this article, I'll first give you a quick overview of the aspect-oriented paradigm and then move on to see the AOP-related capabilities you find in Unity 2.0.

A Quick Guide to AOP

An object-oriented programming (OOP) project is made up of a number of source files, each implementing one or more classes. The project also includes classes representing cross-cutting concerns such as logging or caching. All classes are processed by a compiler and produce executable code. In AOP, an aspect is a reusable component that encapsulates the behavior required by multiple classes within the project. The way in which aspects are actually processed depends on the AOP technology you're considering. In general, we can say that aspects aren't simply and directly processed by the compiler. An additional technology-specific tool is required to modify the executable code to take aspects into account. Let's briefly consider what happens with AspectJ—a Java AOP compiler that was the first AOP tool created.

With AspectJ, you use the Java programming language to write your classes and the AspectJ language to write aspects. AspectJ supports a custom syntax through which you indicate the expected behavior of the aspect. For example, a logging aspect might specify that it will log before and after a certain method is invoked. Aspects are in some way merged into the regular source code and produce an intermediate version of the source code that will then be compiled to an executable format. In AspectJ jargon, the component that preprocesses aspects and merges them with source code is known as the *weaver*. It produces an output that the compiler can render to an executable.

In summary, an aspect describes a reusable piece of code that you want to inject in existing classes without touching the source code of those classes. In other AOP frameworks (such as the .NET PostSharp framework), you won't find a weaver tool. However, the content of an aspect is always processed by the framework and results in some form of code injection.

Full-Featured **Reporting Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Members Choice Winner The Code Project 2010
Awards 2009-10 ComponentSource #1 Product
Awards 2009-10 ComponentSource #1 Publisher
Readers' Choice 2009 asp.netPRO Winner
Visual Studio Readers Choice 2009
tech-ed North America 2009 Attendees' Pick
Best of tech-ed North America 2009

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/REPORTING**
or **CALL US (818) 844-3383**

DevExpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

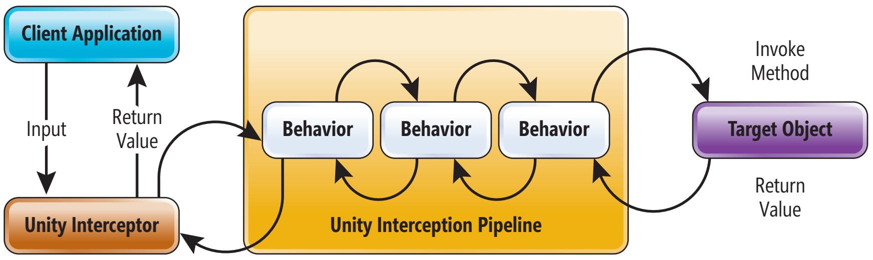


Figure 1 **Object Interception at Work in Unity 2.0**

Note that in this regard, *code* injection is different from *dependency* injection. Code injection refers to the ability of an AOP framework to insert calls to public endpoints in the aspect at specific points within the body of classes decorated with a given aspect. The PostSharp framework, for example, lets you write aspects as .NET attributes that you then attach to methods in your classes. PostSharp attributes are processed by the PostSharp compiler (we could even call it the weaver) in a post-build step. The net effect is that your code is enhanced to include some of the code in the attributes. But the injection points are resolved automatically, and all you do as a developer is write a self-contained aspect component and attach it to a public class method. It's easy to write and even easier to maintain the code.

To finish off this quick overview about AOP, let me present a few specific terms and clarify their intended meaning. A *join point* indicates a point in the source code of the target class where you want to inject the aspect's code. A *pointcut* represents a collection of join points. An *advice* refers to the code to inject in the target class. The code can be injected before, after and around the join point. An advice is associated with a pointcut. These terms come from the original definition of AOP and may not be reflected literally in the particular AOP framework you're using. It's recommended that you try to pick up the concepts behind the terms—the pillars of AOP—and then use this knowledge to better understand the details of a particular framework.

A Quick Guide to Unity 2.0

Unity is an application block available as part of the Microsoft Enterprise Library project, as well as a separate download. The Microsoft Enterprise Library is a collection of application blocks that addresses a bunch of cross-cutting concerns that characterize .NET application development—logging, caching, cryptography, exception handling and more. The latest version of the Enterprise Library is 5.0, released in April 2010 and coming with full support for Visual Studio 2010 (learn more about it at the patterns & practices Developer Center at msdn.microsoft.com/library/ff632023).

Unity is one of the Enterprise Library application blocks. Also available for Silverlight, Unity is essentially a DI container with additional support for an interception mechanism through which you can make your classes a bit more aspect-oriented.

Interception in Unity 2.0

The core idea of interception in Unity is enabling developers to customize the chain of calls that it takes to invoke a method on an object. In other words, the Unity interception mechanism captures calls being made to configured objects and customizes the behavior of the target objects by adding some extra code before, after or around the regular execution of methods. Interception is essentially an extremely flexible approach to add new behavior to an object at run time without touching its source code and without affecting the behavior of classes in the same inheritance path. Unity interception is a way to implement the Decorator pattern, which is a popular design pattern devised to extend the functionality of an object at run time as the object is used. A decorator is a container object that receives (and maintains a reference to) an instance of the target object and augments its capabilities toward the outside world.

The Interception mechanism in Unity 2.0 supports both instance and type interception. Furthermore, interception works regardless of the way in which the object is instantiated, whether the object is created through the Unity container or is a known instance. In the latter case, you can just use a different, completely standalone API. If you do so, however, you lose the configuration file support. **Figure 1** shows the architecture of the interception feature in Unity, detailing how it works on a particular object instance not resolved through the container. (The figure is just a slightly reworked version of a figure you find in the MSDN documentation.)

The interception subsystem is made of three key elements: the interceptor (or proxy); the behavior pipeline; and the behavior or aspect. At the two extremes of the subsystems you find the client application and the target object—that is, the object being assigned additional behaviors not hardcoded in its source code. Once the client application is configured to use the interception API of Unity on a given instance, any method invocation goes through a proxy object—the interceptor. This proxy object looks at the list of registered behaviors and invokes them through the internal pipeline. Each configured behavior is given a chance to run before or after the regular invocation of the object method. The proxy injects input data into the pipeline and receives any return value as initially generated by the target object and then further modified by behaviors.

Configuring Interception

The recommended way of using interception in Unity 2.0 is different from earlier versions, although the approach used in earlier versions is fully supported for backward compatibility. In Unity 2.0, interception is just a new extension you add to the container to describe how an object is actually resolved. Here's the code you need if you want to configure interception via fluent code:

```
var container = new UnityContainer();
container.AddNewExtension<Interception>();
```

The container needs to find information about types to intercept and behaviors to add. This information can be added either using fluent code or via configuration. I find configuration to be particularly flexible, as it lets you modify things without touching the application and without any new compile step. Let's go with the configuration-based approach.

Powerhouse **Analytics Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Members Choice Winner The Code Project 2010
Awards 2009-10 ComponentSource #1 Product
Awards 2009-10 ComponentSource #1 Publisher
Readers' Choice 2009 asp.netPRO Winner
Visual Studio Readers Choice 2009
tech·ed North America 2009 Attendees' Pick
Best of tech·ed North America 2009

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/ANALYTICS**
or **CALL US (818) 844-3383**

Devexpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

To begin with, you add the following in the configuration file:

```
<sectionExtension type="Microsoft.Practices.Unity.InterceptionExtension.
  Configuration.InterceptionConfigurationExtension,
  Microsoft.Practices.Unity.Interception.Configuration"/>
```

The purpose of this script is extending the configuration schema with new elements and aliases that are specific to the interception subsystem. Another due addition is the following:

```
<container>
  <extension type="Interception" />
  <register type="IBankAccount" mapTo="BankAccount">
    <interceptor type="InterfaceInterceptor" />
    <interceptionBehavior type="TraceBehavior" />
  </register>
</container>
```

To achieve the same thing using fluent code, you would call AddNewExtension<T> and RegisterType<T> on the container object.

Let's look at the configuration script more closely. The <extension> element adds interception to the container. Note that the "Interception" being used in the script is one of the aliases defined in the section extension. The interface type IBankAccount is mapped to the concrete type BankAccount (this is the classic job of a DI container) and associated with a particular type of interceptor. Unity offers two main types of interceptors: instance interceptors and type interceptors. Next month, I'll delve deeper into interceptors. For now, suffice it to say that an instance interceptor creates a proxy to filter incoming calls directed at the intercepted instance. Type interceptors, instead, just mock the type of the intercepted object and work on an instance of a derived type. (For more information on interceptors, see msdn.microsoft.com/library/ff660861(PandP.20).)

The interface interceptor is an instance interceptor limited to act as the proxy of only one interface on the object. The interface interceptor uses dynamic code generation to create the proxy class. The interception behavior element in the configuration indicates the external code you want to run around the intercepted object instance. The class TraceBehavior must be declaratively configured so the container can resolve it and any of its dependencies. You use the <register> element to tell the container about the TraceBehavior class and its expected constructor, as shown here:

```
<register type="TraceBehavior">
   <constructor>
     <param name="source" dependencyName="interception" />
   </constructor>
</register>
```

Figure 2 shows an excerpt from the TraceBehavior class.

A behavior class implements IInterceptionBehavior, which basically consists of the Invoke method. The Invoke method contains the entire logic you want to use for any method under the control of the interceptor. If you want to do something before the target method is called, you do it at the beginning of the method. When you want to yield to the target object—or, more precisely, to the next behavior registered in the pipeline—you call the getNext delegate provided by the framework. Finally, you can use any code you like to post-process the target object. The Invoke method needs to return a reference to the next element in the pipeline; if null is returned, then the chain is interrupted and further behaviors will never be invoked.

Configuration Flexibility

Interception and, more generally, AOP, address a number of interesting scenarios. For example, interception allows you to add responsibilities to individual objects without modifying the entire class, keeping the solution much more flexible than it would be with a decorator.

This article just scratched the surface of AOP applied to .NET. In the next few months I'll write more about interception in Unity and AOP in general. ■

Figure 2 **A Sample Unity Behavior**

```
class TraceBehavior : IInterceptionBehavior, IDisposable
{
  private TraceSource source;

  public TraceBehavior(TraceSource source)
  {
    if (source == null)
      throw new ArgumentNullException("source");

    this.source = source;
  }

  public IEnumerable<Type> GetRequiredInterfaces()
  {
    return Type.EmptyTypes;
  }

  public IMethodReturn Invoke(IMethodInvocation input,
    GetNextInterceptionBehaviorDelegate getNext)
  {
     // BEFORE the target method execution
     this.source.TraceInformation("Invoking {0}",
       input.MethodBase.ToString());

     // Yield to the next module in the pipeline
     var methodReturn = getNext().Invoke(input, getNext);

     // AFTER the target method execution
     if (methodReturn.Exception == null)
     {
       this.source.TraceInformation("Successfully finished {0}",
         input.MethodBase.ToString());
     }
     else
     {
       this.source.TraceInformation(
         "Finished {0} with exception {1}: {2}",
         input.MethodBase.ToString(),
         methodReturn.Exception.GetType().Name,
         methodReturn.Exception.Message);
     }

     this.source.Flush();
     return methodReturn;
   }

   public bool WillExecute
   {
     get { return true; }
   }

   public void Dispose()
   {
     this.source.Close();
   }
 }
```

DINO ESPOSITO *is the author of "Programming Microsoft ASP.NET MVC" from Microsoft Press (2010) and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. You can join his blog at weblogs.asp.net/despos.*

THANKS to the following technical expert for reviewing this article: *Chris Tavares*

Elegant **Scheduling Controls**
Optimized for **.NET**

Award-Winning Development Tools by DevExpress

Learn more and *download* your free evaluation copy today
Visit **DEVEXPRESS.COM/SCHEDULING**
or **CALL US (818) 844-3383**

DevExpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

JULIE LERMAN

Profiling Database Activity in the Entity Framework

In last month's Data Points column (msdn.microsoft.com/magazine/gg309181), I wrote about profiling the performance of the Entity Framework against SQL Azure. This month, I look at a different type of profiling—query profiling—to see what queries and commands are being executed on the database in response to queries and other data access activities in the Entity Framework.

One of the Entity Framework's prime capabilities is command generation for executing database queries as well as inserts, updates and deletes. This is a huge benefit for many developers, although there will always be an ongoing debate about the quality of SQL generated by object-relational mapping (ORM) tools versus SQL handwritten by experts (I won't be engaging in that debate in this article). And for the most part, the SQL that's generated is pretty good, especially considering that it has to be constructed dynamically in a generic way, regardless of how creative you get with your LINQ to Entities or Entity SQL query expressions.

Although a lot of attention was paid to improving command generation in the Entity Framework 4, it's still important to be aware of what's happening in your database. The quality of the generated store query is only part of the story. You may be writing code that creates extended database execution times or an unusual number of trips to the database. These are critical things to be aware of when profiling your application.

For the first few years of the Entity Framework's life, there was nothing available outside of database-profiling tools such as SQL Profiler, which, while informative, requires a lot of configuring and mining if you want to view the results in an easy-to-digest manner. Following are a variety of options for query profiling the Entity Framework beyond the database-profiling tools.

The Entity Framework ObjectContext.ToTraceString Method

The Entity Framework API (3.5 and 4) provides a single method for inspecting queries at run time, ToTraceString, which is useful but only provides information on a subset of the calls made to the database. ToTraceString is a method of ObjectQuery, so if you're writing a LINQ to Entities query, you must first cast the query to an ObjectQuery before calling ToTraceString. Here's an example:

```
var query = from c in context.Customers where c.CustomerID == 3 select c;
var objectQuery=query as System.Data.Objects.ObjectQuery;
Console.WriteLine(objectQuery.ToTraceString());
```

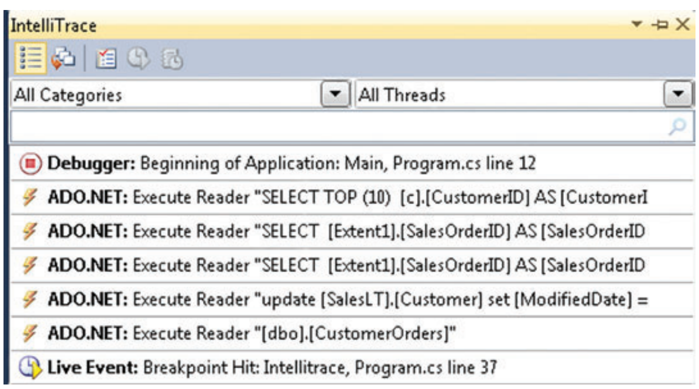
This outputs the following string:

```
SELECT
[Extent1].[CustomerID] AS [CustomerID],
[Extent1].[Title] AS [Title],
[Extent1].[FirstName] AS [FirstName],
[Extent1].[MiddleName] AS [MiddleName],
[Extent1].[LastName] AS [LastName],
[Extent1].[Suffix] AS [Suffix],
[Extent1].[CompanyName] AS [CompanyName],
[Extent1].[SalesPerson] AS [SalesPerson],
[Extent1].[EmailAddress] AS [EmailAddress],
[Extent1].[Phone] AS [Phone],
[Extent1].[ModifiedDate] AS [ModifiedDate],
[Extent1].[TimeStamp] AS [TimeStamp]
FROM [SalesLT].[Customer] AS [Extent1]
WHERE 3 = [Extent1].[CustomerID]
```

Notice that the code sample doesn't execute the query. Nor does ToTraceString, which causes the query to be processed (transformed to a store query) by the Entity Framework with help from the database provider, System.Data.SqlClient, but doesn't force the query to be executed on the database.

You can use ToTraceString only with explicitly defined queries. Therefore, you can't use it to see queries executed as a result of deferred loading with the Load method or lazy loading. Nor can you use it to inspect activity such as inserts, updates, deletes or stored procedure execution.

Finally, it's important to note that you can't easily get the results of ToTraceString into a debug process, by, for example, creating a debugger visualizer. That would require ObjectQuery to be serializable, which it's not.



```
Debugger: Beginning of Application: Main, Program.cs line 12
ADO.NET: Execute Reader "SELECT TOP (10) [c].[CustomerID] AS [CustomerI
ADO.NET: Execute Reader "SELECT [Extent1].[SalesOrderID] AS [SalesOrderID
ADO.NET: Execute Reader "SELECT [Extent1].[SalesOrderID] AS [SalesOrderID
ADO.NET: Execute Reader "update [SalesLT].[Customer] set [ModifiedDate] =
ADO.NET: Execute Reader "[dbo].[CustomerOrders]"
Live Event: Breakpoint Hit: Intellitrace, Program.cs line 37
```

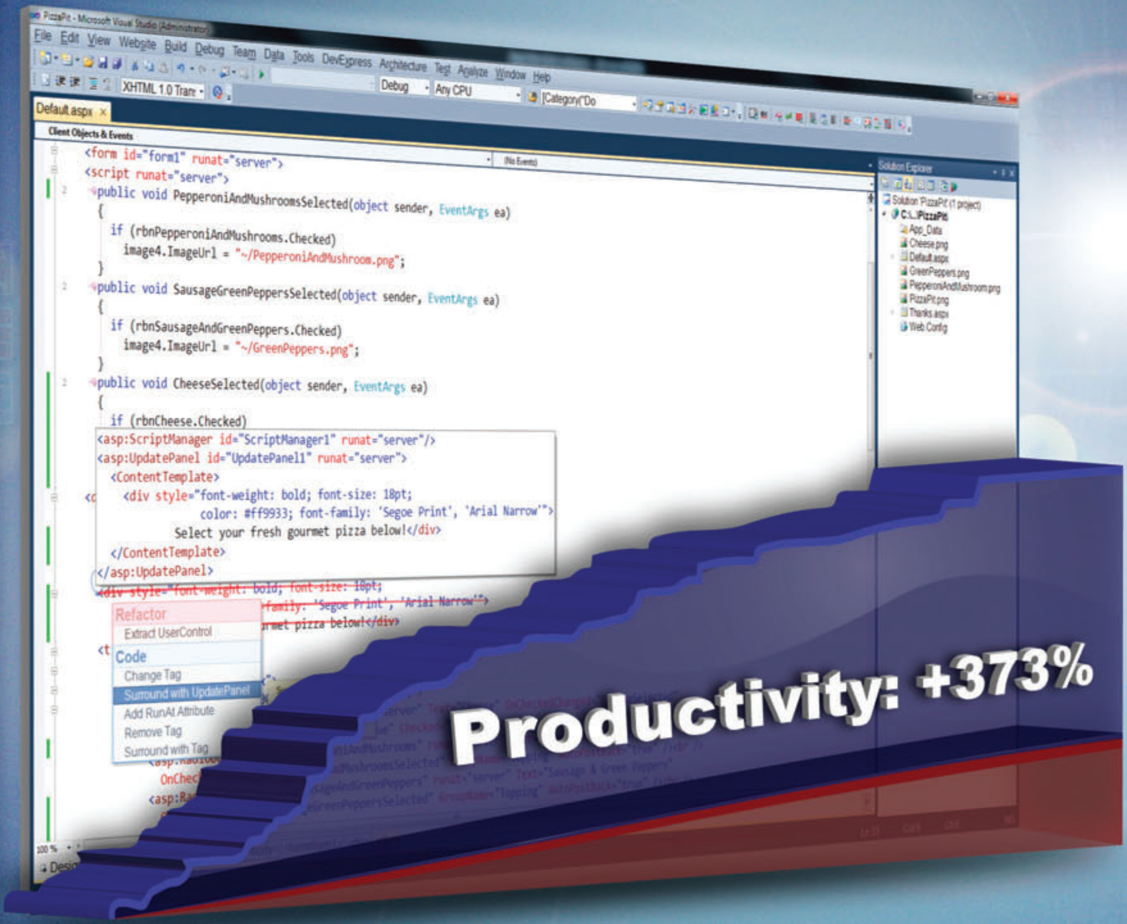
Figure 1 **A Series of Database Commands Displayed in the Visual Studio IntelliTrace Display**

Profiling with Visual Studio 2010 IntelliTrace

IntelliTrace is available in Visual Studio 2010 Ultimate but not the lesser versions. IntelliTrace captures database activity, including that which is triggered by the Entity Framework, but it doesn't display the parameter values that were sent with the command.

Rocket Fuel for **Visual Studio**®

(Superior code, up to **3X** faster)



```
<form id="form1" runat="server">
<script runat="server">
public void PepperoniAndMushroomsSelected(object sender, EventArgs ea)
  {
    if (rbnPepperoniAndMushrooms.Checked)
      image4.ImageUrl = "~/PepperoniAndMushroom.png";
  }
public void SausageGreenPeppersSelected(object sender, EventArgs ea)
  {
    if (rbnSausageAndGreenPeppers.Checked)
      image4.ImageUrl = "~/GreenPeppers.png";
  }
public void CheeseSelected(object sender, EventArgs ea)
  {
    if (rbnCheese.Checked)
```

```
<asp:ScriptManager id="ScriptManager1" runat="server"/>
<asp:UpdatePanel id="UpdatePanel1" runat="server">
  <ContentTemplate>
    <div style="font-weight: bold; font-size: 18pt;
            color: #ff9933; font-family: 'Segoe Print', 'Arial Narrow'">
        Select your fresh gourmet pizza below!</div>
  </ContentTemplate>
</asp:UpdatePanel>
```

See CodeRush in action: DevExpress.com/373

Learn more and *download* your free evaluation copy today

Visit **DEVEXPRESS.COM/CODERUSH**

or **CALL US (818) 844-3383**

DevExpress™

Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS

IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS

Following is some code that performs the following tasks:

1. Executes a query of 10 customers
2. Lazy loads the orders for the first returned customer
3. Disables lazy loading
4. Explicitly loads the orders for the second returned customer
5. Modifies a customer
6. Sends the changes to the database with the SaveChanges method
7. Executes a function that I've mapped to a stored procedure in an Entity Data Model

```
var query = from c in context.Customers select c;
var custList = query.Take(10).ToList();

Customer custFirst = custList[0];
int orderCount = custFirst.Orders.Count;

context.ContextOptions.LazyLoadingEnabled=false;
Customer custSecond = custList[1];
custSecond.Orders.Load();

custSecond.ModifiedDate = DateTime.Now;
context.SaveChanges();

ObjectResult<PartialOrderDetails> orders=
  context.GetCustomerOrdersForId(custList[2].CustomerID);
```

When run, this code will force the execution of three SELECT statements, an UPDATE statement and then an Execute command for the stored procedure in the database.

Looking at the IntelliTrace screenshot in **Figure 1**, you can see all five of these commands.

However, expanding one of these items, as shown in **Figure 2**, shows that the command is there but the parameters are not.

Therefore, if you want to see the database activity including parameters, you'll need to use some type of external profiler.

The EFTracingProvider on MSDN Code Gallery

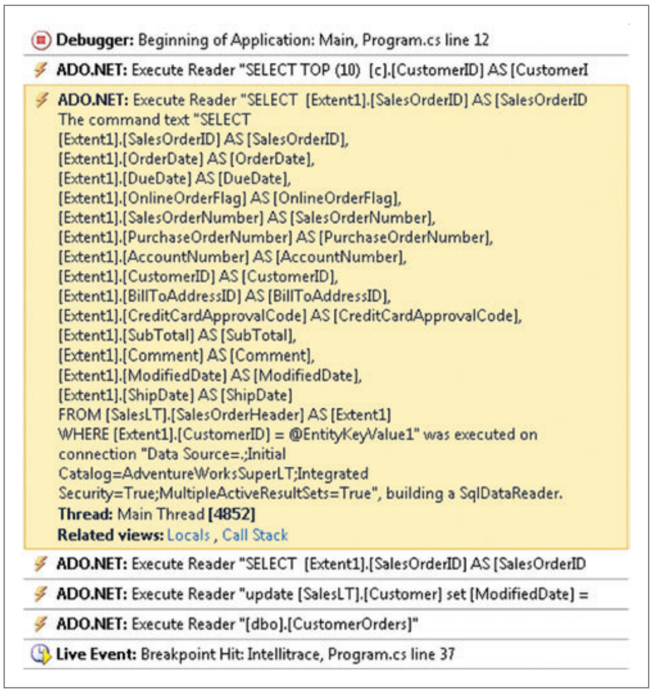
Jarek Kowalski wrote the EFTracingProvider when he was on the Entity Framework team at Microsoft. There's a version for the Microsoft .NET Framework 3.5 and one for the .NET Framework 4.

To use the EFTracingProvider, you'll need to build a wrapper around the ObjectContext class, AWEntities, and use that in place of AWEntities. This extended class provides tracing methods, such as Log, that you can use to log context activities. There's an example of the required class wrapper included with the EFTracingProvider download. You'll also find the relevant code in the download for this article (code.msdn.microsoft.com/mag201012DataPoints). Additionally, you need to add two DbProviderFactories settings in the application's config file. With all of this in place, you can then instantiate the extended context and begin logging.

Here's an example that creates a text file to capture the log events and then uses the TracingProvider.Log method to log all of the activities:

```
using (TextWriter logFile = File.CreateText("sqllogfile.txt"))
{
  using (var context = new ExtendedAWEntities())
  {
    context.Log = logFile;
    var query = from c in context.Customers select c;
    var custList = query.Take(10).ToList();
  }
  Console.WriteLine(File.ReadAllText("sqllogfile.txt"));
}
```

Using the TracingProvider wrapper and the ExtendedAWEntities context class, I reran the same set of code as in the previous IntelliTrace example.



```
Debugger: Beginning of Application: Main, Program.cs line 12
ADO.NET: Execute Reader "SELECT TOP (10) [c].[CustomerID] AS [CustomerI
ADO.NET: Execute Reader "SELECT [Extent1].[SalesOrderID] AS [SalesOrderID
The command text "SELECT
[Extent1].[SalesOrderID] AS [SalesOrderID],
[Extent1].[OrderDate] AS [OrderDate],
[Extent1].[DueDate] AS [DueDate],
[Extent1].[OnlineOrderFlag] AS [OnlineOrderFlag],
[Extent1].[SalesOrderNumber] AS [SalesOrderNumber],
[Extent1].[PurchaseOrderNumber] AS [PurchaseOrderNumber],
[Extent1].[AccountNumber] AS [AccountNumber],
[Extent1].[CustomerID] AS [CustomerID],
[Extent1].[BillToAddressID] AS [BillToAddressID],
[Extent1].[CreditCardApprovalCode] AS [CreditCardApprovalCode],
[Extent1].[SubTotal] AS [SubTotal],
[Extent1].[Comment] AS [Comment],
[Extent1].[ModifiedDate] AS [ModifiedDate],
[Extent1].[ShipDate] AS [ShipDate]
FROM [SalesLT].[SalesOrderHeader] AS [Extent1]
WHERE [Extent1].[CustomerID] = @EntityKeyValue1" was executed on
connection "Data Source=.;Initial
Catalog=AdventureWorksSuperLT;Integrated
Security=True;MultipleActiveResultSets=True", building a SqlDataReader.
Thread: Main Thread [4852]
Related views: Locals , Call Stack
ADO.NET: Execute Reader "SELECT [Extent1].[SalesOrderID] AS [SalesOrderID
ADO.NET: Execute Reader "update [SalesLT].[Customer] set [ModifiedDate] =
ADO.NET: Execute Reader "[dbo].[CustomerOrders]"
Live Event: Breakpoint Hit: Intellitrace, Program.cs line 37
```

Figure 2 **A Detailed Select Statement Collected by the Visual Studio 2010 IntelliTrace Feature**

All five database commands were logged and each command was logged with its relevant parameters.

Figure 3, as an example, shows the command sent as a result of the lazy load where the value of the EntityKeyValue1 parameter is specified after the command is listed.

The EFTracingProvider is simple to implement and provides you with all of the database commands generated by your Entity Framework code as raw text. You can also subscribe to the tracing events: CommandExecuting, CommandFinished and CommandFailed on the context. These give you access to the raw DbCommand just before and after it gets executed so you can analyze or log additional details.

You can download the EFTracingProvider—along with its companion, the EFCachingProvider, and a sample solution, the EFProviderWrapperDemo, which explores all of these features—for free from the MSDN Code Gallery (code.msdn.microsoft.com/EFProviderWrappers).

Figure 3 **A Command Captured by the EFTracingProvider**

```
SELECT
[Extent1].[SalesOrderID] AS [SalesOrderID],
[Extent1].[OrderDate] AS [OrderDate],
[Extent1].[DueDate] AS [DueDate],
[Extent1].[OnlineOrderFlag] AS [OnlineOrderFlag],
[Extent1].[SalesOrderNumber] AS [SalesOrderNumber],
[Extent1].[PurchaseOrderNumber] AS [PurchaseOrderNumber],
[Extent1].[AccountNumber] AS [AccountNumber],
[Extent1].[CustomerID] AS [CustomerID],
[Extent1].[BillToAddressID] AS [BillToAddressID],
[Extent1].[CreditCardApprovalCode] AS [CreditCardApprovalCode],
[Extent1].[SubTotal] AS [SubTotal],
[Extent1].[Comment] AS [Comment],
[Extent1].[ModifiedDate] AS [ModifiedDate],
[Extent1].[ShipDate] AS [ShipDate]
FROM [SalesLT].[SalesOrderHeader] AS [Extent1]
WHERE [Extent1].[CustomerID] = @EntityKeyValue1
-- EntityKeyValue1 (dbtype=Int32, size=0, direction=Input) = 1
```

FREE ASP.NET **Controls**

(Yes, they are absolutely free)

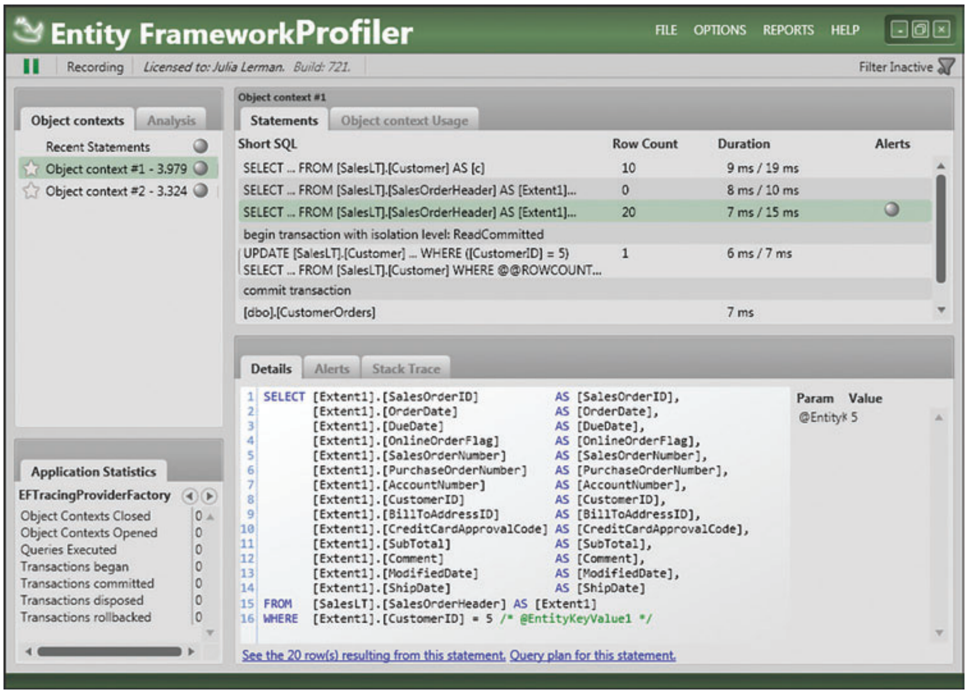
Over **20 Free** ASP.NET WebForms Controls Including:

CALLBACK PANEL | MENU | CLOUD CONTROL | NAVBAR | NEWS TICKER
PAGER | POPUP CONTROL | UPLOAD CONTROL | DATA VIEW

Download and Start Using Today
Visit **DEVEXPRESS.COM/FREEASP**

Devexpress™
Download • Compare • Decide!

PRESENTATION CONTROLS | REPORTING CONTROLS
IDE PRODUCTIVITY TOOLS | BUSINESS APP FRAMEWORKS



Short SQL	Row Count	Duration	Alerts
SELECT ... FROM [SalesLT].[Customer] AS [c]	10	9 ms / 19 ms	
SELECT ... FROM [SalesLT].[SalesOrderHeader] AS [Extent1]...	0	8 ms / 10 ms	
SELECT ... FROM [SalesLT].[SalesOrderHeader] AS [Extent1]...	20	7 ms / 15 ms	●
begin transaction with isolation level: ReadCommitted			
UPDATE [SalesLT].[Customer] ... WHERE ([CustomerID] = 5) SELECT ... FROM [SalesLT].[Customer] WHERE @@ROWCOUNT...	1	6 ms / 7 ms	
commit transaction			
[dbo].[CustomerOrders]		7 ms	

```
SELECT [Extent1].[SalesOrderID]             AS [SalesOrderID],
       [Extent1].[OrderDate]                AS [OrderDate],
       [Extent1].[DueDate]                  AS [DueDate],
       [Extent1].[OnlineOrderFlag]          AS [OnlineOrderFlag],
       [Extent1].[SalesOrderNumber]         AS [SalesOrderNumber],
       [Extent1].[PurchaseOrderNumber]      AS [PurchaseOrderNumber],
       [Extent1].[AccountNumber]            AS [AccountNumber],
       [Extent1].[CustomerID]               AS [CustomerID],
       [Extent1].[BillToAddressID]          AS [BillToAddressID],
       [Extent1].[CreditCardApprovalCode]   AS [CreditCardApprovalCode],
       [Extent1].[SubTotal]                 AS [SubTotal],
       [Extent1].[Comment]                  AS [Comment],
       [Extent1].[ModifiedDate]             AS [ModifiedDate],
       [Extent1].[ShipDate]                 AS [ShipDate]
FROM   [SalesLT].[SalesOrderHeader] AS [Extent1]
WHERE  [Extent1].[CustomerID] = 5 /* @EntityKeyValue1 */
```

Param	Value
@Entityk	5

See the 20 row(s) resulting from this statement. Query plan for this statement.

Application Statistics

EFTracingProviderFactory	
Object Contexts Closed	0
Object Contexts Opened	0
Queries Executed	0
Transactions began	0
Transactions committed	0
Transactions disposed	0
Transactions rollbacked	0

Figure 4 **The EF Prof Query Profiler UI**

Third-Party Profilers

You may, however, wish to go beyond the raw text of the EF-TracingProvider's log file. You can leverage and learn from the code in those log files or take advantage of two tools that have already done the work for you. There are two third-party tools for profiling Entity Framework queries: Hibernating Rhinos Entity Framework Profiler and Huagati Query Profiler.

Additionally, LINQPad, which is focused on allowing you to test query expressions outside of your application, displays SQL for the expressions you're executing. Although this is an indispensable tool for anyone writing LINQ against a large variety of providers, it doesn't allow you to profile the queries generated by your application, and therefore I won't explore it further in this column.

Entity Framework Profiler (EF Prof) is part of the Hibernating Rhinos UberProf family of profilers (hibernatingrhinos.com/products/UberProf). There are also profilers for nHibernate, Hibernate and LINQ to SQL. A fifth profiler, LLBLGen Pro, was in beta at the time of writing. EF Prof combines the existing intellectual property derived from the other UberProf tools with some ideas gleaned from the EFTracingProvider. At its simplest, you can add a single line of code to your application to enable it to talk to EF Prof's engine and have the results reported in the EF Prof client application:

```
HibernatingRhinos.Profiler.
  Appender.EntityFramework.
  EntityFrameworkProfiler.Initialize
```

Database activity is grouped by ObjectContext instance. In **Figure 4**, you can see that there are two ObjectContext instances displayed—that's because I ran my example code twice.

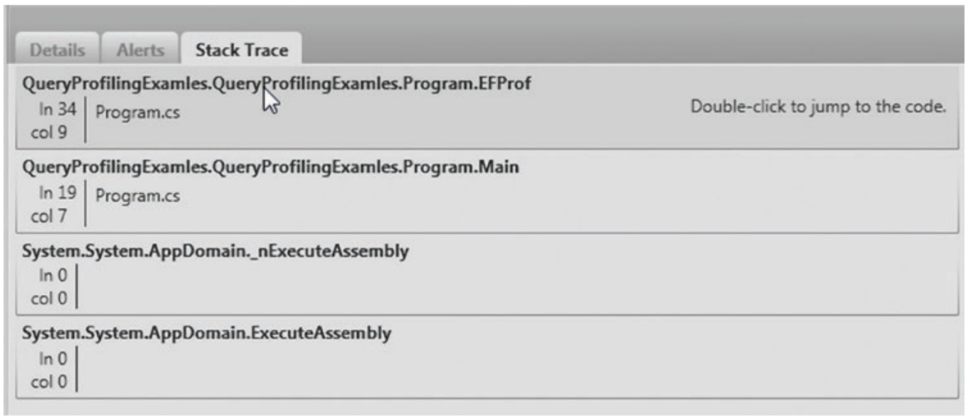
Also in **Figure 4**, on the right, you can see a preview of each of the database calls for the selected context instance. It appears that there's an extra SELECT being called after the UPDATE command. This is, in fact, part of the command that's sent with SaveChanges as the Entity Framework is ensuring that the Customer row's updated Time-Stamp field is returned to the customer instance.

As you highlight a SQL statement in the UI, you can see the complete SQL in the lower screen along with a reference to the fact that the value—5 in this case—was passed in as a parameter, @EntityKeyValue1.

EF Prof also allows you to see the resultant rows from the query, and even the database query plan. The Stack Trace tab, shown in **Figure 5**, lets you see how the application came to execute a particular command and even lets you jump directly to that line of code in Visual Studio.

EF Prof is able to capture all of an application's Entity Framework activity and presents it in an easy-to-navigate UI along with some great bells and whistles—such as the query plan view—and links back to the executing code. A standard license to EF Prof is $305 with discounts for multiple licenses and a subscription plan. EF Prof works with any of the Entity Framework data providers and therefore isn't limited to SQL Server. It works with the .NET Framework versions 3.5 and 4.

Huagati Query Profiler, originally called L2S Profiler, was updated in November to add support for the Entity Framework 4. You can also use it to profile LINQ to SQL and LLBLGen Pro, but it currently only works with SQL Server.



QueryProfilingExamles.QueryProfilingExamles.Program.EFProf
ln 34 col 9 | Program.cs

QueryProfilingExamles.QueryProfilingExamles.Program.Main
ln 19 col 7 | Program.cs

System.System.AppDomain._nExecuteAssembly
ln 0 col 0

System.System.AppDomain.ExecuteAssembly
ln 0 col 0

Figure 5 **The EF Prof Stack Trace Lets You Jump to the Code that Executed the Selected Database Command**

Microsoft Dynamics® AX

TECHNICAL CONFERENCE 2011

Learn | Inspire | Innovate

January 17 - 20, 2011 • Redmond, WA

The Microsoft Dynamics AX team has been hard at work developing our most innovative release to date. We are inviting our partners and customers to Redmond to experience the product pre-release.

Microsoft Dynamics AX provides industry solutions for customers by bringing together a breadth of Microsoft products and technologies, including:

Microsoft® Office | Microsoft® SharePoint® Server 2010 | Microsoft® Visual Studio® 2010 | Windows Server® 2008 R2 | Microsoft® SQL Server® 2008 R2 | Microsoft® .NET

Experience How. Get Involved.

Find out more at **http://www.microsoft.com/dynamics/daxconf2011/MSDN**

Want to get involved? Email: **daxconf@microsoft.com.**

Microsoft®

Figure 6 **The Partial Class to Use with Huagati Query Profiler**

```
partial class AWEntities
{
  private HuagatiEFProfiler.EFProfiler _profiler = null;
  public AWEntities(bool enableProfiling)
    : this(enableProfiling, "name=AWEntities")
  {
  }
  public AWEntities(bool enableProfiling, string connectionString)
    : this(EFProfiler.GetConnection(enableProfiling, connectionString))
  {
    string profilerOutput =
      System.IO.Path.Combine(System.Environment.GetFolderPath(
        Environment.SpecialFolder.Personal),
      @"EFProfiler\Samples");
    _profiler=new HuagatiEFProfiler.EFProfiler(this, profilerOutput, null,
      HuagatiEFProfiler.ExecutionPlanMode.Actual, false);
   _profiler.LogError += EFProfiler_LogError;
  }
}
```

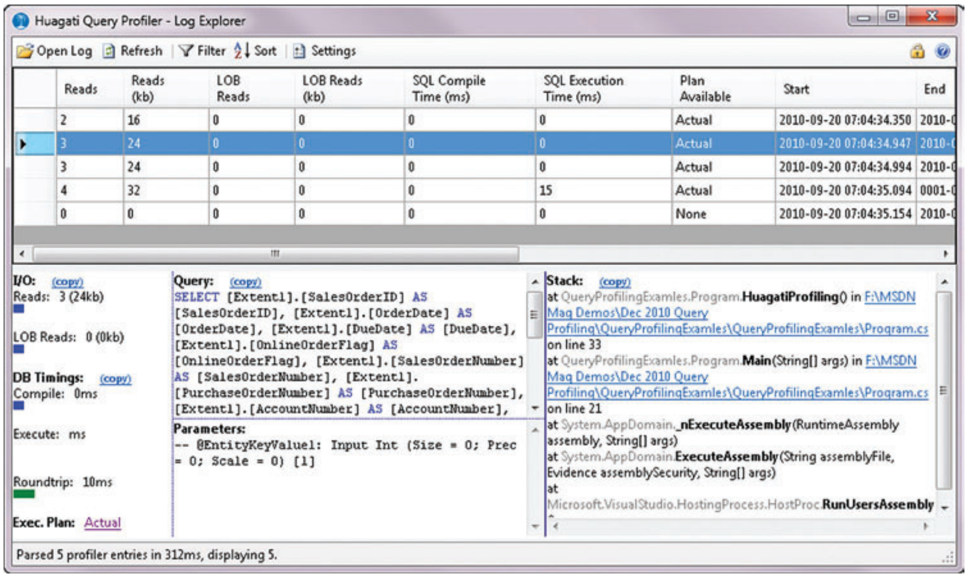
Implementing the Query Profiler is a matter of referencing the profiler's assembly (Huagati.EFProfiler.dll) in your application and adding two new constructors, plus some additional logic to your ObjectContext class in a partial class. **Figure 6** shows the partial class I've created for my AWEntities class.

The EFProfiler.GetConnection method hooks into the database to track its activity. You can learn more about various settings you can use when instantiating the EFProfiler on the Huagati Web site.

The profiler collects its data and outputs it to a file in the designated folder. You can then open that file up in the profiler's Log Explorer, shown in **Figure 7**.

As you can see in **Figure 7**, all five database requests were collected. The Update command is combined with its SELECT command to return the TimeStamp, which is exactly how the command is sent to the database.

The Log Explorer shown in **Figure 7** displays the relevant lines from the SQL Server SQL Profiler data. As with EF Prof, you can see the query with its parameters, link back to the relevant code lines in your application from the Stack view, view the execution plan and see some statistics about the queries.



	Reads	Reads (kb)	LOB Reads	LOB Reads (kb)	SQL Compile Time (ms)	SQL Execution Time (ms)	Plan Available	Start	End
	2	16	0	0	0	0	Actual	2010-09-20 07:04:34.350	2010-
▶	3	24	0	0	0	0	Actual	2010-09-20 07:04:34.947	2010-
	3	24	0	0	0	0	Actual	2010-09-20 07:04:34.994	2010-
	4	32	0	0	0	15	Actual	2010-09-20 07:04:35.094	0001-
	0	0	0	0	0	0	None	2010-09-20 07:04:35.154	2010-

```
I/O: (copy)
Reads: 3 (24kb)
LOB Reads: 0 (0kb)
DB Timings: (copy)
Compile: 0ms
Execute: ms
Roundtrip: 10ms
Exec. Plan: Actual
```

```
Query: (copy)
SELECT [Extent1].[SalesOrderID] AS
[SalesOrderID], [Extent1].[OrderDate] AS
[OrderDate], [Extent1].[DueDate] AS [DueDate],
[Extent1].[OnlineOrderFlag] AS
[OnlineOrderFlag], [Extent1].[SalesOrderNumber]
AS [SalesOrderNumber], [Extent1].
[PurchaseOrderNumber] AS [PurchaseOrderNumber],
[Extent1].[AccountNumber] AS [AccountNumber],
Parameters:
-- @EntityKeyValue1: Input Int (Size = 0; Prec
= 0; Scale = 0) [1]
```

```
Stack: (copy)
at QueryProfilingExamles.Program.HuagatiProfiling() in F:\MSDN
Maq Demos\Dec 2010 Query
Profiling\QueryProfilingExamles\QueryProfilingExamles\Program.cs
on line 33
at QueryProfilingExamles.Program.Main(String[] args) in F:\MSDN
Maq Demos\Dec 2010 Query
Profiling\QueryProfilingExamles\QueryProfilingExamles\Program.cs
on line 21
at System.AppDomain._nExecuteAssembly(RuntimeAssembly
assembly, String[] args)
at System.AppDomain.ExecuteAssembly(String assemblyFile,
Evidence assemblySecurity, String[] args)
at
Microsoft.VisualStudio.HostingProcess.HostProc.RunUsersAssembly
```

```
Parsed 5 profiler entries in 312ms, displaying 5.
```

Figure 7 **The Huagati Entity Framework Query Profiler UI**

The database activity for each context instance is stored in a separate log file, so the Log Explorer will only display one set of commands at a time. The Settings allow you to color-code alerts highlighting unusual activity levels such as noticeably or even alarmingly long execution times.

The Query Profiler UI isn't as slick as the EF Prof UI, and you need to make a slightly greater investment in your code (adding logic to each ObjectContext in your application). But the components are distributable, which means that you can collect profiler information for apps running in your client's environment. Also, it doesn't have as many analysis options as EF Prof. But the $20 Standard and $40 Professional sticker price (which includes all three profilers), may make up for these differences for many developers. Keep in mind that the Huagati Entity Framework Profiler was still in beta when I did my exploration and that it only works with SQL Server, as opposed to the ability of EF Prof to work with any of the available ADO.NET Data Providers that support the Entity Framework.

An introduction to Hugati's Entity Framework support can be found at tinyurl.com/26cfful. At the end of that blog post you'll find a link to download the beta version, 1.31.

You shouldn't take your database for granted when profiling your application.

The Right Tool for the Job

I'm a big believer of using the right tool for the job, and that it's wasteful to try to squeeze functionality out of the Visual Studio 2010 box when there are other great tools available. In this column, you've seen an array of tools built into the Entity Framework APIs and Visual Studio 2010, an extension that will provide you with raw data and two third-party tools that perform not only the task of data collection, but presentation as well. Whichever one of these paths you choose, even if you simply use SQL Profiler, you shouldn't take your database for granted when profiling your application. ■

JULIE LERMAN *is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other Microsoft .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of the highly acclaimed book, "Programming Entity Framework" (O'Reilly Media, 2010). Follow her on Twitter.com: julielerman.*

THANKS to the following technical expert for reviewing this article: *Jarek Kowalski*

version 17

LEADTOOLS®

The World Leader in Imaging SDKs

LEAD TECHNOLOGIES INCORPORATED

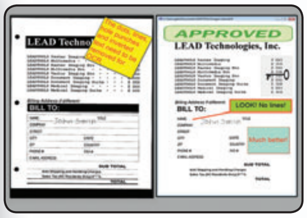
Silverlight, .NET, WPF, WCF, WF, C API, C++ Class Lib, COM & more!

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.**

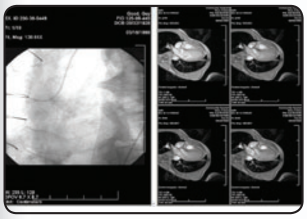
LEADTOOLS provides developers easy access to decades of expertise in color, grayscale, document, medical, vector and multimedia imaging development. Install LEADTOOLS to eliminate months of research and programming time while maintaining high levels of quality, performance and functionality.

•**Silverlight:** 100% pure Silverlight 3 and 4 Imaging SDK.
•**Image Formats & Compression:** Supports 150+ image formats and compressions including TIFF, EXIF, PDF, JPEG2000, JBIG2 and CCITT G3/G4.
•**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 with PDF, PDF/A, XPS, DOC, XML and Text output.
•**Forms Recognition & Processing:** Automatically identify and classify forms and extract user filled data.
•**Barcode:** Auto-detect, read and write 1D and 2D barcodes for multithreaded 32 & 64 bit development.
•**Document Cleanup/Preprocessing:** Auto-deskew, despeckle, hole punch, line and border removal, inverted text correction and more for optimum results in OCR and Barcode recognition.
•**PDF & PDF/A:** Read, write and view searchable PDF with text, images, bookmarks 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.
•**PACS Workstation Framework:** Set of .NET PACS components that can be used to build a full featured PACS Workstation application.
•**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 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.
•**3D:** Construct 3D volumes from 2D DICOM medical images and visualize with a variety of methods including MIP, MinIP, MRP, VRT and SSD.
•**Scanning:** TWAIN & WIA (32 & 64-bit), auto-detect optimum driver settings for high speed scanning.
•**DVD:** Play, create, convert and burn DVD images.
•**MPEG Transport Stream:** With DVR for UDP and TCP/IP streams & auto-live support.
•**Multimedia:** Capture, play, stream and convert MPEG, AVI, WMV, MP4, MP3, OGG, ISO, DVD and more.

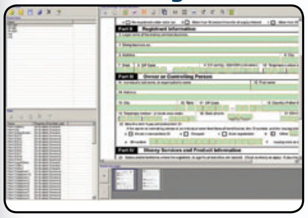
Document


LEAD Technologies, Inc.
APPROVED
BILL TO:
LOOK! No lines!

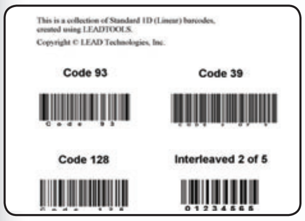
DICOM Medical



Form Recognition & Processing



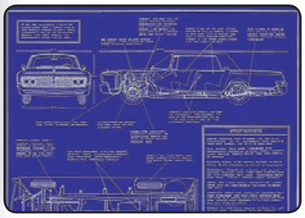
Barcode


This is a collection of Standard 1D (Linear) barcodes, created using LEADTOOLS.
Copyright © LEAD Technologies, Inc.

Code 93 | Code 39
Code 128 | Interleaved 2 of 5

Multimedia

Vector



Free 60 Day Evaluation!
www.leadtools.com/msdn
(800) 637-1840

Microsoft Silverlight Microsoft Visual Studio

Your Data in a Whole New Light

ComponentArt

Data Visualization for .NET

ComponentArt's latest Data Visualization technology allows you to present, navigate and visualize your data like never before. Visit our website and experience a whole new level of interactivity, flexibility and performance.

Charting Gauges Maps GridView TimeNavigator CalcEngine

www.componentart.com

"How do I get my data presented like this?" That is typically the first question we get from business users who have seen the interactive power of our technology. Rest assured, ComponentArt dashboards work with any data source. Contact us today to learn how you can start presenting your data in a whole new light.

ComponentArt
Build Something Amazing

FORECAST: CLOUDY

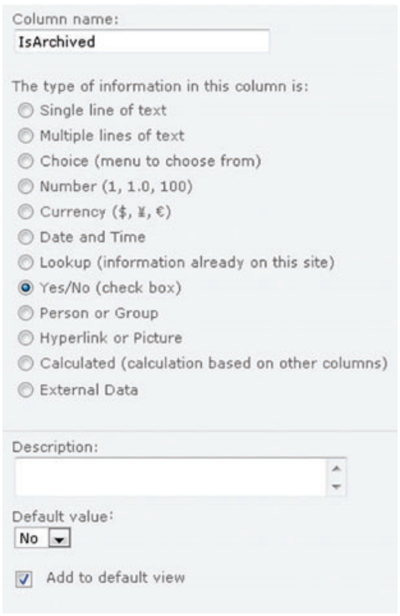
JOSEPH FULTZ AND SHAD PHILLIPS

Pushing Content from SharePoint to Windows Azure Storage

I have a coauthor this month, as colleague Shad Phillips helped me out with a recent project where I was working with a customer on a proof of concept that used SharePoint 2010 as an application platform. On the side, one of the customer's staffers asked me if I could think of a reasonable way to take approved content from SharePoint and publish it, making it available to people outside of the corporate network.

The customer's current infrastructure didn't support external content (downloadable documents and videos). Having done a lot of work with Windows Azure, I immediately thought that it would be pretty simple to incorporate pushing the content to Windows Azure Storage as part of the workflow and then, depending on need, making it publicly available or providing lease-based access for restricted content.

With that in mind, I talked with my colleague, Shad, who had previously solved a similar problem where he had implemented a sample method to archive SharePoint documents from the library to Windows Azure Storage. Although the intent of that solution is different from my goal, the mechanics are the same. This month, Shad and I will walk through a sample implementation that pushes content from SharePoint to Windows Azure Storage, and cover a little bit about lease-access control to the files.



Column name: IsArchived

The type of information in this column is:
- Single line of text
- Multiple lines of text
- Choice (menu to choose from)
- Number (1, 1.0, 100)
- Currency ($, ¥, €)
- Date and Time
- Lookup (information already on this site)
- ● Yes/No (check box)
- Person or Group
- Hyperlink or Picture
- Calculated (calculation based on other columns)
- External Data

Description:

Default value: No

✓ Add to default view

Figure 1 **Column Settings on the Team Site Template**

Scenario and Setup

Specifically, we developed a custom feature that enables a user to selectively push a document from SharePoint to Windows Azure Storage. For some inexplicable reason, users don't typically like it when their documents are moved and links aren't provided to find them, so we left a link in the document library to the cloud location that behaves the same as it would if the document were at a non-cloud location.

Needed software:

- Visual Studio 2010
- Microsoft SharePoint 2010
- Windows Azure SDK
- Windows Azure Development Storage Service

SharePoint 2010 Site and Document Library Configuration

For this scenario, we created a SharePoint site using the Team Site template. In the Shared Documents library, we created a column that allows us to flag an item as archived to Windows Azure. This is done through the Library Settings accessible via the Ribbon. Once in Library Settings, we created a column with the properties illustrated in **Figure 1**.

In Advanced Settings, we also selected "Yes" for the "Allow the management of content types" setting.

We used this as part of the Content Type that we named Link to a Document. Next, we created instances of this Content Type as a means to link to the archived document, as shown in **Figure 2**.

Once the column and the content type were added to the document library, we uploaded a sample Word document named Services SOW.docx.

SharePoint 2010 Web.config

In order to connect to the cloud, we needed to get the settings that are required to connect with Windows Azure. In this case, we used development storage and added the keys to the <appSettings> element in the web.config, as shown in **Figure 3**.

SharePoint Project

Fortunately, for SharePoint 2010 using Visual Studio 2010, it's a nice developer experience to create, debug and publish new features. We created a SharePoint Feature (see msdn.microsoft.com/library/bb861828(office.12) for more information on this), which adds a custom action to the items' action drop-down menus in the document library. The user will click it to use the archive feature.

We started by creating a solution named MSSAzureArchive using the Empty SharePoint Project template (see **Figure 4**).

Next, we specified the site and security level for debugging. We decided to choose "Deploy as a farm solution" because the code will need to make external calls, which the sandboxed solution

Code download available at code.msdn.microsoft.com/mag201012Cloudy.

/n software

We didn't invent the Internet...

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

applications

powered by

connectivity

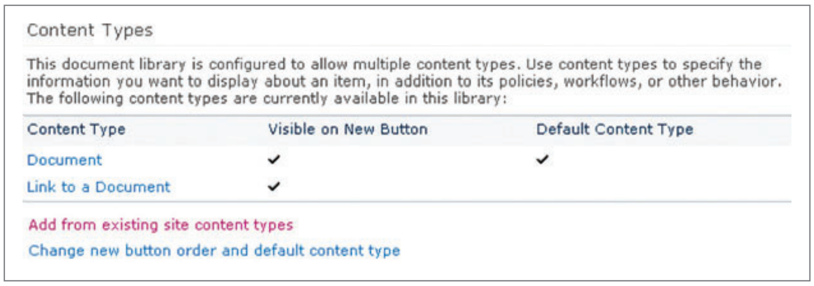
powered by

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

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



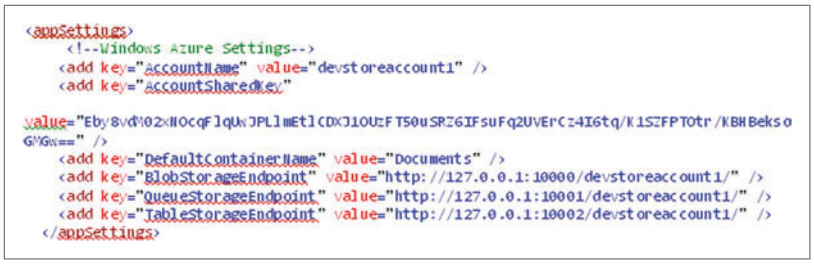
Content Types

This document library is configured to allow multiple content types. Use content types to specify the information you want to display about an item, in addition to its policies, workflows, or other behavior. The following content types are currently available in this library:

Content Type	Visible on New Button	Default Content Type
Document	✓	✓
Link to a Document	✓	

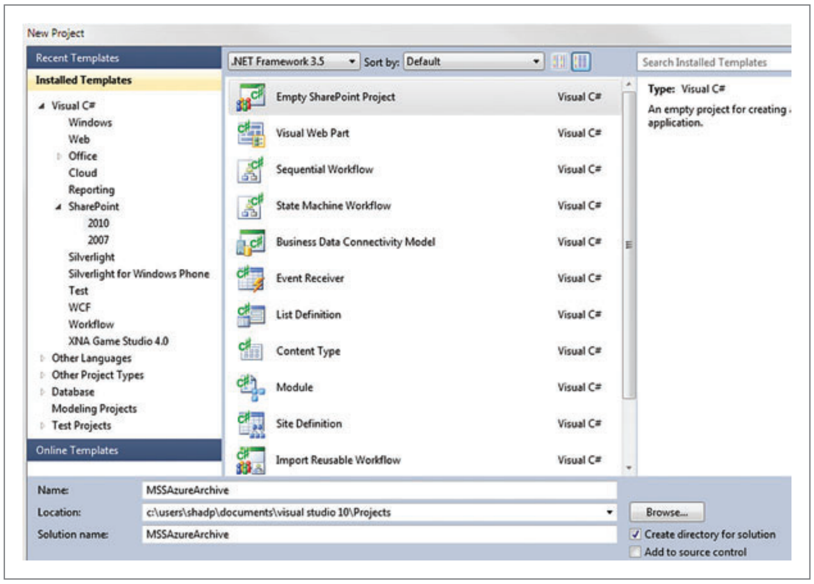
Add from existing site content types
Change new button order and default content type

Figure 2 **Our New "Link to a Document" Content Type**



```
<appSettings>
    <!--Windows Azure Settings-->
   <add key="AccountName" value="devstoreaccount1" />
   <add key="AccountSharedKey"

value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBekso
GMGw==" />
   <add key="DefaultContainerName" value="Documents" />
   <add key="BlobStorageEndpoint" value="http://127.0.0.1:10000/devstoreaccount1/" />
   <add key="QueueStorageEndpoint" value="http://127.0.0.1:10001/devstoreaccount1/" />
   <add key="TableStorageEndpoint" value="http://127.0.0.1:10002/devstoreaccount1/" />
  </appSettings>
```

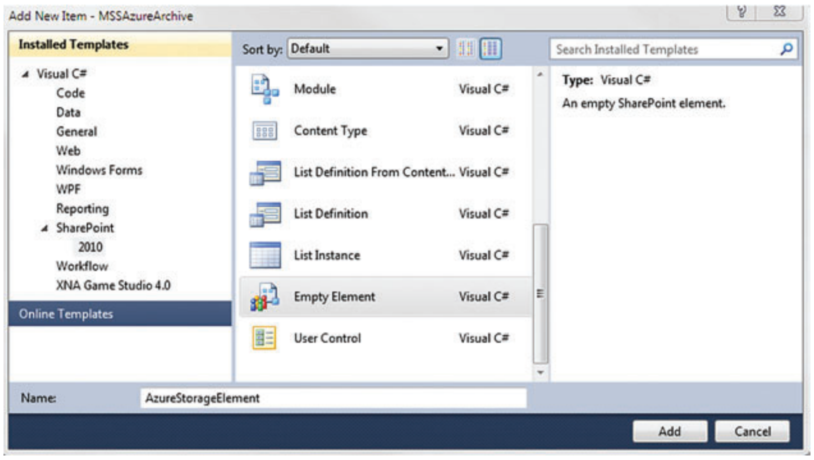
Figure 3 **Adding Keys in Web.config**



Type: Visual C#
An empty project for creating a application.

Name: MSSAzureArchive
Location: c:\users\shadp\documents\visual studio 10\Projects
Solution name: MSSAzureArchive
✓ Create directory for solution

Figure 4 **Project Selection in Visual Studio 2010**



Type: Visual C#
An empty SharePoint element.

Name: AzureStorageElement

Figure 5 **Adding an AzureStorageElement via Add New Item**

won't allow. References needed to be added to the project for Microsoft.Windows.Azure.StorageClient and System.Web. Next, we added an Empty Element item to the project using the Empty-Element template and named it AzureStorage-Element. We added a <CustomAction/> element in order to add a new action item to the context menu for the document library items (see **Figure 5**).

A new Feature called Feature1 was automatically added to the project, which we renamed to MSSAzureArchive. We replaced the contents of the Elements.xml file for the AzureStorageElement that was added with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <CustomAction
    Id="UserInterfaceCustomActions.ECBItemToolbar"
    RegistrationType="List"
    RegistrationId="101"
    Location="EditControlBlock"
    Sequence="106"
    Title="Azure Storage">
    <UrlAction Url="~sitecollection/
      _layouts/MSSAzureArchive/
      AzureStorage.aspx?ItemUrl={ItemUrl}" />
  </CustomAction>
</Elements>
```

For the uninitiated SharePoint developer, **Figure 6** shows a quick description of some of the <Custom-Action/> properties (more information about the <CustomAction/> element and its properties can be found at msdn.microsoft.com/library/ms460194).

Note the Url property of the UrlAction element; this is the navigation action that happens in order to handle the command to archive the document. Based on this configuration, SharePoint knows where to put the feature in the UI and also what to do when someone clicks on it. SharePoint will navigate to a page we create that will handle the archiving of the selected document. This page will allow the user to select a target storage container or create a new storage container for the item, so we needed to add an Application Page item to the project. Once again using the SharePoint 2010 templates, we chose the Application Page template and named it AzureStorage.aspx (see **Figure 7**).

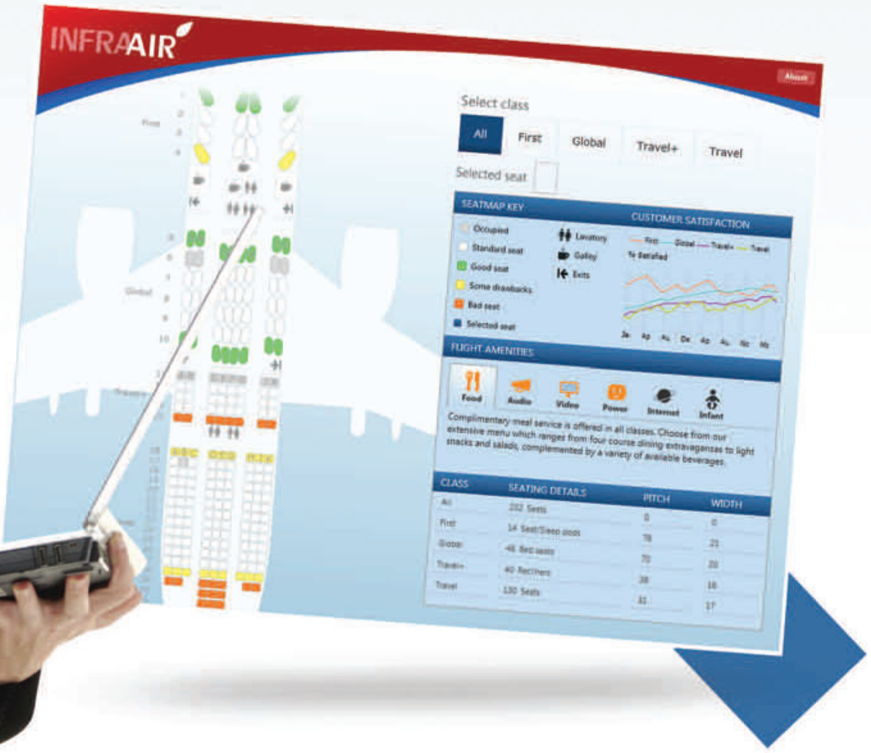
Because this sample wasn't meant to impress anyone with a genius UI design, we added only minimal controls needed to get the job done. In the <asp:Content> element of the page markup, we added the code shown in **Figure 8**.

Next, we edited the code-behind and wired the UI elements up to some code to talk to Windows Azure Storage, and rendered the information. We initialized the cloud storage client within the load event for the page and grabbed the available containers using the previous web.config settings (see **Figure 9**).

Because the focus here is on the archive functionality, we concentrated on that. The other code is

DESIGN

Design Applications That Help Run the Business



Complimentary meal service is offered in all classes. Choose from our extensive menu which ranges from four course dining extravaganzas to light snacks and salads, complemented by a variety of available beverages.

CLASS	SEATING DETAILS	PITCH	WIDTH
All	232 Seats	0	0
First	14 Seat/Sleep pods	78	21
Global	48 Bed seats	70	20
Travel+	40 Recliners	38	18
Travel	130 Seats	31	17

Our xamMap™ control in Silverlight and WPF lets you map out any geospatial data like this airplane seating app to manage your business. Come to infragistics.com to try it today!

NetAdvantage® ULTIMATE

for ASP.NET, Windows Forms, WPF, Silverlight, WPF Data Visualization, Silverlight Data Visualization

Infragistics®

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91 80 4151 8042
@infragistics

Figure 6 **Properties of the <CustomAction/> Element**

Property	Function
Id	Unique identifier.
Location	Specifies where in the SharePoint UI the element should appear. In this case, the item menu (EditControlBlock) is the desired location versus, for example, the ribbon.
Sequence	Specifies the ordering priority for the actions.

available via download (code.msdn.microsoft.com/mag201012Cloudy). We added a click handler for the archiveFile button and wired the Archive_Click function to it. Based on the UrlAction element, we can retrieve the path to the item. In the click function, the item is fetched from SharePoint using the object model, checked to see if it has already been archived and—if not—uploaded to the selected container (see **Figure 10**).

After the item is uploaded to storage, a new archive item of the type "Link to a Document" is created in place of the original document and the original document is deleted. If this were a publishing case instead of archival, the original item likely wouldn't be deleted, but rather just marked as published with a link to the published version. The original item is used to get the target document library and the path of the original document. The new item is marked as archived by adding the IsArchived property and assigning the value "true." First we did some work to get some of the values we needed, and then we created the new item and assigned the values to it, as shown here:

```
SPDocumentLibrary docLib =
  fileToArchive.DocumentLibrary;

Hashtable docProperties = new Hashtable();
docProperties["IsArchived"] = true;
string docLibRelPath =
  docLib.RootFolder.ServerRelativeUrl;
string docLibPath = string.Empty;
webSiteCollection = SPContext.Current.Site;
docLibPath =
  webSiteCollection.MakeFullUrl(docLibRelPath);

string azureURL = cloudBlob.Uri.ToString();
```

The function BuildLinkToItem creates an instance of the content type "Link to a Document" using the path to the item in Windows Azure Storage. This instance of the content type will be added to the library as the link to retrieve the item from Windows Azure Storage via the SharePoint UI, as shown here:

```
string azureStub = this.BuildLinkToItem(azureURL).ToString();

  SPFile newFile = webSite.Files.Add(documentPath,
    UTF8Encoding.UTF8.GetBytes(azureStub), docProperties, true);

  SPListItem item = newFile.Item;
  item["Content Type"] = "Link to a Document";
  SPFieldUrlValue itemUrl = new SPFieldUrlValue();
  itemUrl.Description = fileToArchive.Name;
  itemUrl.Url = azureURL;
  item["URL"] = itemUrl;
  item["IsArchived"] = true;
  item.Update();
  fileToArchive.Delete();
```

With the code completed to save the document, move it and replace it with a link to Windows Azure Storage, it was time to focus on the build and deployment of the solution. We double-clicked on the Package.package file to bring up the package designer and subsequently selected the Advanced tab at the bottom of the screen. This is where we added the package assemblies that we needed in order to include the Microsoft.WindowsAzure.StorageClient.dll. Keeping it simple for this sample, we set the Deployment Target to the GlobalAssemblyCache. We ensured that Development Storage was running by navigating to the Server Explorer, clicking on the Windows Azure Storage node, then clicking on the "(Development)" node.

Throwing caution to the wind, we pressed F5 to build, deploy, attach to a process and initiate a browser session to start debugging

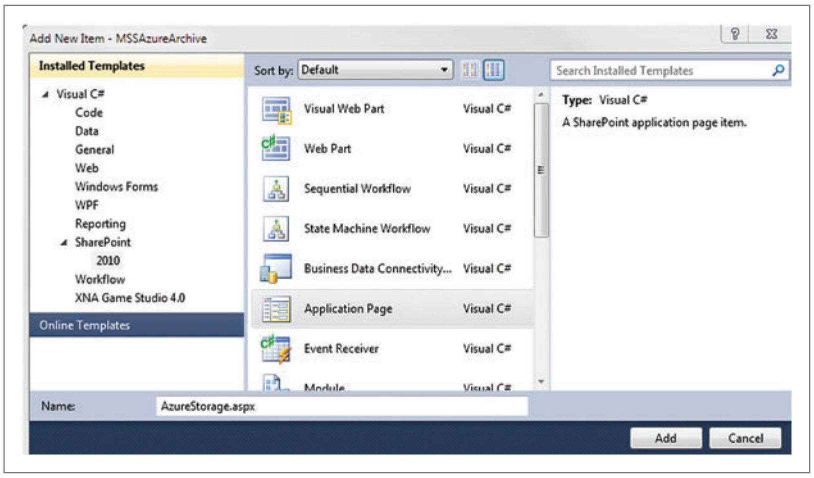


Figure 7 **Adding a New Page in SharePoint 2010**

our feature. We navigated back to the Shared Documents library mentioned earlier and opened the drop-down menu attached to the document we previously loaded. In the drop-down, we selected our new element, Azure Storage, which took us to the custom application page to select the destination container (see **Figure 11**).

Once on the page, we could have created a new container, but instead we used the documents container that we created and clicked the Archive File button to execute the code from earlier (see **Figure 12**).

Figure 8 **Adding the Minimum-Needed Controls**

```
 Document to Archive:
<asp:Label ID="fileName" runat="server" ></asp:Label>   <br/>
Choose Azure Container:
<asp:DropDownList ID="azureContainers" runat="server"
  Visible="true"></asp:DropDownList>
<asp:TextBox id="newContainerName" runat="server" Visible="false"></asp:TextBox>
<asp:Button ID="saveContainer" runat="server" Text="Save Container"
  OnClick="SaveContainer_Click" Visible="false"></asp:Button>
<br />
<asp:Button ID="createContainer" runat="server" Text="Create New Container"
  OnClick="CreateContainer_Click" />
<br/>
<asp:Button ID="archiveFile" runat="server" Text="Archive File"
  OnClick="Archive_Click" />
<br/>
<asp:Label ID="errMessage" runat="server" Text=""></asp:Label>
```

Powerful Tools for Developers

v4.5!

High-Performance PDF Printer Driver

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

v4.5!

PDF Editor for .NET, now Webform Enabled

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

New!

PDF Integration into Silverlight Applications

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

New Touchscreen Tablet for Mobile Development!

The DevTouch Pro is a new color touchscreen tablet designed to provide mobile application developers with a customizable development, testing and deployment platform.

Microsoft Windows CE   android   Linux

- Fully open customizable tablet
- Develop with .NET, Java or C++
- Unrestricted development and flexible quantities
- Fully supported in North America

WiFi™ Bluetooth® MADE IN CANADA · FABRIQUÉ AU CANADA

Learn more at www.devtouchpro.com

Windows 7   CERTIFIED FOR Windows Server 2008

More Development Tools Available at:

www.amyuni.com

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

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

AMYUNI Technologies

Figure 9 **Initializing the Cloud Storage Client**

```
protected void Page_Load(object sender, EventArgs e)
{

  this.InitializeCloudStorage();
  if (!IsPostBack)
  {
    this.GetContainers();
  }
}

private void GetContainers()
{
  IEnumerable<CloudBlobContainer> blobContainers =
    cloudBlobClient.ListContainers();

  this.azureContainers.DataSource = blobContainers;
  azureContainers.DataTextField = "Name";
  this.azureContainers.DataBind();
  if (azureContainers.Items.Count < 1)
  {
    ListItem defaultContainer = new ListItem(defaultContainerName);
    defaultContainer.Selected = true;
    azureContainers.Items.Add(defaultContainer);
  }
}
```

Figure 10 **Code for the Click Function to Fetch the Item from SharePoint**

```
protected void Archive_Click(object o, EventArgs e)
{
  try
  {
    webSite = SPContext.Current.Web;
    filePath = webSite.Url.ToString() +
    Request.QueryString["ItemUrl"].ToString();
    fileToArchive = webSite.GetFile(filePath);
    string sArchived = fileToArchive.Item["IsArchived"].ToString();

    bool isArchived = Convert.ToBoolean(sArchived);

    if (isArchived)
    {
      errMessage.Text = "This document has already been archived!";
    }
    else
    {
      string newGuid = Guid.NewGuid().ToString();
      string uniqueBlobName = string.Format(newGuid + "_" +
        fileToArchive.Name.ToString());

      blobContainer = cloudBlobClient.GetContainerReference(
        this.azureContainers.SelectedValue);
      blobContainer.CreateIfNotExist();
      cloudBlob = blobContainer.GetBlockBlobReference(uniqueBlobName);
      cloudBlob.UploadByteArray(fileToArchive.OpenBinary());
```

With the file archived to Windows Azure Storage, we navigated back to the Shared Documents library. Instead of seeing the document, we saw a Link to a Document item that replaced the Services SOW.docx Word document (see **Figure 13**).

When we looked at the properties of the item, we saw the fields related to the content type, and in particular the URL to where the document now resides in Windows Azure Storage (see **Figure 14**).

We could open the document directly from Windows Azure Storage by clicking on Link to a Document. We could use the URL property to access it directly or via some other code or UI. For example, if we still wanted to index these items via the SharePoint Index Service, we could create a custom IFilter that would know how to process my Link to a Document Content Type to ensure the content would get indexed properly.

With the implementation of archiving content from a SharePoint Document Library to a Windows Azure Storage Container out of the way, it left us with only public or no access to the archived documents for unauthenticated requests.

Access Control When Publishing

As mentioned earlier, what lead me to talk with Shad about his archival piece was the use of Windows Azure Storage as a means to provide a public landing spot for content that had gone through review and approval. In the case that I was considering, I didn't have to include any access control because the documents were to be shared with everyone. However, it only took a few minutes for someone to ask the question: "What if we want to publish something and make it available only to certain people, for example, vendors, customers or employees?" Often a task like this is accomplished within companies by including those people as part of a corporate domain or having them somehow federated so they're identified via username and password challenge. This wasn't the case here, and the customer didn't really want to set up some application layer or front end to control access; developing the front end would reduce the value by increasing implementation costs.

Our solution concepts don't have to be in the cloud or on-premises exclusively. The two can be easily mixed.

One solution is to use a SharedAccessPolicy on the blobs. The container and blobs in the container would have their PublicAccess set to Off with a little bit of code you would likely write doing Windows Azure Storage development anyway. The following code sample shows how I can set the PublicAccess to Off, but allow for SharedAccess on the container should I generate a signature and hand it out:

```
BlobContainerPermissions permissions = new BlobContainerPermissions();
permissions.PublicAccess = BlobContainerPublicAccessType.Off;

SharedAccessPolicy accesspolicy = new SharedAccessPolicy();
accesspolicy.Permissions = SharedAccessPermissions.Read;
permissions.SharedAccessPolicies.Add("Read", accesspolicy);

BlobContainer.SetPermissions(permissions);
```

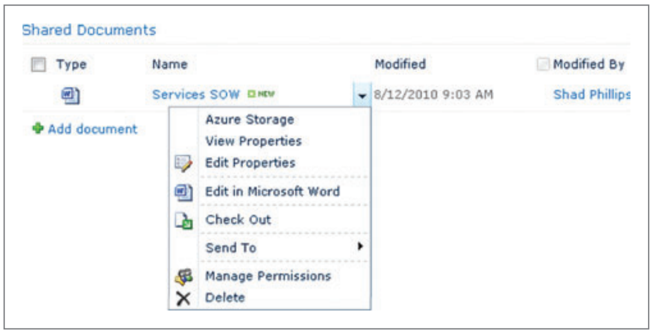


Figure 11 **Selecting the Windows Azure Storage Element**

WE ARE

REPORTING

The defacto standard reporting tool for Microsoft Visual Studio.NET

- Fast and Flexible reporting engine
- Flexible event-driven API to completely control the rendering of reports
- Wide range of Export and Preview formats including Windows Forms Viewer, Web Viewer, Adobe Flash and PDF
- XCopy deployment
- Royalty-Free Licensing for Web and Windows applications

ACTIVEREPORTS

ACTIVEREPORTS   SPREAD   DATA DYNAMICS REPORTS   ACTIVEANALYSIS

GrapeCity PowerTools
Report Analyze Excel

www.GCPowerTools.com

WE ARE
SPREADSHEETS

Award-winning Microsoft® Excel® compatible spreadsheet components for .NET and ASP.NET

- World's best-selling .NET spreadsheet technology
- Hundreds of Chart styles for data visualization
- Full featured Formula support, including most Excel functions
- Full support for native Microsoft Excel files and data import/export
- Spreadsheet Designers, Quick-start Wizard and Chart Wizards

SPREAD

ACTIVEREPORTS SPREAD DATA DYNAMICS REPORTS ACTIVEANALYSIS

GrapeCity PowerTools
Report Analyze Excel

www.GCPowerTools.com

If we ask for a resource directly in the storage container, we'll get a 404 Page Not Found. As we upload the blobs, we do a similar bit of work for the blob itself, but we create a SharedAccessPolicy that allows read, set an expiry time for it and ask for a Shared Access Signature back, like this:

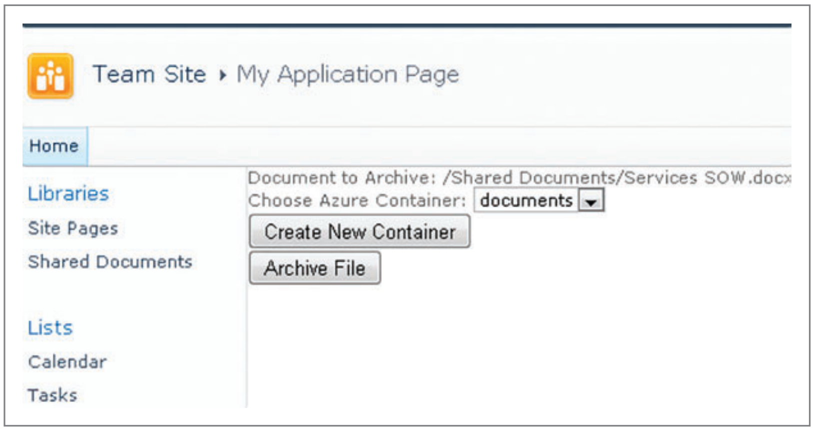
```
SharedAccessPolicy policy = new SharedAccessPolicy();
policy.Permissions = SharedAccessPermissions.Read;
policy.SharedAccessExpiryTime = DateTime.Now.AddDays(5);
string SharedAccessSignature = destBlob.GetSharedAccessSignature(policy));
```

The call to GetSharedAccessSignature returns a string like this:

```
?se=2010-08-26T18%3A22%3A07Z&sr=b&sp=r&sig=WdUHKvQYnbOcMwUdFavn4QSOlvhAn
qBAnVnC6xOzPj8%3D
```

If I concatenate that query string onto the end of the URI for the blob, I should receive it back, providing the expiry hasn't passed. More information about the signature and Shared Access Policies can be found at msdn.microsoft.com/library/ee395415.

To solve the problem, I'd generate signatures and provide signed URIs that had a long expiry, which makes it easy to create them at the time of upload and then store a list of links to the published documents. For something a little more secure and for which I want to provide access to a single user for a short period of time, I would need a piece of UI. That UI would allow a user to request access to one or more resources and get back signed URIs that would provide access for a short duration of time.

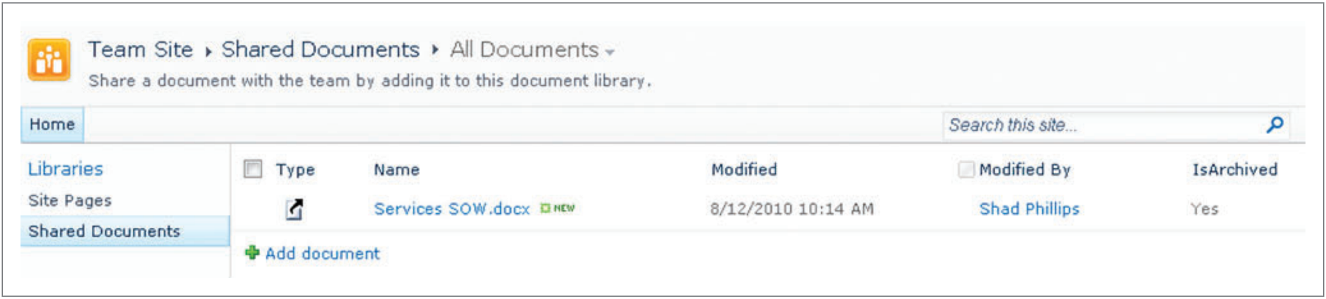


Document to Archive: /Shared Documents/Services SOW.docx
Choose Azure Container: documents

Figure 12 **Selecting the Container**

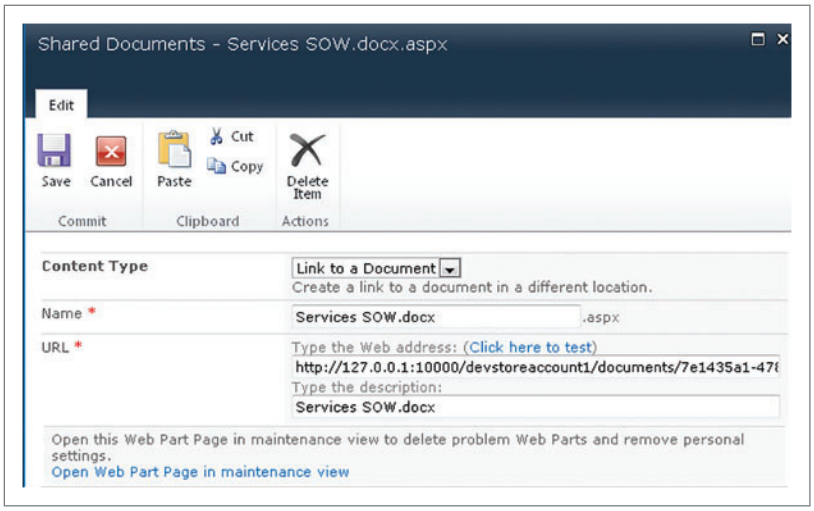
Mixing with the Cloud

Here Shad and I used one general implementation to address two different scenarios. This was particularly useful for both scenarios, as we needed a particular piece of functionality (scalable, reliable and expandable storage) that the cloud provided without us having to do much in the way of setup and costing only what we used. The main idea that we hope to convey is that, as professionals looking to create solutions for our customers (internal or external), our solution concepts don't have to be in the cloud or on-premises exclusively. The two can be easily mixed. As the Service Updates get applied over time, it will become easier and easier to blend the corporate network to the cloud network. I expect in the future that it will blend to the point of there not being much of a difference. So, as you're looking at solutions for your software or business systems, you might take a moment to pause and think, "Is there something in the cloud that will help me?" ■


Team Site ▸ Shared Documents ▸ All Documents
Share a document with the team by adding it to this document library.
Home
Search this site...
Libraries
Site Pages
Shared Documents
Add document

Type	Name	Modified	Modified By	IsArchived
	Services SOW.docx NEW	8/12/2010 10:14 AM	Shad Phillips	Yes

Figure 13 **Link to Document in the SharePoint Documents Library**



Content Type	Link to a Document Create a link to a document in a different location.
Name *	Services SOW.docx .aspx
URL *	Type the Web address: (Click here to test) http://127.0.0.1:10000/devstoreaccount1/documents/7e1435a1-47 Type the description: Services SOW.docx

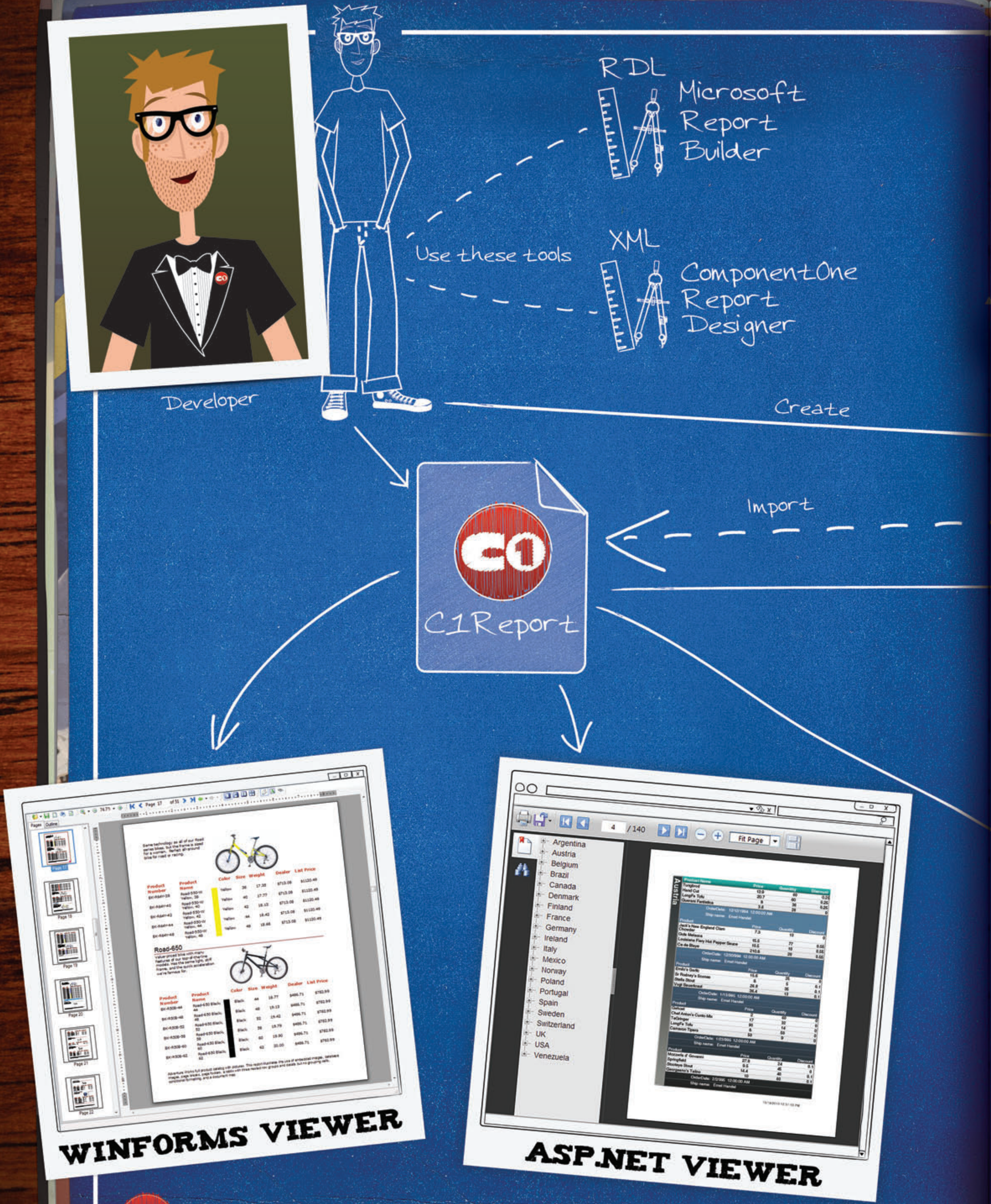
Open this Web Part Page in maintenance view to delete problem Web Parts and remove personal settings.
Open Web Part Page in maintenance view

Figure 14 **Link Properties**

JOSEPH FULTZ *is an architect at the Microsoft Technology Center in Dallas where he works with enterprise customers and ISVs designing and prototyping software solutions to meet business and market demands. He's spoken at events such as Tech·Ed and similar internal training events.*

SHAD PHILLIPS *is an architect at the Microsoft Technology Center in Dallas where he works with enterprise customers and partners designing and deploying enterprise content management solutions built on Microsoft SharePoint 2010.*

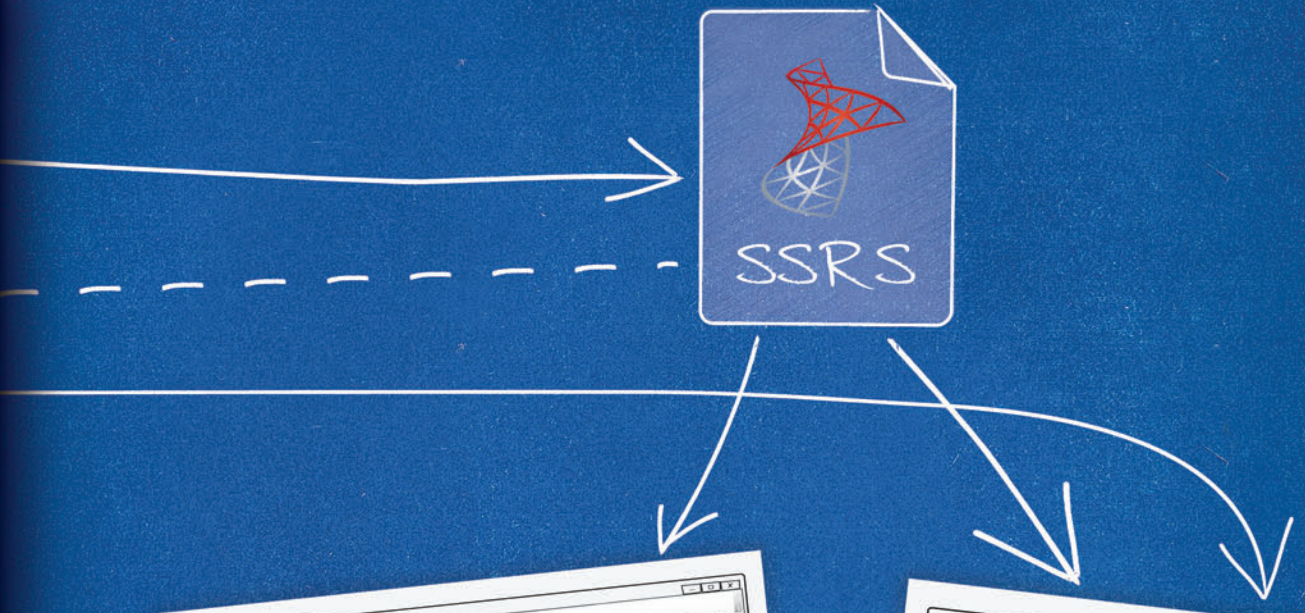
THANKS to the following technical expert for reviewing this article: *Jasen Tenney*

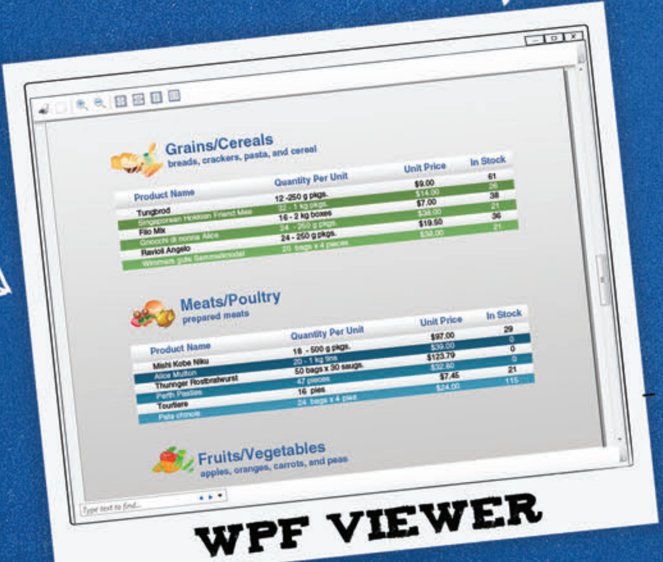


ComponentOne

YOUR GUIDE TO REPORTS

Developers, just like you, continue to turn to ComponentOne Reports™ for their enterprise solutions. Our flexible reports support Microsoft SQL Server Reporting Services (SSRS) with a fully customizable object model, report designers, schedulers, viewers for all .NET platforms, and more. Start your blueprint today; download ComponentOne Studio® Enterprise for high quality, professional apps.





Grains/Cereals
breads, crackers, pasta, and cereal

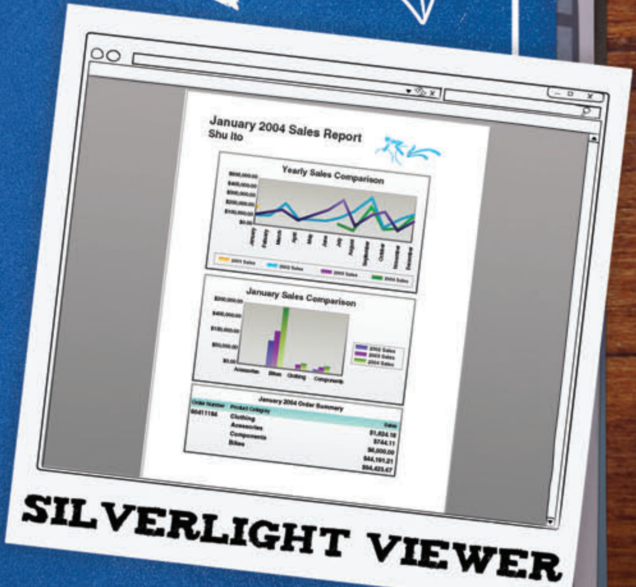
Product Name	Quantity Per Unit	Unit Price	In Stock
Tungbrod	12 -250 g pkgs.	$9.00	61
Singaporean Hokkien Friend Mee	32 - 1 kg pkgs.	$14.00	26
Filo Mix	16 - 2 kg boxes	$7.00	38
Gnocchi di nonna Alice	24 - 250 g pkgs.	$38.00	21
Ravioli Angelo	24 - 250 g pkgs.	$19.50	36
Wimmers gute Semmelknodel	20 bags x 4 pieces	$38.00	21

Meats/Poultry
prepared meats

Product Name	Quantity Per Unit	Unit Price	In Stock
Mishi Kobe Niku	18 - 500 g pkgs.	$97.00	29
Alice Mutton	20 - 1 kg tins	$39.00	0
Thuringer Rostbratwurst	50 bags x 30 saugs.	$123.79	0
Perth Pasties	47 pieces	$32.80	0
Tourtiere	16 pies	$7.45	21
Pate chinois	24 bags x 4 pies	$24.00	115

Fruits/Vegetables
apples, oranges, carrots, and peas

WPF VIEWER



January 2004 Sales Report
Shu Ito

January 2004 Order Summary

Order Number	Product Category	Sales
50411186	Clothing	$1,824.18
	Accessories	$744.11
	Components	$6,000.00
	Bikes	$44,191.21
		$94,423.47

SILVERLIGHT VIEWER

ComponentOne®
Studio® Enterprise 2010

Download your free trial @
www.componentone.com/blueprint

Sudoku for Windows Phone 7

Adam Miller

Sudoku has become a popular game over the last 10 years, finding a home in most newspapers right next to the crossword puzzle. Game shows based on Sudoku have even been created. If you're unfamiliar with it, Sudoku is a number placement game. The game board is a 9x9 grid and the goal is to place the numbers 1-9 in the grid such that each row, column and sub 3x3 grid contains each number once. The nature of the game lends itself nicely to play on a portable device, and Windows Phone 7 should be no exception. You can soon expect a handful of Sudoku applications in the marketplace after the recent release of Windows Phone 7, and you can even add your own to that list by following this article.

This article discusses:

- The Model-View-ViewModel design pattern
- Creating a Visual Studio application
- Implementing individual squares
- Implementing the game board
- Implementing the input grid
- Bringing the application together in MainPage.xaml

Technologies discussed:

Windows Phone 7

Code download available at:

code.msdn.microsoft.com/mag201012Sudoku

MVVM Introduction

My application will roughly follow the Model-View-ViewModel (MVVM) design pattern. Although there won't be any actual Models (because this app doesn't require database storage), it will still be a good learning tool because the ViewModel is really the core of the pattern.

There can be a bit of a learning curve to understand the MVVM pattern, but once it clicks, you'll find you can get a really nice separation between the UI and business logic. Furthermore, it reveals the power of data binding in Silverlight while freeing you from the majority of the tedious code of updating a UI (FirstNameTextBox.Text = MyPerson.FirstName will be a thing of the past!). For more information about data binding in Silverlight, take a look at the MSDN Library article "Data Binding" at tinyurl.com/SLdatabind.

Because of the size and simplicity of this app, and the focus of this article, a third-party MVVM framework won't be used. However, it's likely that your application will grow to be more complex than this one, and you'd be wise to start with a third-party framework such as the MVVM Light Toolkit (mwmlight.codeplex.com). It will provide you with free, tested code that you'll end up writing anyway (noted from experience).

Creating the Application

After you've installed the developer tools from create.msdn.com, start by creating your new Windows Phone 7 project by opening Visual Studio and selecting File | New | Project, then in the new project dialog, Visual C# | Silverlight for Windows Phone | Windows

Phone Application. Start by creating two new folders, Views and ViewModels, following a common MVVM pattern. At this point, you can also start debugging if you want to take a peek at the emulator provided as part of the SDK.

The Sudoku game can be broken down into three conceptual types: each of the individual squares (81 total in the typical 9x9 board); the overall game board that houses the squares; and a grid for the numbers 1-9 for input. To create the views for these items, right-click on the Views folder and select Add | New Item. Select Windows Phone User Control from the dialog and name the first file GameBoardView.xaml. Repeat for SquareView.xaml and InputView.xaml. Now, in the ViewModel folder, add the following classes: GameBoardViewModel and SquareViewModel. The Input View won't require a ViewModel. You'll also want to create a base class for your ViewModels to avoid code duplication. Add a ViewModelBase class to your ViewModels folder. At this point, your solution should look similar to **Figure 1**.

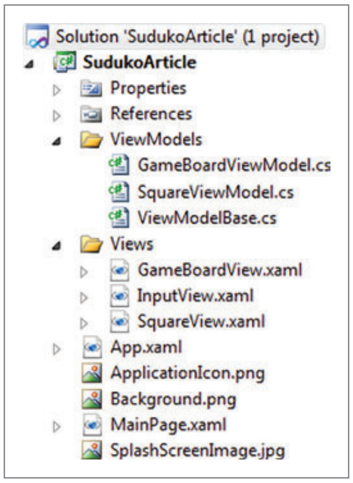


Figure 1 **Sudoku Windows Phone 7 Solution with Views and ViewModels**

There can be a bit of a learning curve to understand the MVVM pattern.

ViewModel Base Class

The ViewModelBase class will need to implement the INotifyPropertyChanged interface found in System.ComponentModel. This interface is what allows the public properties in ViewModels to bind to controls in the views. The implementation of the INotifyPropertyChanged interface is pretty simple—only the PropertyChanged event must be implemented. Your ViewModelBase.cs class should look similar to the following (don't forget the using statement for System.ComponentModel):

```
public class ViewModelBase : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler
    PropertyChanged;
  private void NotifyPropertyChanged(String info)
  {
    if (PropertyChanged != null)
    {
      PropertyChanged(this,
        new PropertyChangedEventArgs(info));
    }
  }
}
```

Most of the third-party MVVM frameworks will include a ViewModel base class that contains this boilerplate code. All of your ViewModels will inherit from ViewModelBase. The properties in a ViewModel that the UI will bind to need to call NotifyPropertyChanged in the setter. This is what allows the UI to automatically update when the value of a property changes. It does get a bit tedious implementing all of your properties in this way, so it's a bit of a tradeoff for the code that you don't have to write to update the UI.

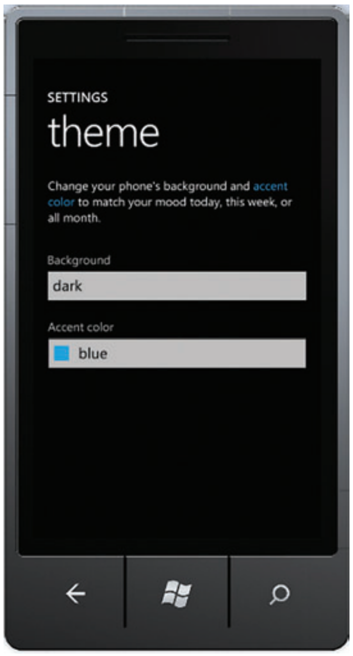

SETTINGS
theme
Change your phone's background and accent color to match your mood today, this week, or all month.
Background
dark
Accent color
blue

Figure 2 **Windows Phone 7 Theme Settings Screen**

Implementing the Individual Squares

Start by implementing the SquareViewModel class. Add public properties for Value, Row and Column as integers; and IsSelected, IsValid and IsEditable as Booleans. Although the UI can bind to the Value property directly, this will cause issues because "0" will be displayed for unassigned squares. To resolve this, you can either implement a binding converter or create a read-only "StringValue" property that will return an empty string when the Value property is zero.

The SquareViewModel will also be responsible for notifying the UI of its current state. The four states for an individual square in this app are Default, Invalid, Selected and UnEditable. Normally this would be implemented as an enum; however, enums in the Silverlight framework don't have a couple of the methods that enums in the full Microsoft .NET Framework have. This causes an exception to be thrown during serialization, so the states have been implemented as constants:

```
public class BoxStates
{
  public const int Default = 1;
  public const int Invalid = 2;
  public const int Selected = 3;
  public const int UnEditable = 4;
}
```

Now, open SquareView.xaml. You'll notice some styles have been applied at the control level for the font size and color. The preset style resources are usually found in a separate resources file, but in this case, Windows Phone 7 provides them to your application by default. The resources are described on the MSDN Library page, "Theme Resources for Windows Phone," at tinyurl.com/WP7Resources. Some of these styles will be used in this application so the application colors will match the user-selected theme. The theme can be selected in the emulator by going to the home screen and clicking the more arrow | Settings | theme. From here you can change the background and accent colors (**Figure 2**).

Inside the grid in SquareView.xaml, place a Border and a TextBlock:

```
<Grid x:Name="LayoutRoot" MouseLeftButtonDown=
  "LayoutRoot_MouseLeftButtonDown">
  <Border x:Name="BoxGridBorder"
    BorderBrush="{StaticResource PhoneForegroundBrush}"
    BorderThickness="{Binding Path=BorderThickness}">
    <TextBlock x:Name="MainText"
      VerticalAlignment="Center" Margin="0" Padding="0"
      TextAlignment="Center" Text=
      "{Binding Path=StringValue}">
    </TextBlock>
  </Border>
</Grid>
```

The code-behind for SquareView.xaml.cs can be seen in the accompanying code download.

Figure 3 **The Board Storage Code**

```
public void SaveToDisk()
{
  using (IsolatedStorageFile store = IsolatedStorageFile.GetUserStoreForApplication())
  {
    if (store.FileExists(FileName))
    {
      store.DeleteFile(FileName);
    }

    using (IsolatedStorageFileStream stream = store.CreateFile(FileName))
    {
      using (StreamWriter writer = new StreamWriter(stream))
      {
        List<SquareViewModel> s = new List<SquareViewModel>();
        foreach (SquareViewModel item in GameArray)
          s.Add(item);

        XmlSerializer serializer = new XmlSerializer(s.GetType());
        serializer.Serialize(writer, s);
      }
    }
  }
}

public static GameBoardViewModel LoadFromDisk()
{
  GameBoardViewModel result = null;

  using (IsolatedStorageFile store = IsolatedStorageFile.
    GetUserStoreForApplication())
  {
    if (store.FileExists(FileName))
    {
      using (IsolatedStorageFileStream stream =
        store.OpenFile(FileName, FileMode.Open))
      {
        using (StreamReader reader = new StreamReader(stream))
        {
          List<SquareViewModel> s = new List<SquareViewModel>();
          XmlSerializer serializer = new XmlSerializer(s.GetType());
          s = (List<SquareViewModel>)serializer.Deserialize(
            new StringReader(reader.ReadToEnd()));

          result = new GameBoardViewModel();
          result.GameArray = LoadFromSquareList(s);
        }
      }
    }
  }

  return result;
}
```

Figure 4 **The Code-Behind for the Input View**

```
public event EventHandler SendInput;

private void UserInput_Click(object sender, RoutedEventArgs e)
{
  int inputValue = int.Parse(((Button)sender).Tag.ToString());
  if (SendInput != null)
      SendInput(inputValue, null);
}

public void RotateVertical()
{
  TopRow.Orientation = Orientation.Vertical;
  BottomRow.Orientation = Orientation.Vertical;
  OuterPanel.Orientation = Orientation.Horizontal;
}

public void RotateHorizontal()
{
  TopRow.Orientation = Orientation.Horizontal;
  BottomRow.Orientation = Orientation.Horizontal;
  OuterPanel.Orientation = Orientation.Vertical;
}
```

Figure 5 **Code to Handle Phone Rotation**

```
protected override void OnOrientationChanged(OrientationChangedEventArgs e)
{
  switch (e.Orientation)
  {
    case PageOrientation.Landscape:
    case PageOrientation.LandscapeLeft:
    case PageOrientation.LandscapeRight:
      TitlePanel.Visibility = Visibility.Collapsed;
      Grid.SetColumn(InputControl, 1);
      Grid.SetRow(InputControl, 0);
      InputControl.RotateVertical();
      break;
    case PageOrientation.Portrait:
    case PageOrientation.PortraitUp:
    case PageOrientation.PortraitDown:
      TitlePanel.Visibility = Visibility.Visible;
      Grid.SetColumn(InputControl, 0);
      Grid.SetRow(InputControl, 1);
      InputControl.RotateHorizontal();
      break;
    default:
      break;
  }
  base.OnOrientationChanged(e);
}
```

The constructor requires an instance of the SquareViewModel. This will be provided when the game board is bound. Also, there's a custom event raised when the user clicks inside the grid. Using custom events is one way to allow ViewModels to communicate with each other; however, for larger applications, this approach can get messy. Another option is to implement a Messenger class that will facilitate the communication. Most MVVM frameworks provide a Messenger (sometimes referred to as Mediator) class.

It may seem messy from an MVVM purist's standpoint to update the UI using the code-behind, but these items don't lend themselves nicely to a BindingConverter. The BoxGridBorder's BorderThickness is based on two properties, and the Foreground and Background brushes come from the application resources, which aren't readily accessible in a BindingConverter.

Implementing the Game Board

The GameBoard view and ViewModel can now be implemented. The view is simple, just a 9x9 grid. The code-behind, available in the code download, is almost as simple—just a public property to expose the ViewModel and a couple private methods to handle the child box click and binding the game array.

Using the Application Bar will make the application feel more integrated with the phone.

The ViewModel contains the bulk of the code. It contains methods to validate the board after user input, to solve the puzzle and to save and load the board from storage. The board is serialized to XML when saving, and IsolatedStorage is used to save the file. For full implementation, please see the source code download; the storage code is of most interest and is shown in **Figure 3** (note that you'll need a reference to System.Xml.Serialization).

ADVERTISEMENT

VSP²

VISUAL STUDIO PARTNER PROFILE

Just in Time for the Holidays: Solutions to Speed Up Your .NET Development Projects

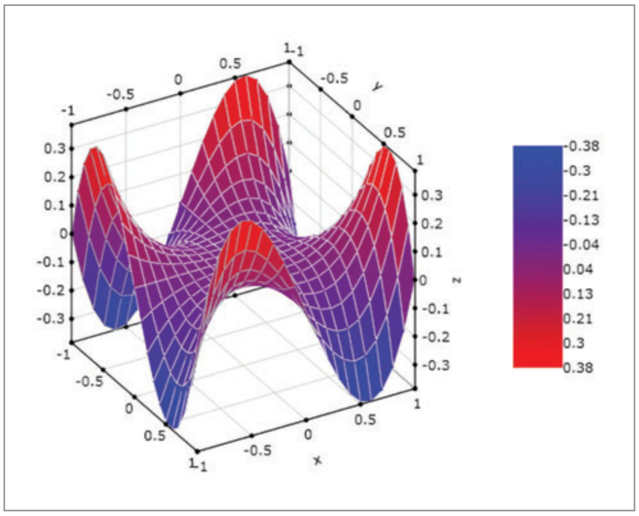
Rich Dudley
Technical Evangelist
ComponentOne

This time of year is always busy for developers. If we're not at an event like PDC, TechEd, or DevConnections, we're trying to figure out how to get started with new projects while finishing up existing projects. This is where third-party control vendors, like ComponentOne, help to jumpstart projects with unique controls, enhanced performance, and easy setup.

In their latest developer tools release, Studio Enterprise 2010 v3, ComponentOne continues to expand .NET capabilities in reporting, charting, and grids. This partner solution summarizes some of the new and updated controls found in this release.

Unified Reporting on Multiple Technologies

In many enterprises, the Business Intelligence (BI) infrastructure includes a number of reporting technologies, such as Microsoft Access, Crystal Reports, and SSRS. This variety hampers the delivery of information, as well as the ongoing development and maintenance of BI systems. ComponentOne's reporting controls allow enterprises to reap the benefits of a common BI infrastructure in less time than a total revamp of BI resources. For instance, they can import and display Access, Crystal, and SSRS reports in a single application. This application can be built in ASP.NET, Silverlight, WPF or WinForms. Also included in each studio are the C1ReportDesigner application and C1ReportScheduler control.



New Chart3D Control for Silverlight and WPF

Studios for Silverlight and WPF

In addition to the new reporting components, Studio Enterprise has expanded to include a PdfViewer, ScrollViewer, and Chart3D control. In the nearby screenshot, you can see the versatility of the 3D Chart control. Updated popular controls include the Scheduler, which now supports resource grouping; the famous FlexGrid, which supports native Silverlight 4 printing and additional group total options; and the Toolbar, which now has tabs to organize controls in a logical manner. For ease of design, ComponentOne ClearStyle™ technology is now supported in all controls, allowing designers or developers to achieve a uniform look with little effort.

Studio for WinForms

The popular gauges and charts controls have been updated too. The gauges now ship with over 20 templates to speed the development of BI dashboards. The C1Chart can now expose the VisualEffectsEditor, allowing end-users to design their own charts at run time. Since applications are typically more than a single screen, clean navigation is essential. The navigation controls, C1Ribbon and C1NavBar, have been updated to support the sleek Microsoft Office 2010 styles, and the C1DockingTab now supports two docking styles inspired by Microsoft Visual Studio 2008 and 2010.

Studio for ASP.NET AJAX

This studio includes the reporting improvements as well as more templates for gauges, additional groupings and subtotals for rows and columns in Excel for .NET, and an enhanced framework with jQuery.

Studio for iPhone

ComponentOne Studio for iPhone, a unique suite of ASP.NET controls, renders pages sized and styled for mobile browsers, including those found on iOS, Android, and WebOS. The controls employ the conventions of popular smartphone UI elements and come equipped with attractive built-in styles, meaning that developers only need to add the business aspects to build professional-looking, user-friendly mobile sites. Support for iPhone 4, iPad, and more Android phones have recently been added to this suite of controls.

While I understand this is a lot of functionality for you to take in a single article, I was hoping you'd find a control or feature you've been looking to add to your enterprise solution. You may also like to know that it's easy to get started. All the ComponentOne Studios listed above are available individually or as part of the Studio Enterprise. Additionally, the OLAP and LiveLinq components (available separately) can be combined with these controls to provide a highly performant and well-rounded BI implementation.

To access your 30-day, risk-free trial visit:
www.componentone.com/vsp2

C1 ComponentOne®

Implementing the Input Grid

The input view is simple as well, just some buttons nested in stackpanels.

The code-behind, shown in **Figure 4**, exposes a custom event to send the clicked button's value to the application, as well as two methods that will assist in making this game playable in portrait or landscape mode.

Bringing Views Together on MainPage.xaml

Finally, the application is brought together with the implementation of MainPage.xaml. The Input and GameBoard views are placed in a grid. This application will require all of the screen real estate available, so the PageTitle TextBlock that was automatically inserted when the project was created has to be removed. The ApplicationTitle TextBlock will only be visible in portrait mode. The Windows Phone 7 Application Bar will also be taken advantage of. Using the Application Bar will make the application feel more integrated with the phone and will provide the Sudoku application a nice interface to allow the user to solve, reset and start a new puzzle:

```
    <phone:PhoneApplicationPage.ApplicationBar>
      <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
        <shell:ApplicationBarIconButton x:Name="NewGame"
         IconUri="/Images/appbar.favs.rest.png" Text="New Game"
         Click="NewGame_Click"></shell:ApplicationBarIconButton>
        <shell:ApplicationBarIconButton x:Name="Solve"
         IconUri="/Images/appbar.share.rest.png" Text="Solve"
         Click="Solve_Click"></shell:ApplicationBarIconButton>
        <shell:ApplicationBarIconButton x:Name="Clear"
         IconUri="/Images/appbar.refresh.rest.png" Text="Clear"
         Click="Clear_Click"></shell:ApplicationBarIconButton>
     </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
```

The images are taken from a set of icons provided by Microsoft specifically for Windows Phone 7 that are installed with the tools at C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.0\Icons. After images are imported into the project, select the image properties and change Build Action from "Resource" to "Content" and Copy to Output Directory from "Do Not Copy" to "Copy If Newer."

The final piece of this application puzzle is to implement the MainPage code-behind. In the constructor, the SupportedOrientations property is set to allow the application to rotate when the user rotates the phone. Also, the InputView's SendInput event is handled and the input value is forwarded to the GameBoard:

```
public MainPage()
{
  InitializeComponent();
  SupportedOrientations = SupportedPageOrientation.Portrait |
    SupportedPageOrientation.Landscape;
  InputControl.SendInput += new
    EventHandler(InputControl_SendInput);
}

void InputControl_SendInput(object sender, EventArgs e)
{
  MainBoard.GameBoard.SendInput((int)sender);
}
```

The Navigation methods also need to be implemented to handle loading and saving the game board:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  GameBoardViewModel board =
    GameBoardViewModel.LoadFromDisk();
  if (board == null)
    board = GameBoardViewModel.LoadNewPuzzle();

  MainBoard.GameBoard = board;
  base.OnNavigatedTo(e);
}

protected override void OnNavigatedFrom(NavigationEventArgs e)
{
  MainBoard.GameBoard.SaveToDisk();
  base.OnNavigatedFrom(e);
}
```

When the phone is rotated, the application will receive a notification. This is where the InputView moves from below the game board to the right of it and is rotated (see **Figure 5**).

This is also where the menu item clicks are handled:

```
private void NewGame_Click(object sender, EventArgs e)
{
  MainBoard.GameBoard = GameBoardViewModel.LoadNewPuzzle();
}

private void Solve_Click(object sender, EventArgs e)
{
  MainBoard.GameBoard.Solve();
}

private void Clear_Click(object sender, EventArgs e)
{
  MainBoard.GameBoard.Clear();
}
```

At this point, the game is complete and playable (see **Figures 6** and **7**).

So there you have it, a nice game waiting for you the next time you're waiting in line. This article has demonstrated how to get started creating Silverlight-based Windows Phone 7 applications. Also, it has shown how to use serialization and user storage to persist an application and how to allow your application to support multiple orientations. In addition, you should now be familiar with the MVVM pattern and how to use data binding with it. ■

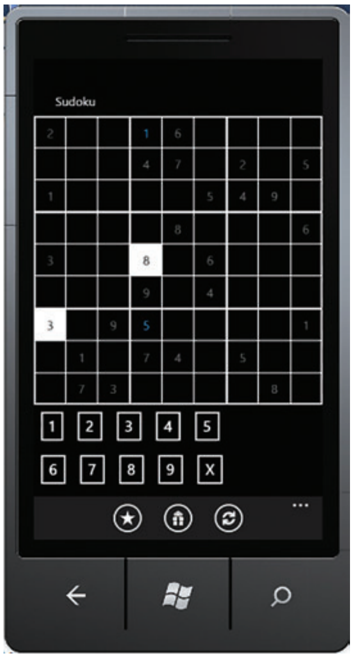


Figure 6 **Sudoku Game in Portrait Mode**

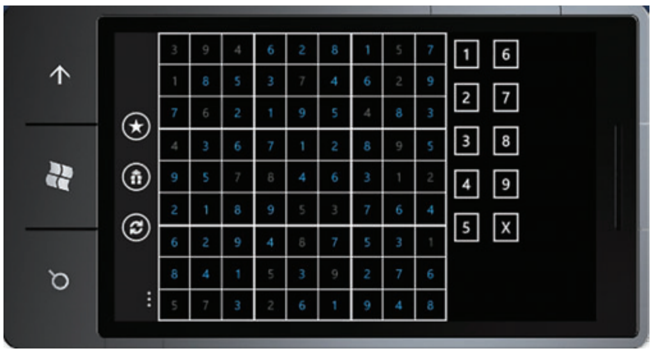


Figure 7 **Solved Game in Landscape Mode**

ADAM MILLER *is a software engineer for Nebraska Global in Lincoln, Neb. You can follow him at blog.milrr.com.*

THANKS to the following technical experts for reviewing this article: *Larry Lieberman and Nick Sherrill*

Try our Jump Start Package Today
(FIRST 10 USERS FREE)
Call 1.800.439.7782 or visit www.techexcel.com

TechExcel

Knowledge-Centric ALM for Today's Enterprises

DevSuite

Agile Studio — DevTest Studio

DevSpec	DevPlan	DevTrack	DevTest
Requirements Management	Development Project Planning	Implementation and Defect Tracking	QA Test Management

With DevSuite, you can deploy one module, a multi-module solution or the complete fully integrated ALM solution

- DevSpec – Requirements management
- DevPlan – Project planning and tracking
- DevTrack – Defect tracking and task management
- DevTest – Test management
- Agile Studio – Agile planning and implementation
- DevTest Studio – Complete quality management
- DevSuite – Complete Application Lifecycle Management

1.800.439.7782

www.techexcel.com

Build Data-Driven Apps with Windows Azure and Windows Phone 7

Danilo Diaz and Max Zilberman

In the last 30 years, we've seen an explosion in the computer hardware industry. From mainframes to desktop computers to handheld devices, the hardware keeps getting more powerful even as it shrinks. Developers have, to some extent, become a bit spoiled by this constant increase in computing power and now expect limitless computer resources on every device for which they write applications. Many younger developers have no memory of a time when the size and efficiency of your code were important factors.

The latest trend in development is in embracing the rise in popularity of smartphones. When coding for smartphone devices, many developers have to adjust to the fact that, although today's phones are extremely powerful compared to devices of just a few years ago, they do face limitations. These limitations are related to size, processor power, memory and connectivity. You need to understand how to work around these limitations when creating mobile applications to ensure good performance and the best user experience.

This article discusses:

- UI performance
- Storing and returning data
- Dealing with network failures
- Caching local data

Technologies discussed:

Windows Phone 7, Windows Azure

Some of the reasons for less-than-optimal app performance can be attributed directly to poor design decisions by the developer. However, in other cases, some of these factors are not directly in the control of the developer. A poorly performing application could be a symptom of a slow or offline third-party service, dropped mobile broadband connections or the nature of the data you're working with (such as streaming media files or large sets of data).

Whatever the cause might be, the performance perceived by the end user of your application must be one of the top concerns of any software developer. In this article, we'll cover some high-level considerations for designing robust, data-driven Windows Phone 7 applications in a way that can provide a great user experience and scale gracefully.

Let's first take a moment and set up a scenario within which we can examine some design and coding choices. As an example, we're going to work with a fictitious travel information application that provides information about user-selected airline flights. As shown in **Figure 1**, the main screen of the application shows a number of data elements including current weather and flight status. You can see that, as applications become more expressive and data-centric, developing them becomes a bit more challenging. There are simply more areas where your code can fall short.

UI Thread Blocking

Let's start by looking at the UI. It's easy to get the pattern wrong by designing the app as if you're coding for the desktop, so let's take a look at some phone-specific UI issues.

When an application doesn't respond as expected to user commands, the effect on the overall user experience can be dramatic. Slow response to swipe, tap or pinch can be detrimental to the overall appeal of the app. However, these are pretty simple issues to anticipate and address, as you'll see.

Consider a ListBox. When an ItemTemplate contains images or is loading data from a feed, there's a very good chance the UI thread will be blocked and the UI will pause until the requests or calculations complete. Therefore, as you develop the UI, one approach is to perform long calculations—including WebRequests—off of the UI thread. In fact, this is a good approach for any app, mobile or not.

Another issue that may create performance problems is when you're binding lots of items to the ItemSource without throttling injection into the ListBox control. A better approach would be to bind an ObservableCollection and populate a few items into the collection every 20-30 ms. This unlocks the UI thread to be responsive to the user.

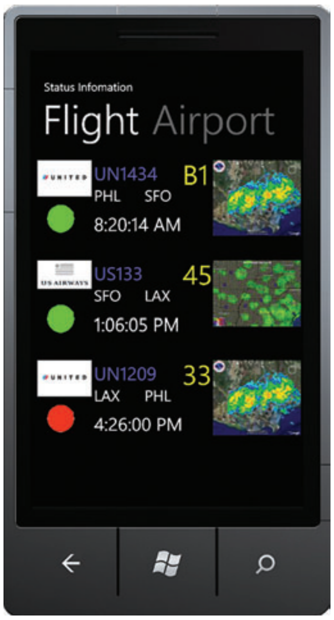


Figure 1 **The Flight Information Sample App**

In the case of our sample app, we're also making heavy use of images on the screen. The ListBox needs to actually download the image in order to display that data. While this seems OK, doing this work on the UI thread would block the user from any gesture input. Loading images on a background thread solves a number of problems both in terms of memory requirements and freeing up the UI thread, while at the same time making the application faster.

Everything that we display to the user must be rendered. Rendering requires layout, alignment and calculation to display correctly. As more layers are added to the UI, the calculation and overall rendering costs increase. Although Silverlight already virtualizes the UI, it doesn't virtualize the data that's being bound. This means that if we were to bind 10,000 items to our ListBox, Silverlight would instantiate all 10,000 ListItems before they were rendered.

Today's phones are extremely powerful compared to devices of just a few years ago.

Be aware of what you're data-binding and keep the bound set as small as possible. If you need to handle large sets of data-bind items, consider dynamically handling the rendering behind the scenes. This is true of desktop apps as well, of course—the impact of these choices is just amplified on a phone.

ValueConverters may have a drastic impact on rendering performance because they're defined using custom code and rendering cannot be pre-determined and cached ahead of actual element rendering and layout.

Dealing with Data

Next, we need to talk about data storage in Windows Phone 7. Let's just get straight to the point: There's no relational database engine available to developers. SQL Server Compact (SQL CE) is installed with the Windows Phone 7 OS, but currently there's no API available to developers. So creating a database to store application data—in our example, trip information—isn't going to work.

That said, there's a wide range of options for getting data to and from our application. A common approach is to use a cloud service such as Windows Azure for data persistence. There are many technologies available for building the service layer of your application, REST and SOAP being the most popular. SOAP is the first choice for many developers, but we think REST provides a more efficient and simpler-to-implement method of making data requests.

We employ a few methods that provide data to the application and that we can access by using REST expressions such as these:

```
/Trip/Create/PHL-BOS-SEA/xxxx/2010-04-10
/Flight/CheckStatus/US743
```

REST enables us to use either XML or JSON for a message format.

From the Web front-end perspective, we chose the ASP.NET MVC framework (asp.net/mvc) because it allows us to process the request and return any type of markup using a custom view.

Our sample application needs to handle both trip and flight information, so we create a FlightController and a TripController that intercept requests for this information:

```
// GET: /Flight/CheckStatus/US743
public ActionResult CheckStatusByFlight(
  string flightNumber) {
  return CheckStatus(flightNumber, DateTime.Now);
}

// GET: /Flight/RegisterInterest/US743/2010-04-12
public ActionResult CheckStatus(
  string flightNumber, DateTime date) {
  Flight f = new Flight(flightNumber, date);
  GetFlightStatus(f);
  return new XmlResultView<Flight>(f);
}
```

To provide simplified access methods and save a few bytes of bandwidth, if the date is today we might design a shortcut method for getting this data without implicitly specifying today's date.

Cached and Persistent Data

The flight status service is an element in our application that's not in our control and thus will be part of the performance puzzle. Because successful application may receive a considerable number of requests, it's important to think about a caching strategy.

Typically, the closer the flight is to departure, the higher the number of requests for its information can be expected. Higher numbers of near-concurrent requests can affect not only the performance of the application, but also the costs associated with storing and manipulating the data. As a general rule, Windows Azure applications accrue bandwidth charges both on request and return, and flight information services could also incur access charges. The amount of data returned will need to be no more than what's needed by the application.

The Windows Azure platform provides a wide range of options for data storage—from tables, blobs and queues to relational database-like storage via SQL Azure. We decided to use SQL Azure because it uses familiar SQL Server programming techniques and enables us to easily store and access both cached flight data and persistent trip information.

Figure 2 shows the simple storage layer we designed using the Entity Framework.

Returning Data

We return data to the client via our custom view. Because we're using ASP.NET MVC, each view needs to derive from ActionResult and implement ExecuteResult.

As mentioned earlier, we can provide either XML or JSON representations of the flight information via our REST service. Let's take a look at the XML option first. The serializer to produce XML requires a type, so we create a Generics class as shown in **Figure 3**.

We could just as easily work with JSON for our data. The only element of our solution that would change would be the contents of the ExecuteResult method. Using JsonResult, we can produce the JSON return from our service in just a few lines of code:

```
// Create the serializer
var result = new JsonResult();
// Enable the requests that originate from an HTTP GET
result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
// Set data to return
result.Data = _model;
// Write the data to the stream
result.ExecuteResult(context);
```

What about saving data to the actual device? It wouldn't make sense to force the application to pull data from the service every time the user needs access to trip information. Although there's no relational data store in Windows Phone 7, developers do have

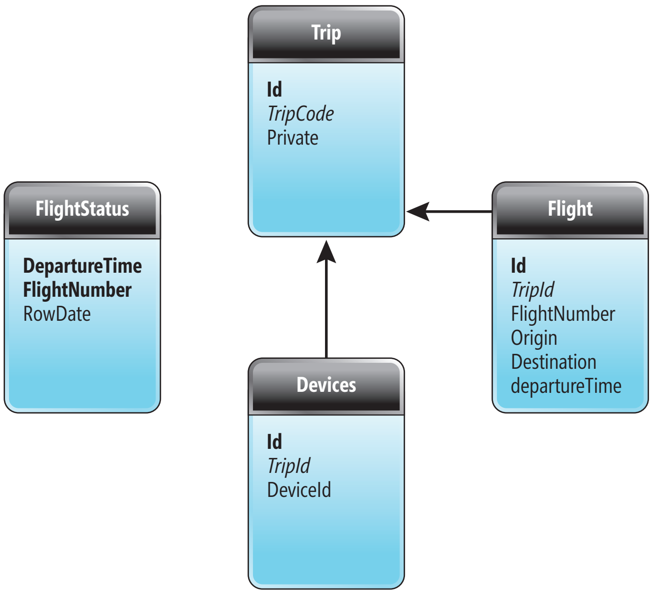


Figure 2 **Flight Data Storage Schema**

access to a feature called Isolated Storage. This works just like Silverlight 4 Isolated Storage, but without a size limit.

There are two main methods you need to save and retrieve data on the phone: SaveData and GetSavedData. Examples showing how we'd use these for the flight information app are shown in **Figure 4**.

Dealing with Network Failures

The networks used by mobile devices can have widely variable connectivity—sometimes becoming completely unavailable due to location, congestion or even users disconnecting manually (in the case of airplane mode, for instance). You have to accept this as a fact of life. As mobile application developers we must take this into consideration when building applications.

Be aware of what you're data-binding and keep the bound set as small as possible.

Another type of network failure is service layer failure. Many mobile applications consume data from third-party services. These services may not come with service-level agreements, which leaves your application at the mercy of the provider. In other words, it's out of your control, and you have to be prepared to deal with outages.

Regardless of the source of the network failure, you still need to provide the best user experience possible. You need to provide some level of functionality in the event of any type of network failure. For our flight status application this means that we want to allow the user to access as much information as possible even if network connectivity is lost from either the server or the client side.

There are numerous ways to achieve this. For now, we'll concentrate on three simple ways in which you could accomplish this: get the data while you can, cache data locally and cache data on a server you control.

Using Push Notifications

When the user enters trip information into the application, the information will be uploaded to a cloud service. The service will then keep polling the various services that provide its flight and weather data. It also looks for changes in the data over time, such as a flight status change or an airport that reports a delay.

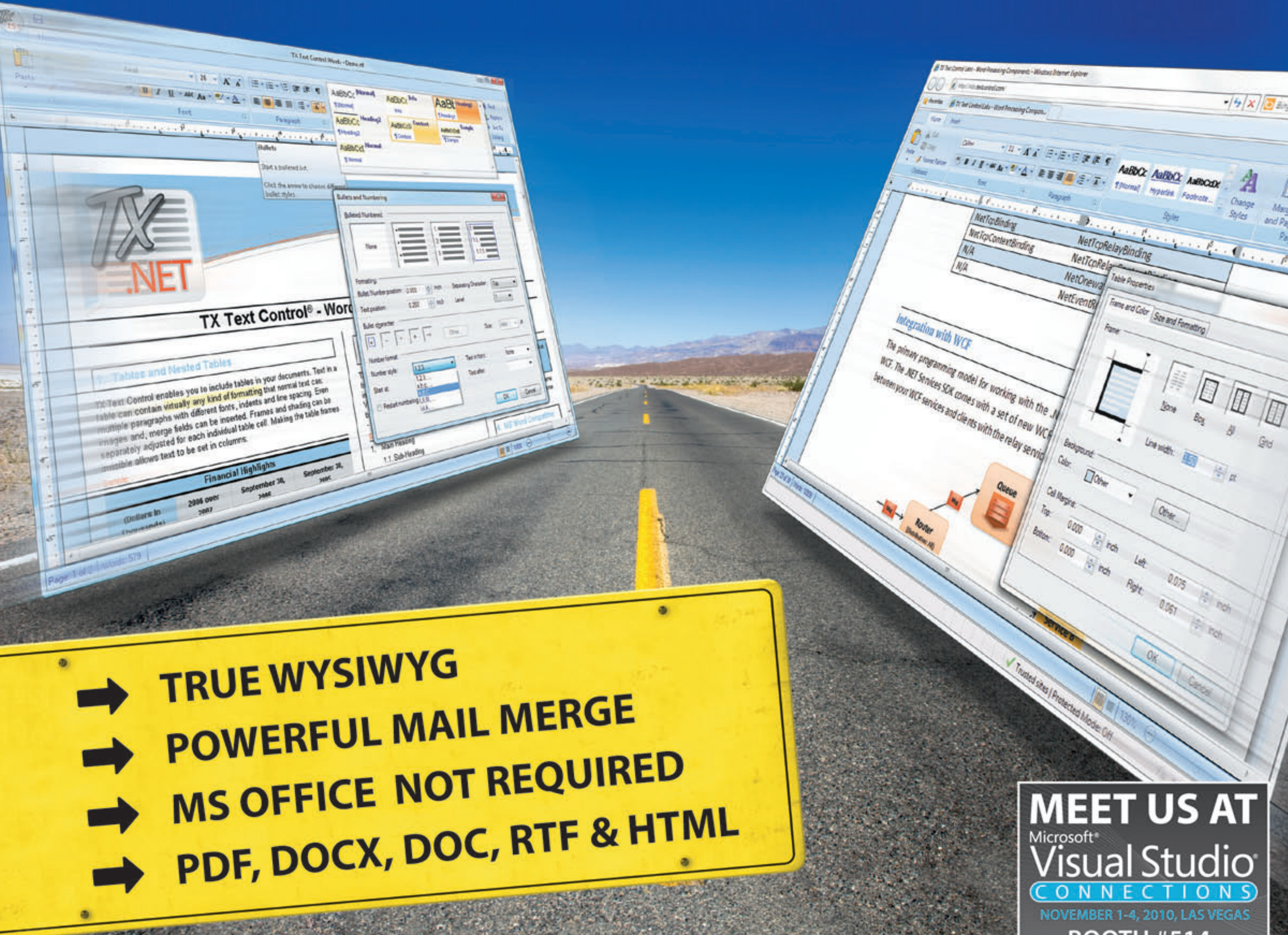
When a change is found, you want to get that information to the user as promptly and efficiently as possible. One way to do that is for the service to push the information to the client app. This will provide the user access to the most current set of available data the moment the data becomes available. Because the data was pushed to the client, the data is available even if the user loses his network connection.

We can accomplish this with our Windows Azure Service by using Windows Phone Push Notification. The Windows Phone Push Notification feature is made up of three components: monitor services, the Microsoft Push Notification Service and a message handling method.

WINDOWS FORMS / WPF / ASP.NET / ACTIVEX

WORD PROCESSING COMPONENTS

MILES BEYOND RICH TEXT



- TRUE WYSIWYG
- POWERFUL MAIL MERGE
- MS OFFICE NOT REQUIRED
- PDF, DOCX, DOC, RTF & HTML

MEET US AT
Microsoft® Visual Studio® CONNECTIONS
NOVEMBER 1-4, 2010, LAS VEGAS
BOOTH #514

TX TEXTCONTROL®
word processing components

Microsoft® Visual Studio®
PARTNER

Word Processing Components
for Windows Forms & ASP.NET

WWW.TEXTCONTROL.COM

TX Text Control Sales:
US +1 877 - 462 - 4772 (toll-free)
EU +49 421 - 4270671 - 0

Figure 3 **Serializing XML**

```
public class XmlResultView<T> : ActionResult {
  object _model = null;
  public XmlResultView(object model) {
    this._model = model;
  }

  public override void ExecuteResult(ControllerContext context) {
    // Create where to write
    MemoryStream mem = new MemoryStream();

    // Pack characters as compact as possible,
    // remove the decl, do not indent.
    XmlWriterSettings settings = new XmlWriterSettings() {
      Encoding = System.Text.Encoding.UTF8,
      Indent = false, OmitXmlDeclaration = true };
    XmlWriter writer = XmlTextWriter.Create(mem, settings);

    // Create a type serializer
    XmlSerializer ser = new XmlSerializer(typeof(T));

    // Write the model to the stream
    ser.Serialize(writer, _model);

    context.HttpContext.Response.OutputStream.Write(
      mem.ToArray(), 0, (int)mem.Length);
  }
}
```

A monitor service is a cloud service that constantly looks for new information for our application. We'll discuss this in more detail later.

The Push Notification Service is part of the Microsoft hosted services that are used to relay messages to Windows Phone 7 devices. This service is available to all Windows Phone 7 app developers.

The message handler method does what its name suggests: It simply receives messages from the Push Notification Service.

There are three default notification types in Windows Phone 7: Tile, Push and Toast notifications. Notifications are an important part of the user experience and you need to consider their use carefully. Repetitive or intrusive notifications can degrade performance of your application and others running on the device. They can also annoy users. Consider the frequency at which notifications are sent and the types of events that you want to get the attention of users.

In Windows Phone 7, notifications are delivered via batching, so the transaction may not be instant. The timeliness of the notification is not guaranteed and the decision on how to deliver the notification to the client is handled by the service; the service does its best to determine how quickly it can deliver the message to the phone.

The Windows Azure platform provides a wide range of options for data storage.

The workflow for push notifications is:

1. Client app requests a channel connection to the Push Notification Service.
2. The Push Notification Service responds with the channel URI.
3. The client app sends a message to the monitoring service with the Push Notification Service channel URI as well as a payload.
4. When the monitoring service detects a change of information (flight cancellations, flight delays or weather alerts in our sample app) it sends a message to the Push Notification Service.
5. The Push Notification Service relays the message to the Windows Phone 7 device.
6. The message handler processes the message on the device.

Caching Data Locally

Another way to make data available to your app is to cache it locally so there's always some data in the UI. Then you can use other means in the background to update the local data (if possible). The upside of this method is that the application can load and be usable quickly even if updating information has to occur asynchronously behind the scenes.

In a nutshell, you use Isolated Storage to save the most recent set of data. When the application opens, it immediately grabs any data available in local Isolated Storage and renders it. In the meantime, the application calls the Windows Azure Service for updated information. If new information is found, it's serialized and transferred to the device, Isolated Storage gets updated, and you render the UI again with updated information. For a better user experience, you probably want to indicate in the UI what time and date the information was refreshed.

The networks used by mobile devices can have widely variable connectivity.

On a side note, if the application is using the Model-View-ViewModel (MVVM) design pattern, the update to the UI can happen automatically via Silverlight data-binding features. For more information on MVVM and Silverlight, see Robert McCarter's article, "Problems and Solutions with Model-View-ViewModel," at msdn.microsoft.com/magazine/ff798279.

Caching Data on Your Server

There's a middle ground between pushing data directly to your app as it becomes available and storing data on the device: grabbing data from third-party services and caching it in your cloud app until the Windows Phone 7 app requests it.

This technique requires a new layer of abstraction in your application. In essence, the goal here is to remove the dependency of a third-party service from your application. Your service pulls and caches the data for any third-party service dependencies. If the third-party service goes down, you'll at least have some data in the cache that you can provide to the application on devices.

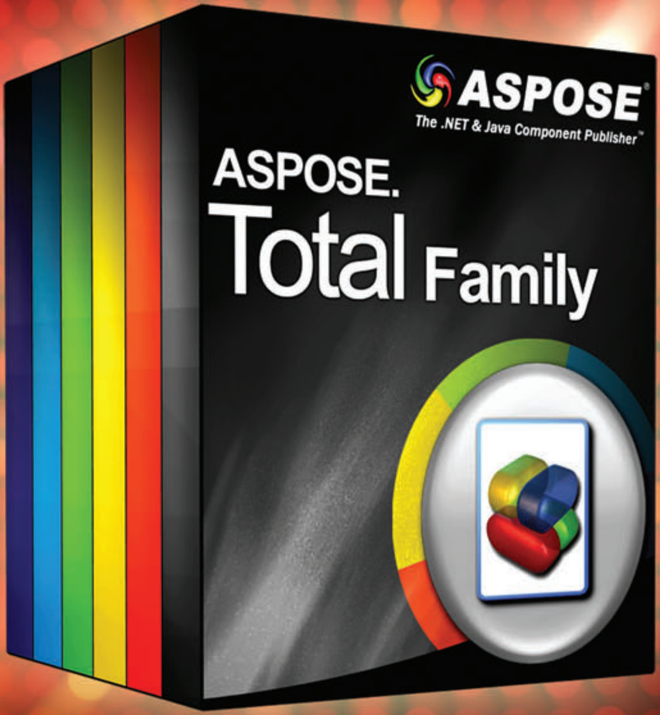
A service like this could be easily cloned or extended to pull the data from various services, thus reducing your dependency on any one vendor or data source, which makes changing vendors a lot easier.

For more information about setting up data-focused solutions in Windows Azure, see "Fueling Your Application's Engine with Windows

ASPOSE®

Your File Format Experts

Aspose Total Product Family



Every Aspose Component combined in one powerful suite!

Serving more than 50% of the Fortune 100 Companies!

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) ++

The TOTAL Solutions for .NET, Java, SQL Server Rendering Extensions, SharePoint and JasperReports Exporters.

Get your FREE evaluation copy at www.aspose.com

ASPOSE
Your File Format Experts

US Sales: 1.888.277.6734 · EU Sales: +44 (0)800 098 8425 · email: sales@aspose.com

Figure 4 **Saving and Retrieving Local Data**

```
public static IEnumerable<Trips> GetSavedData() {
  IEnumerable<Trips> trips = new List<Trips>();
  try {
    using (var store =
      IsolatedStorageFile.GetUserStoreForApplication()) {
      string offlineData =
        Path.Combine("TravelBuddy", "Offline");
      string offlineDataFile =
        Path.Combine(offlineData, "offline.xml");

      IsolatedStorageFileStream dataFile = null;

      if (store.FileExists(offlineDataFile)) {
        dataFile =
          store.OpenFile(offlineDataFile, FileMode.Open);
        DataContractSerializer ser =
          new DataContractSerializer(
          typeof(IEnumerable<Trips>));

        // Deserialize the data and read it
        trips =
          (IEnumerable<Trips>)ser.ReadObject(dataFile);
        dataFile.Close();
      }
      else
        MessageBox.Show("No data available");
    }
  }

  catch (IsolatedStorageException) {
    // Fail gracefully
  }

  return trips;
}
```

```
public static void SaveOfflineData(IEnumerable<Trips> trip) {
  try {
    using (var store =
      IsolatedStorageFile.GetUserStoreForApplication()) {

      // Create three directories in the root.
      store.CreateDirectory("TravelBuddy");

      // Create three subdirectories under MyApp1.
      string offlineData =
        Path.Combine("TravelBuddy", "Offline");

      if (!store.DirectoryExists(offlineData))
        store.CreateDirectory(offlineData);

      string offlineDataFile =
        Path.Combine(offlineData, "offline.xml");
      IsolatedStorageFileStream dataFile =
        dataFile = store.OpenFile(offlineDataFile,
        FileMode.OpenOrCreate);

      DataContractSerializer ser =
        new DataContractSerializer(typeof(IEnumerable<Trip>));
      ser.WriteObject(dataFile, trip);

      dataFile.Close();
    }
  }

  catch (IsolatedStorageException) {
    // Fail gracefully
  }
}
```

Azure Storage" by Kevin Hoffman and Nathan Dudek (msdn.microsoft.com/magazine/ee335721). In addition, although not directly focused on Windows Phone 7 scenarios, Paul Stubb's article, "Create a Silverlight 4 Web Part for SharePoint 2010," is good reading on data-bound design for Silverlight and Web services (msdn.microsoft.com/magazine/ff956224).

Another way to make data available to your app is to cache it locally.

Monitoring Service

As mentioned earlier, the notification feature is an important part of our flight status application. This feature is actually composed of several different services within the application. Perhaps most important to the usefulness of the app, the monitoring service periodically polls third-party data services and relays information like flight delays, airport delays and weather alerts back to the device.

In our application, the monitoring service reads the current list of flights and airport codes and uses this information to collect relevant data. This information is then stored back in the SQL Azure database as a cache entry so that it can be retrieved by the /Flight/CheckStatus service shown earlier. Our monitoring service is implemented with a Windows Azure Worker Role. The main goal on this Worker Role is to pull status information on flight delays and airport status for every user's flight collection. The frequency of the update pull increases as it approaches the scheduled flight departure time.

For some ideas about how you could implement such a service, be sure to check out the Azure Publish-Subscribe project on CodePlex (azurepubsub.codeplex.com) or read Joseph Fultz's blog post, "Migrating Windows Service to Azure Worker Role: Image Conversion Example Using Storage" (bit.ly/aKY8iv).

Putting It All Together

Hopefully we've given you a broad overview of the issues you need to consider when designing a data-driven Windows Phone 7 application. UI responsiveness, as well as time access to your data sources, play into making a great user experience for your app.

To dig a bit deeper, start with Joshua Partlow's article, "Getting Started with Windows Phone Development Tools" (msdn.microsoft.com/magazine/gg232764). You'll also want to see the article, "Developing and Deploying Windows Azure Apps in Visual Studio 2010," by Jim Nakashima, Hani Atassi and Danny Thorpe (msdn.microsoft.com/magazine/ee336122).

To put your Windows Azure and Windows Phone 7 development together, take a look at Ramon Arjona's article, "Windows Phone and the Cloud—an Introduction" (msdn.microsoft.com/magazine/ff872395). ■

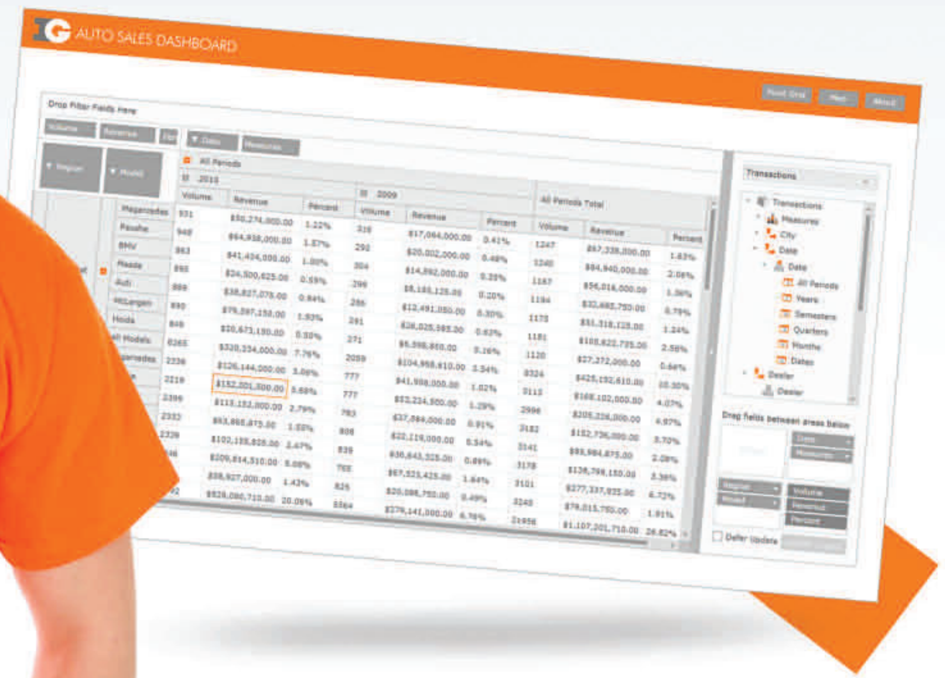
DANILO DIAZ *is a developer evangelist for the Microsoft Mid-Atlantic State district. In this role, he helps developers understand Microsoft product offerings and strategy.*

MAX ZILBERMAN *is an architect evangelist in the NYC and Mid-Atlantic States districts. Prior to joining Microsoft, Zilberman held various senior technical positions at a top-tier health insurer.*

THANKS to the following technical expert for reviewing this article: *Ramon Arjona*

DEVELOP

Rich Business Intelligence Applications in WPF and Silverlight



	All Periods								
	2010			2009			All Periods Total		
	Volume	Revenue	Percent	Volume	Revenue	Percent	Volume	Revenue	Percent
Mercedes	931	$50,274,000.00	1.22%	318	$17,064,000.00	0.41%	1247	$67,338,000.00	1.63%
Porsche	948	$64,938,000.00	1.57%	292	$20,002,000.00	0.48%	1240	$84,940,000.00	2.06%
BMW	863	$41,424,000.00	1.00%	304	$14,592,000.00	0.35%	1167	$56,016,000.00	1.36%
Mazda	895	$24,500,625.00	0.59%	299	$8,185,125.00	0.20%	1194	$32,685,750.00	0.79%
Audi	889	$38,827,075.00	0.94%	286	$12,491,050.00	0.30%	1175	$51,318,125.00	1.24%
McLaren	890	$79,597,150.00	1.93%	291	$26,025,585.00	0.63%	1181	$105,622,735.00	2.56%
Honda	849	$20,673,150.00	0.50%	271	$6,598,850.00	0.16%	1120	$27,272,000.00	0.66%
All Models	6265	$320,234,000.00	7.76%	2059	$104,958,610.00	2.54%	8324	$425,192,610.00	10.30%
ercedes	2336	$126,144,000.00	3.06%	777	$41,958,000.00	1.02%	3113	$168,102,000.00	4.07%
	2219	$152,001,500.00	3.68%	777	$53,224,500.00	1.29%	2996	$205,226,000.00	4.97%
	2399	$115,152,000.00	2.79%	783	$37,584,000.00	0.91%	3182	$152,736,000.00	3.70%
	2333	$63,865,875.00	1.55%	808	$22,119,000.00	0.54%	3141	$85,984,875.00	2.08%
	2339	$102,135,825.00	2.47%	839	$36,643,325.00	0.89%	3178	$136,798,150.00	3.30%
	46	$209,814,510.00	5.08%	765	$67,523,425.00	1.64%	3101	$277,337,935.00	6.72%
		$58,927,000.00	1.43%	825	$20,088,750.00	0.49%	3243	$79,015,750.00	1.91%
	92	$828,090,710.00	20.06%	5564	$279,141,000.00	6.76%	21856	$1,107,201,710.00	26.82%

Robust Pivot Grids for WPF and Silverlight let your users analyze data to make key business decisions. Visit infragistics.com to try it today!

NetAdvantage®
for Silverlight Data Visualization

NetAdvantage®
for WPF Data Visualization

Infragistics®

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91 80 4151 8042
@infragistics

Behavior-Driven Development with SpecFlow and WatiN

Brandon Satrom

As automated unit testing becomes more ubiquitous in software development, so does the adoption of various test-first methods. These practices each present a unique set of opportunities and challenges to development teams, but all strive to establish a "testing as design" mindset with practitioners.

For much of the test-first era, however, the method for expressing the behavior of a user has been through unit tests written in the language of the system—a language disconnected from that of the user. With the advent of Behavior-Driven Development (BDD) techniques, this dynamic is changing. Using BDD techniques you can author automated tests in the language of the business, all while maintaining a connection to your implemented system.

Of course, a number of tools have been created to help you implement BDD in your development process. These include Cucumber in Ruby and SpecFlow and WatiN for the Microsoft .NET Framework. SpecFlow helps you write and execute specifications within Visual Studio, while WatiN enables you to drive the browser for automated end-to-end system testing.

In this article, I'll provide a brief overview of BDD and then explain how the BDD cycle wraps the traditional Test-Driven Development (TDD) cycle with feature-level tests that drive unit-level implementation. Once I've laid the groundwork for test-first methods, I'll introduce SpecFlow and WatiN and show you examples of how these tools can be used with MSTest to implement BDD for your projects.

This article discusses:

- Automated testing basics
- Using SpecFlow and WatiN
- Your first acceptance test
- Writing unit tests to implement steps

Technologies discussed:

Visual Studio 2010, SpecFlow, WatiN

Code download available at:

code.msdn.microsoft.com/mag201012BDD

A Brief History of Automated Testing

One of the most valuable practices to emerge from the Agile Software movement is an automated, test-first development style, often referred to as Test-Driven Development, or TDD. A key tenet of TDD is that test creation is as much about design and development guidance as it is about verification and regression. It's also about using the test to specify a unit of required functionality, and using that test to then write only the code needed to deliver that functionality. Therefore, the first step in implementing any new functionality is to describe your expectations with a failing test (see **Figure 1**).

Many developers and teams have had great success with TDD. Others have not, and find that they struggle with managing the process over time, especially as the volume of tests begins to grow and the flexibility of those tests begins to degrade. Some aren't sure how to start with TDD, while others find TDD easy to initiate, only to watch it abandoned as deadlines near and large backlogs loom. Finally, many interested developers meet resistance to the practice within their organizations, either because the word "test" implies a function that belongs on another team or because of the false perception that TDD results in too much extra code and slows down projects.

Steve Freeman and Nat Pryce, in their book, "Growing Object-Oriented Software, Guided by Tests" (Addison-Wesley Professional, 2009), note that "traditional" TDD misses some of the benefits of true test-first development:

"It is tempting to start the TDD process by writing unit tests for classes in the application. This is better than having no tests at all and can catch those basic programming errors that we all know but find so hard to avoid … But a project with only unit tests is missing out on critical benefits of the TDD process. We've seen projects with high-quality, well unit-tested code that turned out not to be called from anywhere, or that could not be integrated with the rest of the system and had to be rewritten."

In 2006, Dan North documented many of these challenges in an article in *Better Software* magazine (blog.dannorth.net/introducing-bdd). In his article, North described a series of practices that he had adopted over the prior three years while in the trenches with testing. While still TDD by definition, these practices led North to adopt a more analysis-centric view of testing and to coin the term Behavior-Driven Development to encapsulate this shift.

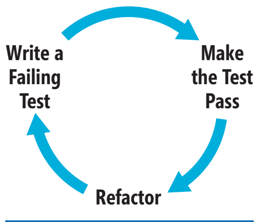


Figure 1 **The Test-Driven Development Cycle**

One popular application of BDD attempts to extend TDD by tightening the focus and process of creating tests through Acceptance Tests, or executable specifications. Each specification serves as an entry point into the development cycle and describes, from the user's point of view and in a step-by-step form, how the system should behave. Once written, the developer uses the specification and their existing TDD process to implement just enough production code to yield a passing scenario (see **Figure 2**).

Where Design Begins

BDD is considered by many a superset of TDD, not a replacement for it. The key difference is the focus on initial design and test creation. Rather than focusing on tests against units or objects, as with TDD, I focus on the goals of my users and the steps they take to achieve those goals. Because I'm no longer starting with tests of small units, I'm less inclined to speculate on fine-grained usage or design details. Rather, I'm documenting executable specifications that prove out my system. I still write unit tests, but BDD encourages an outside-in approach that starts with a full description of the feature to be implemented.

Let's look at an example of the difference. In a traditional TDD practice, you could write the test in **Figure 3** to exercise the Create method of a CustomersController.

With TDD, this tends to be one of the first tests I write. I design a public API to my CustomersController object by setting expectations of how it will behave. With BDD I still create that test, but not at first. Instead, I elevate the focus to feature-level functionality by writing something more like **Figure 4**. I then use that scenario as guidance toward implementing each unit of code needed to make this scenario pass.

This is the outer loop in **Figure 2**, the failing Acceptance Test. Once this test has been created and fails, I implement each step of each scenario in my feature by following the inner TDD loop depicted in **Figure 2**. In the case of the CustomersController in **Figure 3**, I'll write this test once I reach the proper step in my feature, but before I implement the controller logic needed to make that step pass.

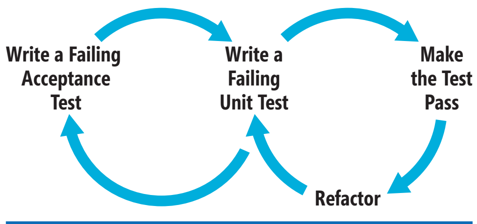


Figure 2 **The Behavior-Driven Development Cycle**

BDD and Automated Testing

From the start, the BDD community has sought to provide the same level of automated testing with Acceptance Tests that has been the norm in unit testing for some time. One notable example is Cucumber (cukes.info), a Ruby-based testing tool that emphasizes the creation of feature-level Acceptance Tests written in a "business-readable, domain-specific language."

Cucumber tests are written using User Story syntax for each feature file and a Given, When, Then (GWT) syntax for each scenario. (For details on User Story syntax, see c2.com/cgi/wiki?UserStory.) GWT describes the current context of the scenario (Given), the actions taken as a part of the test (When) and the expected, observable results (Then). The feature in **Figure 4** is an example of such syntax.

In Cucumber, user-readable feature files are parsed, and each scenario step is matched to Ruby code that exercises the public interfaces of the system in question and determines if that step passes or fails.

In recent years, innovations enabling the use of scenarios as automated tests have extended into the .NET Framework ecosystem. Developers now have tools that enable specifications to be written using the same structured English syntax that Cucumber utilizes, and which can then use those specifications as tests that exercise the code. BDD testing tools like SpecFlow (specflow.org), Cuke4Nuke (github.com/richardlawrence/Cuke4Nuke) and others enable you to create executable specifications first in the process, leverage those specifications as you build out functionality and end with a documented feature that's tied directly to your development and testing process.

Getting Started with SpecFlow and WatiN

In this article, I'll utilize SpecFlow to test a Model-View-Controller (MVC) application. To get started with SpecFlow, you'll first need to download and install it. Once SpecFlow is installed, create a new ASP.NET MVC application with a unit test project. I prefer that my unit test project contain only unit tests (controller tests, repository tests and so on), so I also create an AcceptanceTests test project for my SpecFlow tests.

Once you've added an AcceptanceTests project and added references to the TechTalk.SpecFlow assembly, add a new Feature using the Add | New Item templates that SpecFlow creates on installation and name it CreateCustomer.feature.

Notice that the file is created with a .feature extension, and that Visual Studio recognizes this as a supported file, thanks to SpecFlow's integrated tooling. You may also notice that your feature file has a related .cs code-behind file. Each time you save a .feature file, SpecFlow parses the file and converts the text in that file into a test fixture. The code in the associated .cs file represents that test fixture, which is the code that's actually executed each time you run your test suite.

By default, SpecFlow uses NUnit as its test-runner, but it also supports MSTest with a simple configuration change. All

Figure 3 **Unit Test for Creating a Customer**

```
[TestMethod]
public void PostCreateShouldSaveCustomerAndReturnDetailsView() {
  var customersController = new CustomersController();
  var customer = new Customer {
    Name = "Hugo Reyes",
    Email = "hreyes@dharmainitiative.com",
    Phone = "720-123-5477"
  };

  var result = customersController.Create(customer) as ViewResult;

  Assert.IsNotNull(result);
  Assert.AreEqual("Details", result.ViewName);
  Assert.IsInstanceOfType(result.ViewData.Model, typeof(Customer));

  customer = result.ViewData.Model as Customer;
  Assert.IsNotNull(customer);
  Assert.IsTrue(customer.Id > 0);
}
```

you need to do is add an app.config file to your test project and add the following elements:

```
<configSections>
  <section name="specFlow"
    type="TechTalk.SpecFlow.Configuration.ConfigurationSectionHandler,
TechTalk.SpecFlow"/>
</configSections>
<specFlow>
  <unitTestProvider name="MsTest" />
</specFlow>
```

Your First Acceptance Test

When you create a new feature, SpecFlow populates that file with default text to illustrate the syntax used to describe a feature. Replace the default text in your CreateCustomer.feature file with the text in **Figure 4**.

Each feature file has two parts. The first part is the feature name and description at the top, which uses User Story syntax to describe the role of the user, the user's goal, and the types of things the user needs to be able to do to achieve that goal in the system. This section is required by SpecFlow to auto-generate tests, but the content itself is not used in those tests.

The second part of each feature file is one or more scenarios. Each scenario is used to generate a test method in the associated .feature.cs file, as shown in **Figure 5**, and each step within a scenario is passed to the SpecFlow test runner, which performs a RegEx-based match of the step to an entry in a SpecFlow file called a Step Definition file.

Once you've defined your first feature, Press Ctrl+R,T to run your SpecFlow tests. Your CreateCustomer test will fail as inconclusive because SpecFlow cannot find a matching step definition for the first step in your test (see **Figure 6**). Notice how the exception is reported in the actual .feature file, as opposed to the code-behind file.

Because you haven't yet created a Step Definition file, this exception is expected. Click OK on the exception dialog and look for the CreateABasicCustomerRecord test in the Visual Studio Test Results window. If a matching step isn't found, SpecFlow uses your feature file to generate the code you need in your step definition file, which you can copy and use to begin implementing those steps.

In your AcceptanceTests project, create a step definition file using the SpecFlow Step Definition template and name it CreateCustomer.cs. Then copy the output from SpecFlow into the class. You'll notice that each method is decorated with a SpecFlow attribute that designates

Figure 4 **Feature-level Specification**

```
Feature: Create a new customer
  In order to improve customer service and visibility
  As a site administrator
  I want to be able to create, view and manage customer records

Scenario: Create a basic customer record
  Given I am logged into the site as an administrator
  When I click the "Create New Customer" link
  And I enter the following information
    | Field | Value                       |
    | Name  | Hugo Reyes                  |
    | Email | hreyes@dharmainitiative.com |
    | Phone | 720-123-5477                |
  And I click the "Create" button
  Then I should see the following details on the screen:
    | Value                       |
    | Hugo Reyes                  |
    | hreyes@dharmainitiative.com |
    | 720-123-5477                |
```

the method as a Given, When or Then step, and provides the RegEx used to match the method to a step in the feature file.

Integrating WatiN for Browser Testing

Part of the goal with BDD is to create an automated test suite that exercises as much end-to-end system functionality as possible. Because I'm building an ASP.NET MVC application, I can use tools that help script the Web browser to interact with the site.

One such tool is WatiN, an open source library for automating Web browser testing. You can download WatiN from watin.sourceforge.net and add a reference to WatiN.Core to your Acceptance Tests project to use it.

Figure 5 **Test Method Generated by SpecFlow**

```
public virtual void CreateABasicCustomerRecord() {
  TechTalk.SpecFlow.ScenarioInfo scenarioInfo =
    new TechTalk.SpecFlow.ScenarioInfo(
    "Create a basic customer record", ((string[])(null)));

  this.ScenarioSetup(scenarioInfo);
  testRunner.Given(
    "I am logged into the site as an administrator");
  testRunner.When("I click the \"Create New Customer\" link");

  TechTalk.SpecFlow.Table table1 =
    new TechTalk.SpecFlow.Table(new string[] {
    "Field", "Value"});
  table1.AddRow(new string[] {
    "Name", "Hugo Reyesv"});
  table1.AddRow(new string[] {
    "Email", "hreyes@dharmainitiative.com"});
  table1.AddRow(new string[] {
    "Phone", "720-123-5477"});

  testRunner.And("I enter the following information",
    ((string)(null)), table1);
  testRunner.And("I click the \"Create\" button");

  TechTalk.SpecFlow.Table table2 =
   new TechTalk.SpecFlow.Table(new string[] {
  "Value"});
  table2.AddRow(new string[] {
    "Hugo Reyes"});
  table2.AddRow(new string[] {
    "hreyes@dharmainitiative.com"});
  table2.AddRow(new string[] {
    "720-123-5477"});
  testRunner.Then("I should see the following details on screen:",
    ((string)(null)), table2);
  testRunner.CollectScenarioErrors();
}
```

Microsoft Dynamics ERP

A New Evolution, a New Opportunity with Microsoft Dynamics AX

Q What is Microsoft Dynamics?
A Microsoft Dynamics is a line of ERP and CRM applications developed by the Microsoft Business Solutions group within Microsoft. You can read more at **http://www.microsoft.com/dynamics**.

Q What is Microsoft Dynamic AX?
A Microsoft Dynamics AX is a Microsoft enterprise resource planning product that provides solutions for industries like distribution, manufacturing, professional services, retail, and public sector. **http://www.microsoft.com/dynamics/en/us/products/ax-overview.aspx**

Q Why is this interesting for developers?
A Microsoft Dynamics AX is an application built on the Microsoft stack. Developers can leverage their skills in the .NET Framework to extend the solution into specific industry vertical.

Q How is the .NET Framework is leveraged?
A For example, the Windows Workflow foundation is the orchestration engine coordinating business processes in Microsoft Dynamics AX. The Windows Communication Foundation provides a unified programming model for leveraging business processes through services.

Q How is the application lifecycle managed?
A Microsoft Dynamics AX is integrated with Visual Studio Team System allowing developers to manage their solution with standard Microsoft tools.

Q Is there a new release coming?
A Yes. A new version of Microsoft Dynamics AX is expected in 2011.

Q Is there somewhere that I could go and learn about the upcoming release?
A Yes. We are hosting a pre-release conference called the Microsoft Dynamics AX Technical Conference 2011 focused on helping developers get their solutions prepared for the next release.

Q What can people expect when attending this conference?
A We have a great lineup. There will be over 65 sessions covering the upcoming release, with time set aside to collaborate with the development team in Chalk and Talk and Ask the Experts sessions. Microsoft Dynamics AX Certified Trainers will host instructor-led labs.

Microsoft Dynamics AX
TECHNICAL CONFERENCE 2011
Learn | Inspire | Innovate
January 17 - 20, 2011 • Redmond, WA

Also, partners and customers can network with the development team, not only from Microsoft Dynamics AX, but with others that work on the core technologies that AX builds on.

Q What topics will be covered?
A With all of the innovation in this new version there is something for everyone. Topics cover developer tools, database modeling, application component design, business intelligence tools, office integration around Microsoft SharePoint and the core Microsoft Office products, services integration, IT management, performance, and application lifecycle management.

Q Sounds great! How do I sign up?
A We are excited to be able to put on this event, as we think it is our little version of PDC for developers or TechEd for the IT folks, but just focused on Microsoft Dynamics AX and specifically the next release.

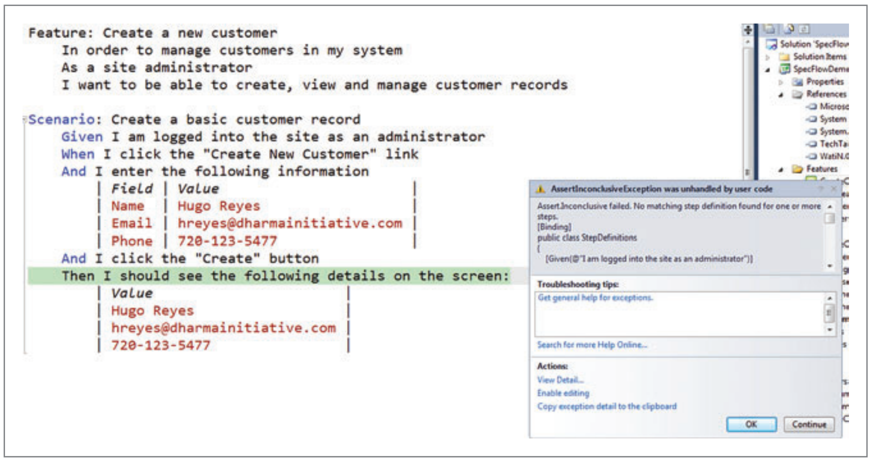
Register at **http://www.microsoft.com/dynamics/daxconf2011/msdn**. And be sure to check out the sample of the session content. Look for the full catalog by the end of October.

Experience How. Get Involved.

Find out more at **http://www.microsoft.com/dynamics/daxconf2011/MSDN**

Want to get involved? Email: **daxconf@microsoft.com.**

Microsoft



```
Feature: Create a new customer
    In order to manage customers in my system
    As a site administrator
    I want to be able to create, view and manage customer records

Scenario: Create a basic customer record
    Given I am logged into the site as an administrator
    When I click the "Create New Customer" link
    And I enter the following information
        | Field | Value                        |
        | Name  | Hugo Reyes                   |
        | Email | hreyes@dharmainitiative.com  |
        | Phone | 720-123-5477                 |
    And I click the "Create" button
    Then I should see the following details on the screen:
        | Value                        |
        | Hugo Reyes                   |
        | hreyes@dharmainitiative.com  |
        | 720-123-5477                 |
```

```
AssertInconclusiveException was unhandled by user code

Assert.Inconclusive failed. No matching step definition found for one or more steps.
[Binding]
public class StepDefinitions
{
  [Given(@"I am logged into the site as an administrator")]
```

Troubleshooting tips:
Get general help for exceptions.
Search for more Help Online...
Actions:
View Detail...
Enable editing
Copy exception detail to the clipboard
OK
Continue

Figure 6 **SpecFlow Cannot Find a Step Definition**

The primary way you interact with WatiN is through a browser object—either IE() or FireFox(), depending on your browser of choice—that provides a public interface to control an instance of an installed browser. Because you need to walk the browser through several steps in a scenario, you need a way to pass the same browser object between steps in the step definition class. To handle this, I usually create a WebBrowser static class as part of my AcceptanceTests project, and use that class to work with the WatiN IE object and the ScenarioContext that SpecFlow uses to store state between steps in a scenario:

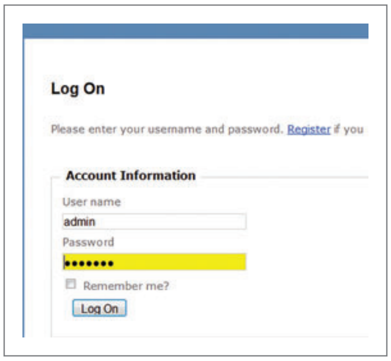
```
public static class WebBrowser {
  public static IE Current {
    get {
      if (!ScenarioContext.Current.ContainsKey("browser"))
        ScenarioContext.Current["browser"] = new IE();
      return ScenarioContext.Current["browser"] as IE;
    }
  }
}
```

The first step you'll need to implement in CreateCustomer.cs is the Given step, which begins the test by logging the user into the site as an administrator:

```
[Given(@"I am logged into the site as an administrator")]
public void GivenIAmLoggedIntoTheSiteAsAnAdministrator() {
  WebBrowser.Current.GoTo(http://localhost:24613/Account/LogOn);

  WebBrowser.Current.TextField(Find.ByName("UserName")).TypeText("admin");
  WebBrowser.Current.TextField(Find.ByName("Password")).TypeText("pass123");
  WebBrowser.Current.Button(Find.ByValue("Log On")).Click();

  Assert.IsTrue(WebBrowser.Current.Link(Find.ByText("Log Off")).Exists);
}
```

Remember that the Given portion of a scenario is for setting up the context of the current test. With WatiN, you can have your test drive and interact with the browser to implement this step.

For this step, I use WatiN to open Internet Explorer, navigate to the Log On page of the site, fill out the User name and Password textboxes, and then click the Log On button on the screen. When I run the tests again, an Internet Explorer window will open automatically and I can observe WatiN in action as it interacts with the site, clicking links and entering text (see **Figure 7**).



Log On

Please enter your username and password. Register if you

Account Information

User name

admin

Password

•••••••

☐ Remember me?

Log On

Figure 7 **The Browser on Autopilot with WatiN**

The Given step will now pass and I'm a step closer to implementing the feature. SpecFlow will now fail on the first When step because the step is not yet implemented. You can implement it with the following code:

```
[When("I click the \" (.*)\" link")]
public void WhenIClickALinkNamed(string linkName) {
  var link = WebBrowser.Link(Find.ByText(linkName));

  if (!link.Exists)
    Assert.Fail(string.Format(
      "Could not find {0} link on the page",
linkName));

  link.Click();
}
```

Now, when I run the tests again, they fail because WatiN cannot find a link with the text "Create New Customer" on the page. By simply adding a link with that text to the homepage, the next step will pass.

Sensing a pattern yet? SpecFlow encourages the same Red-Green-Refactor methodology that's a staple of test-first development methods. The granularity of each step in a feature acts like virtual blinders for implementation, encouraging you to implement only the functionality that you need to make that step pass.

But what about TDD inside of the BDD process? I'm only working at the page level at this point, and I have yet to implement the functionality that actually creates the customer record. For the sake of brevity, let's implement the rest of the steps now (see **Figure 8**).

I re-run my tests, and things now fail because I don't have a page to enter customer information. To allow customers to be created, I need a Create Customer View page. In order to deliver such a view in ASP.NET MVC, I need a CustomersController that delivers that view. I now need new code, which means I'm stepping from the outer loop of BDD and into the inner loop of TDD, as shown back in **Figure 2**.

The first step is to create a failing unit test.

Writing Unit Tests to Implement Steps

After creating a CustomerControllersTests test class in the UnitTest project, you need to create a test method that exercises the functionality to be exposed in the CustomersController. Specifically, you want to create a new instance of the Controller, call its Create method and ensure that you receive the proper View and Model in return:

```
[TestMethod]
public void GetCreateShouldReturnCustomerView() {
  var customersController = new CustomersController();
  var result = customersController.Create() as ViewResult;

  Assert.AreEqual("Create", result.ViewName);
  Assert.IsInstanceOfType(
    result.ViewData.Model, typeof(Customer));
}
```

This code doesn't yet compile because you haven't created CustomersController or its Create method. Upon creating that controller and an empty Create method, the code compiles and the test now fails, which is the desired next step. If you complete the Create method, the test now passes:

```
public ActionResult Create() {
  return View("Create", new Customer());
}
```

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

ActiveReports 6 | from $685.02

GrapeCity

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

- Now supports Windows Server 2008, 64Bit & IE8.0
- First Flash based Report Viewer for end users
- Supports PDF Digital Signatures, RSS Bar Codes and external stylesheets
- Features an easy-to-use Visual Studio report designer and a powerful API
- Offers seamless run-time deployment, royalty free

TX Text Control .NET and .NET Server | from $499.59

TX TEXT CONTROL word processing components

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

FusionCharts | from $195.02

InfoSoft Global empowering human thoughts

Interactive Flash & JavaScript (HTML5) charts for web apps.

- Liven up your web applications using animated & data-driven charts
- Create AJAX-enabled charts with drill-down capabilities in minutes
- Export charts as images/PDF and data as CSV from charts itself
- Create gauges, dashboards, financial charts and over 550 maps
- Trusted by over 17,000 customers and 330,000 users in 110 countries

Spread for Windows Forms | from $959.04

GrapeCity

A comprehensive Excel compatible spreadsheet component for Windows Forms applications.

- Speed development with Spreadsheet Designers, Quick-start Wizard and Chart Designers
- Automatic completion - provide "type ahead" from user input to cell
- Features built-in tool to design charts with 85 different styles
- Preserves Excel files and restores unsupported functionality
- Includes sleek new predefined skins and the ability to create custom skins

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

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

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

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

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

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

Figure 8 **Remaining Steps in the Step Definition**

```
[When(@"I enter the following information")]
public void WhenIEnterTheFollowingInformation(Table table) {
  foreach (var tableRow in table.Rows) {
    var field = WebBrowser.TextField(
      Find.ByName(tableRow["Field"]));

    if (!field.Exists)
      Assert.Fail(string.Format(
        "Could not find {0} field on the page", field));
    field.TypeText(tableRow["Value"]);
  }
}

[When("I click the \"(.*)\" button")]
public void WhenIClickAButtonWithValue(string buttonValue) {
  var button = WebBrowser.Button(Find.ByValue(buttonValue));

  if (!button.Exists)
    Assert.Fail(string.Format(
      "Could not find {0} button on the page", buttonValue));

  button.Click();
}

[Then(@"I should see the following details on the screen:")]
public void ThenIShouldSeeTheFollowingDetailsOnTheScreen(
  Table table) {
  foreach (var tableRow in table.Rows) {
    var value = tableRow["Value"];

    Assert.IsTrue(WebBrowser.ContainsText(value),
      string.Format(
        "Could not find text {0} on the page", value));
  }
}
```

If you re-run the SpecFlow tests, you get a bit further, but the Feature still doesn't pass. This time, the test will fail because you don't have a Create.aspx view page. If you add it along with the proper fields as directed by the feature, you'll move another step closer to a completed feature.

The outside-in process for implementing this Create functionality looks something like **Figure 9**.

Those same steps will repeat themselves often in this process, and your speed in iterating over them will increase greatly over time, especially as you implement helper steps (clicking links and buttons, filling in forms and so on) in the AcceptanceTests project and get down to testing the key functionality in each scenario.

From a valid Create View, the Feature will now fill out the appropriate form fields and will attempt to submit the form. You can guess by now what happens next: The test will fail because you don't yet have the logic needed to save the customer record.

Following the same process as before, create the test using the unit-test code shown earlier in **Figure 3**. After adding an empty Create method that accepts a customer object to allow this test to compile, you watch it fail, then complete the Create method like so:

```
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Create(Customer customer) {
  _repository.Create(customer);

  return View("Details", customer);
}
```

My Controller is just a controller, and the actual creation of the customer record belongs to a Repository object that has knowledge of a data storage mechanism. I've left that implementation out of this article for brevity, but it's important to note that, in a real scenario, the need for a repository to save a customer should kick off another

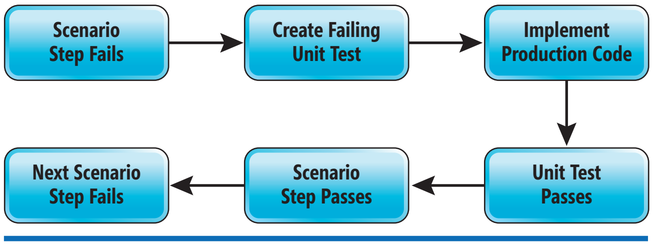


Figure 9 **Scenario-to-Unit Test Process**

sub-loop of unit testing. When you need access to any collaborating object and that object does not yet exist, or doesn't offer the functionality you require, you should follow the same unit test loop that you're following for your Feature and Controllers.

Once you've implemented the Create method and have a working repository, you'll need to create the Details View, which takes the new customer record and displays it on the page. Then you can run SpecFlow once more. Finally, after many TDD loops and sub-loops, you now have a passing feature that proves out some end-to-end functionality in your system.

Congratulations! You've now implemented a unit of end-to-end functionality with an acceptance test and a complete set of unit tests that will ensure the new functionality will continue to work as your system expands to add new features.

A Word About Refactoring

Hopefully, as you create unit-level tests in your UnitTests project, you're constantly refactoring with each test creation. As you move back up the chain from passing unit tests to a passing acceptance test, you should follow the same process, watching for opportunities to refactor and refine your implementation for each feature and all the features that come after.

Be on the lookout for opportunities to refactor the code in your AcceptanceTests project as well. You'll find that some steps tend to be repeated often across several features, especially your Given steps. With SpecFlow, you can easily move these steps into separate Step Definition files organized by function, such as LogInSteps.cs. This leaves your main Step Definition files clean and targeted at the unique scenario you're specifying.

BDD is about focus in design and development. By elevating your focus from an object to a feature, you enable yourself and your team to design from the perspective of the user of a system. As feature design becomes unit design, be sure to author tests with your feature in mind, and ensure the tests are guided by discrete steps or tasks.

Like any other practice or discipline, BDD takes time to fit into your workflow. I encourage you to try it out for yourself, using any of the available tools, and see how it works over time. As you develop in this style, pay attention to the questions that BDD encourages you to ask. Constantly pause and look for ways to improve your practice and process, and collaborate with others on ideas for improvement. My hope is that, no matter your toolset, the study of BDD adds value and focus to your own software development practice. ■

BRANDON SATROM *works as a developer evangelist for Microsoft in Austin, Texas. He blogs at userinexperience.com and can be found on Twitter as @BrandonSatrom.*

THANKS to the following technical experts for refviewing this article: *Paul Rayner and Clark Sell*

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

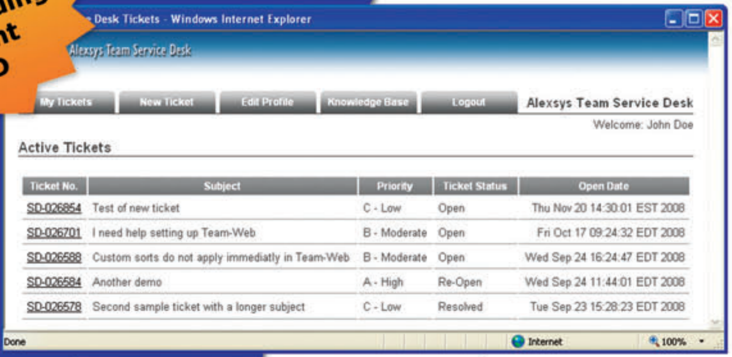
Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

Alexsys Team

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD



Alexsys Team Service Desk

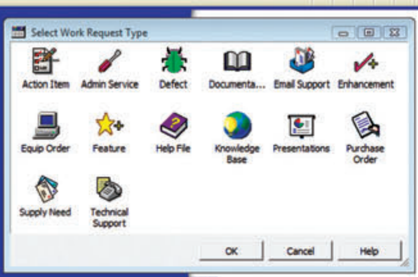
Welcome: John Doe

Active Tickets

Ticket No.	Subject	Priority	Ticket Status	Open Date
SD-026854	Test of new ticket	C - Low	Open	Thu Nov 20 14:30:01 EST 2008
SD-026701	I need help setting up Team-Web	B - Moderate	Open	Fri Oct 17 09:24:32 EDT 2008
SD-026588	Custom sorts do not apply immediatly in Team-Web	B - Moderate	Open	Wed Sep 24 16:24:47 EDT 2008
SD-026584	Another demo	A - High	Re-Open	Wed Sep 24 11:44:01 EDT 2008
SD-026578	Second sample ticket with a longer subject	C - Low	Resolved	Tue Sep 23 15:28:23 EDT 2008

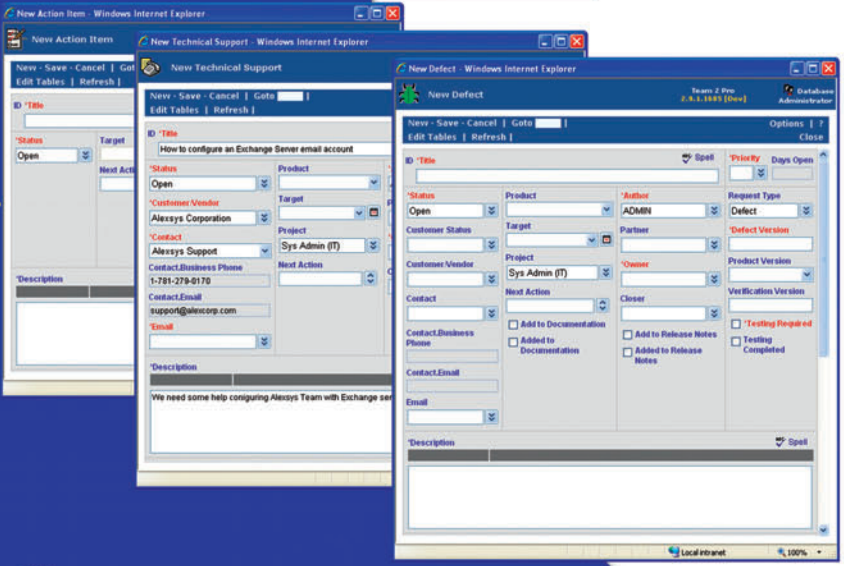
New in Team 2.11

- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment



Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder



New Action Item — *Status: Open

New Technical Support — *Title: How to configure an Exchange Server email account; *Status: Open; *Customer/Vendor: Alexsys Corporation; *Contact: Alexsys Support; Contact.Business Phone: 1-781-279-0170; Contact.Email: support@alexcorp.com; Project: Sys Admin (IT); *Description: We need some help coniguring Alexsys Team with Exchange se

New Defect — *Status: Open; *Author: ADMIN; Request Type: Defect; Project: Sys Admin (IT)

Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration

ZDNet Editors' Pick

Free Trial and Single User FreePack™ available at www.alexcorp.com.
FreePack™ includes a free single user Team Pro and Team-Web license.
Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.

Diagnosing Performance Issues in .NET Applications Using Event Tracing for Windows

Subramanian Ramaswamy

You write a managed application and take it for a spin—and it's slow. Your application is functionally correct, but its performance is much to be desired. You'd like to diagnose the performance issues and resolve them, but your application is running in a production environment, so you can't install profilers or disrupt it. Or your application may not be used widely enough to justify buying Visual Studio Profiler for CPU profiling.

Happily, Event Tracing for Windows (ETW) can mitigate these issues. This powerful logging technology is built into many parts of the Windows infrastructure and is leveraged in the Microsoft .NET Framework 4 CLR to make it simpler than ever to profile your managed application. ETW collects system-wide data and profiles all resources (CPU, disk, network and memory), making it extremely useful for obtaining a holistic view. Moreover, the ETW ecosystem can be tuned so it's low overhead, making it suitable for production diagnostics.

The goal of this article is to give you an idea of the power of using ETW to profile your managed application. I won't cover everything—there are several OS events and CLR ETW events available for diagnostics that we won't look at. But you will get insights into how the performance and functionality of your managed app can be improved dramatically using the ETW ecosystem. To get you started with ETW-based diagnostics for your managed code, I'll showcase a sample investigation using the freely available ETW tool, PerfMonitor, downloadable from bcl.codeplex.com/releases/view/49601.

This article discusses:

- Event Tracing for Windows
- Diagnosing performance issues
- Using PerfMonitor to profile managed code

Technologies discussed:

Event Tracing for Windows, Microsoft .NET Framework 4 CLR

Code download available at:

code.msdn.microsoft.com/mag201012ETW

PerfMonitor

PerfMonitor lets you quickly and easily collect ETW performance data and generate useful reports. It's not intended to be a replacement for deep analysis tools, such as the Visual Studio Profiler; rather, it provides you with an overview of an application's performance characteristics and lets you perform some quick analyses.

There's another tool for ETW diagnostics called XPerf, which is freely available through the Windows Performance Toolkit. However, while XPerf is great for native code profiling on Windows, it does not yet have deep support for managed code profiling. PerfMonitor, on the other hand, exposes the scope and power of profiling managed code using ETW. PerfMonitor has the ability to gather symbolic information associated with .NET Framework runtime code, making it valuable for .NET Framework performance investigations, although it doesn't support the in-depth analysis that XPerf can provide.

PerfMonitor is a fully self-contained tool, and is all you need to start profiling and diagnosing your managed application. The only other requirement is that you must be running at least Windows Vista or Windows Server 2008. PerfMonitor is a command-line tool, and typing PerfMonitor.exe usersGuide from its location will bring up an overview. If you have a customer whose program you want to diagnose under operating conditions—for instance, on a production server—all you need to do is copy the file over to that machine and you're ready to start collecting profiles. The profiles can be analyzed offline if needed.

Four factors are generally examined during any performance investigation: CPU, disk I/O, memory and scalability. Most investigations start with the CPU, which affects both the startup and execution time of your application. Examining disk I/O is most helpful when diagnosing lengthy startup times (disk I/O is a major factor in cold startup times, which is the time it takes for an application to start when it's not present in memory, such as after a reboot), whereas excessive memory consumption (or leaks) can cause your app to grow slower over time. Scalability matters if you want your application to achieve throughput proportional to the number of processors.

PerfMonitor helps you get a quick snapshot of all these except scalability, and it also provides you with enough information to dig deeper using other specialized tools. For example, for diagnosing issues with the CLR .NET garbage collection (GC) heap, the CLRProfiler is a better bet. However, PerfMonitor quickly informs you whether there's an issue and you need to dig deeper using other tools. In some cases, PerfMonitor itself points out the problem and contains all the information you need to tackle a performance bug, as you'll soon see. Take a look at the CLR Inside Out column, "Memory Usage Auditing for .NET Applications" (msdn.microsoft.com/magazine/dd882521), which discusses the importance of auditing your program for memory usage and planning for performance. Extending that philosophy, PerfMonitor quickly lets you audit many performance aspects of your managed program, not just memory.

A Sample Investigation: CsvToXml

The sample program I'll diagnose using ETW converts a CSV file into an XML file. The source code, along with the solution package (and a sample input CSV file, data.csv), is available at code.msdn.microsoft.com/mag201012ETW. To execute the program, run the command CsvToXml.exe data.csv output.xml.

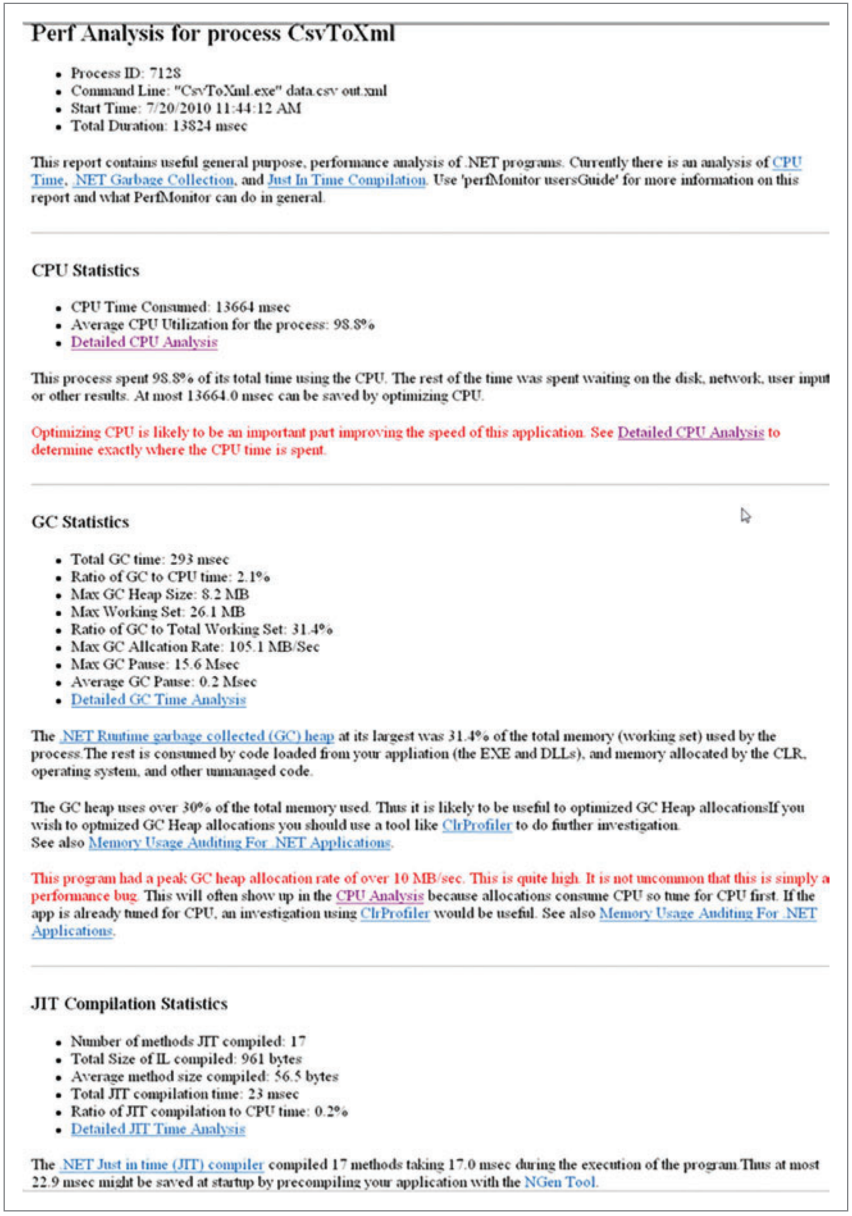
Like many programs, CsvToXml was quickly stitched together and the developer never anticipated that it would be used for large CSV files. When I began using it in the real world, I found it was too slow. It took more than 15 seconds to process a 750K file! I knew there was a problem, but without a profiling tool, I'd really just be guessing as to the cause of the slowness. (Can you spot it just by looking at the source?) Luckily, PerfMonitor can help you figure it out.

Generating and Viewing the Program Trace

The first step is to do a quick audit of the application by executing the following command in an administrator command-prompt window (ETW collects data machine-wide, and hence needs admin privileges):

```
PerfMonitor runAnalyze CsvToXml.exe data.csv out.xml
```

This will start ETW logging, launch CsvToXml.exe, wait for CsvToXml to complete, stop logging and, finally, bring up a Web



Perf Analysis for process CsvToXml

- Process ID: 7128
- Command Line: "CsvToXml.exe" data.csv out.xml
- Start Time: 7/20/2010 11:44:12 AM
- Total Duration: 13824 msec

This report contains useful general purpose, performance analysis of .NET programs. Currently there is an analysis of CPU Time, .NET Garbage Collection, and Just In Time Compilation. Use 'perfMonitor usersGuide' for more information on this report and what PerfMonitor can do in general.

CPU Statistics

- CPU Time Consumed: 13664 msec
- Average CPU Utilization for the process: 98.8%
- Detailed CPU Analysis

This process spent 98.8% of its total time using the CPU. The rest of the time was spent waiting on the disk, network, user input or other results. At most 13664.0 msec can be saved by optimizing CPU.

Optimizing CPU is likely to be an important part improving the speed of this application. See Detailed CPU Analysis to determine exactly where the CPU time is spent.

GC Statistics

- Total GC time: 293 msec
- Ratio of GC to CPU time: 2.1%
- Max GC Heap Size: 8.2 MB
- Max Working Set: 26.1 MB
- Ratio of GC to Total Working Set: 31.4%
- Max GC Allcation Rate: 105.1 MB/Sec
- Max GC Pause: 15.6 Msec
- Average GC Pause: 0.2 Msec
- Detailed GC Time Analysis

The .NET Runtime garbage collected (GC) heap at its largest was 31.4% of the total memory (working set) used by the process.The rest is consumed by code loaded from your appliation (the EXE and DLLs), and memory allocated by the CLR, operating system, and other unmanaged code.

The GC heap uses over 30% of the total memory used. Thus it is likely to be useful to optimized GC Heap allocationsIf you wish to optmized GC Heap allocations you should use a tool like ClrProfiler to do further investigation.
See also Memory Usage Auditing For .NET Applications.

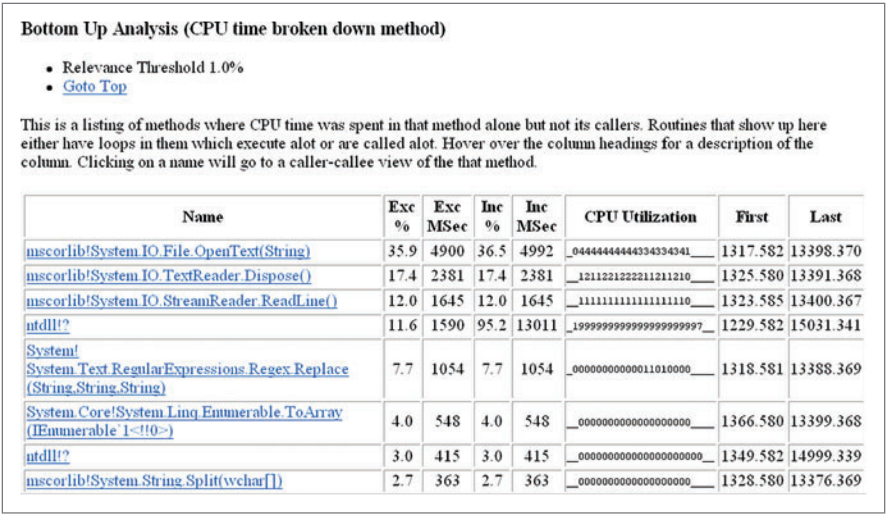
This program had a peak GC heap allocation rate of over 10 MB/sec. This is quite high. It is not uncommon that this is simply a performance bug. This will often show up in the CPU Analysis because allocations consume CPU so tune for CPU first. If the app is already tuned for CPU, an investigation using ClrProfiler would be useful. See also Memory Usage Auditing For .NET Applications.

JIT Compilation Statistics

- Number of methods JIT compiled: 17
- Total Size of IL compiled: 961 bytes
- Average method size compiled: 56.5 bytes
- Total JIT compilation time: 23 msec
- Ratio of JIT compilation to CPU time: 0.2%
- Detailed JIT Time Analysis

The .NET Just in time (JIT) compiler compiled 17 methods taking 17.0 msec during the execution of the program.Thus at most 22.9 msec might be saved at startup by precompiling your application with the NGen Tool.

Figure 1 **Performance Analysis for CsvToXml**


Bottom Up Analysis (CPU time broken down method)

- Relevance Threshold 1.0%
- Goto Top

This is a listing of methods where CPU time was spent in that method alone but not its callers. Routines that show up here either have loops in them which execute alot or are called alot. Hover over the column headings for a description of the column. Clicking on a name will go to a caller-callee view of the that method.

Name	Exc %	Exc MSec	Inc %	Inc MSec	CPU Utilization	First	Last
mscorlib!System.IO.File.OpenText(String)	35.9	4900	36.5	4992	_04444444444433433434341___	1317.582	13398.370
mscorlib!System.IO.TextReader.Dispose()	17.4	2381	17.4	2381	_121122122222211211210___	1325.580	13391.368
mscorlib!System.IO.StreamReader.ReadLine()	12.0	1645	12.0	1645	_1111111111111111110___	1323.585	13400.367
ntdll!?	11.6	1590	95.2	13011	_1999999999999999999997_	1229.582	15031.341
System! System.Text.RegularExpressions.Regex.Replace (String,String,String)	7.7	1054	7.7	1054	_0000000000000011010000___	1318.581	13388.369
System.Core!System.Linq.Enumerable.ToArray (IEnumerable`1<!!0>)	4.0	548	4.0	548	_0000000000000000000___	1366.580	13399.368
ntdll!?	3.0	415	3.0	415	_0000000000000000000000_	1349.582	14999.339
mscorlib!System.String.Split(wchar[])	2.7	363	2.7	363	_0000000000000000000___	1328.580	13376.369

Figure 2 **Bottom-Up Analysis of CsvToXml.exe**

page showing the analysis for CsvToXml. In one easy step, you have a wealth of data to help you uncover the performance bottlenecks in CsvToXml.

The result of this command is captured in **Figure 1**. The page contains, among other data, the process ID, the command line used, and a breakdown of high-level performance data including CPU statistics, GC statistics and just-in-time (JIT) statistics. PerfMonitor also provides a first-level analysis on where to begin the diagnostics, with useful links to informational articles or other tools.

The report shows that the format conversion took nearly 14 seconds, of which 13.6 seconds were in the CPU with an average utilization of 99 percent. Thus, the scenario was CPU-bound.

The total time in GC and the GC Pause times are low, which is good; however, the max GC allocation rate is 105.1MB/sec, which is excessive—this merits further investigation.

CPU Analysis

The Detailed CPU Analysis provides a breakdown of CPU time, as shown in **Figure 2**, and there are three ways of reading CPU profile data. The bottom-up view quickly tells you which methods are consuming the most CPU time and should be diagnosed first. The top-down view is useful for finding out if your code needs architectural or structural changes and helps you understand the overall performance of your program. The caller-callee view indicates the relationship among methods—for example, which methods call which.

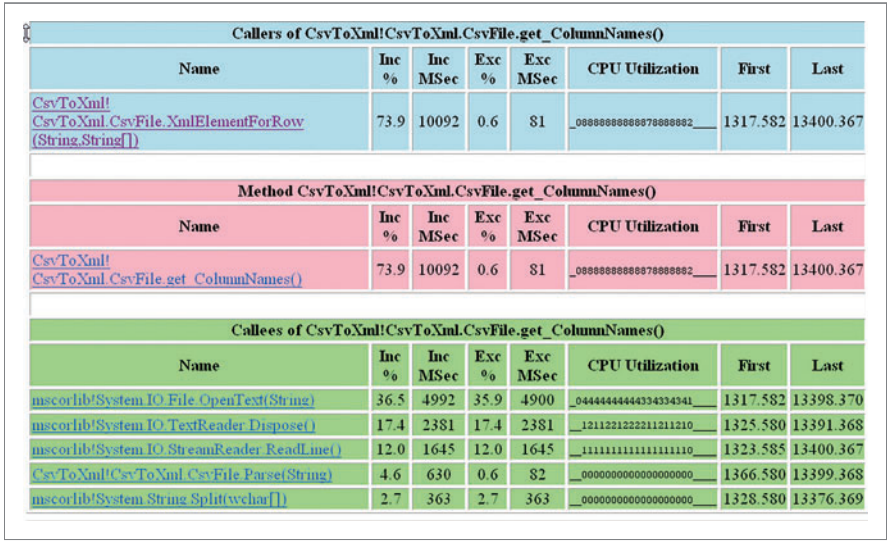
Like other CPU profilers, PerfMonitor views give you the inclusive time (the time spent in a particular method, including time spent in its callees) and the exclusive time (the time spent in a particular method excluding callees). When inclusive and exclusive times are equal, the work is done within that particular method. PerfMonitor also provides a CPU Utilization graph that breaks down CPU usage over time for a particular method. Hovering over the column headings in the report gives you more details on what they mean.

Most performance investigations start with the bottom-up view, which is a listing of methods by exclusive time (this view is shown in **Figure 2**). Upon selecting the bottom-up view, you can see that the mscorlib method System.IO.File.OpenText is the one using the most CPU. Clicking that link brings up the caller-callee view for the OpenText method, which reveals that the CsvToXml.CsvFile.get_ColumnNames method is invoking OpenText from the program—and get_ColumnNames is consuming almost 10 seconds of CPU time (**Figure 3**). Furthermore, this method is called from CsvToXml.CsvFile.XmlElementForRow within a loop (XmlElementForRow itself is called from the Main method).

Thus, something seems to be amiss in these methods. Pulling up the code of these methods leads you to the problem, highlighted in **Figure 4**: the file is opened and parsed repeatedly inside a loop!

Similar scenarios happen more frequently than you'd think. When the method was originally written, the developer may have believed it was going to be invoked only rarely (as was the case with ColumnNames), and thus may not have paid too much attention to its performance. However, situations often come along later that end up calling the method in a loop, and the performance of the application suffers.

In a CSV file, all rows have the same format, so there's no point in doing this every time. You can hoist the ColumnNames functionality into the constructor, as in **Figure 5**, leaving the property to provide the cached column names. This ensures that the file is read only once.


Callers of CsvToXml!CsvToXml.CsvFile.get_ColumnNames()

Name	Inc %	Inc MSec	Exc %	Exc MSec	CPU Utilization	First	Last
CsvToXml! CsvToXml.CsvFile.XmlElementForRow (String,String[])	73.9	10092	0.6	81	_0888888888887888882___	1317.582	13400.367

Method CsvToXml!CsvToXml.CsvFile.get_ColumnNames()

Name	Inc %	Inc MSec	Exc %	Exc MSec	CPU Utilization	First	Last
CsvToXml! CsvToXml.CsvFile.get_ColumnNames()	73.9	10092	0.6	81	_0888888888887888882___	1317.582	13400.367

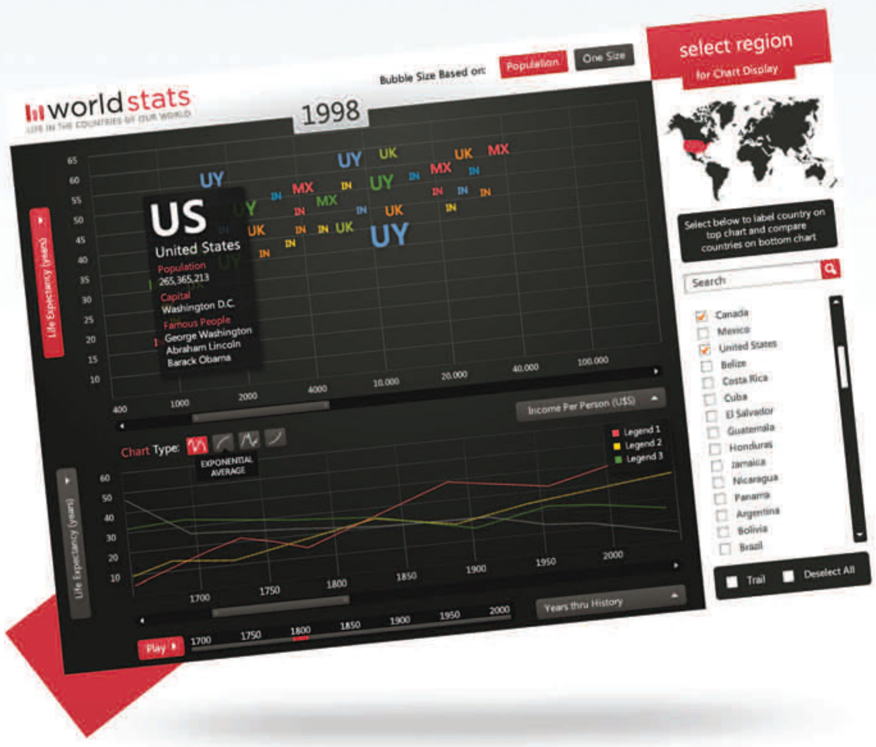
Callees of CsvToXml!CsvToXml.CsvFile.get_ColumnNames()

Name	Inc %	Inc MSec	Exc %	Exc MSec	CPU Utilization	First	Last
mscorlib!System.IO.File.OpenText(String)	36.5	4992	35.9	4900	_04444444444433433434341___	1317.582	13398.370
mscorlib!System.IO.TextReader.Dispose()	17.4	2381	17.4	2381	_121122122222211211210___	1325.580	13391.368
mscorlib!System.IO.StreamReader.ReadLine()	12.0	1645	12.0	1645	_1111111111111111110___	1323.585	13400.367
CsvToXml!CsvToXml.CsvFile.Parse(String)	4.6	630	0.6	82	_0000000000000000000___	1366.580	13399.368
mscorlib!System.String.Split(wchar[])	2.7	363	2.7	363	_0000000000000000000___	1328.580	13376.369

Figure 3 **Caller-Callee View for get_ColumnNames**

EXPERIENCE

Beautiful Data Visualizations That Bring Your Data to Life



Use our Motion Framework™ to see your data over time and give your users new insight into their data. Visit infragistics.com to try it today!

NetAdvantage® ULTIMATE
for ASP.NET, Windows Forms, WPF, Silverlight, WPF Data Visualization, Silverlight Data Visualization

Infragistics®

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91 80 4151 8042
@infragistics

Figure 4 **Method ColumnNames Is Invoked by Method XmlElementForRow**

```
public string[] ColumnNames
{
  get
  {
    using (var reader = File.OpenText(Filename))
      return Parse(reader.ReadLine());
  }
}

public string XmlElementForRow(string elementName, string[] row)
{
  string ret = "<" + elementName;
  for (int i = 0; i < row.Length; i++)
    ret += " " + ToValidXmlName(ColumnNames[i]) + "=\"" +
EscapeXml(row[i]) + "\"";
  ret += "/>";
  return ret;
}
```

After rebuilding, we execute the previous command again and find the application much snappier; the duration is now only 2.5 seconds.

Nonetheless, reviewing the data with the fix, you'll notice that CPU time is still dominant. By again drilling into CPU time and looking at the bottom-up analysis, you can see that Regex.Replace is now the most-expensive method, and that it's called from EscapeXml and ToValidXmlName. Because EscapeXml is the more-expensive method (330 ms exclusive time), check its source code:

```
private static string EscapeXml(string str)
{
  str = Regex.Replace(str, "\"", "&quote;");
  str = Regex.Replace(str, "<", "&lt;");
  str = Regex.Replace(str, ">", "&gt;");
  str = Regex.Replace(str, "&", "&amp;");
  return str;
}
```

EscapeXml is also called within a loop in XmlElementForRow, and thus has the potential to be a bottleneck. Regular expressions are a bit of overkill for these replacements, and using a string Replace method would be more efficient. So replace EscapeXml with the following:

```
private static string EscapeXml(string str)
{
  str = str.Replace("\"", "&quote;");
  str = str.Replace("<", "&lt;");
  str = str.Replace(">", "&gt;");
  str = str.Replace("&", "&amp;");
  return str;
}
```

With this transformation, you've reduced the overall time to approximately two seconds, with CPU time still dominant. This is acceptable performance—you've improved the execution speed almost sevenfold.

As an exercise for the reader, I've left a few more performance bugs in the sample program that can be identified using ETW events.

Figure 5 **Caching the Column Names for Better Performance**

```
public CsvFile(string csvFileName)
{
  Filename = csvFileName;

    using (var reader = File.OpenText(Filename))
      ColumnNames = Parse(reader.ReadLine());

}

public string Filename { get; private set; }

public string[] ColumnNames { get; private set;}
```

Exploring GC Statistics

PerfMonitor GC statistics supply a quick overview of the memory profile. As you may recall, I strongly recommend memory usage auditing, and the information provided through GC ETW events provides a quick snapshot of any problems with the .NET GC heap. The quick summary view tells you the aggregate GC heap size, the allocation rates and the GC pause times. Selecting the GC Time Analysis link on the PerfMonitor results tab shows you the details of the GCs, when they occurred, how much time they consumed and so forth.

The information lets you determine if you need to dig further into any memory issues using the CLRProfiler or other memory profilers. The article "Profiling the .NET Garbage-Collected Heap" (msdn.microsoft.com/magazine/ee309515) digs into debugging the .NET GC heap using the CLRProfiler.

For this particular program, none of the GC statistics looks worrisome. The allocation rate is high; a good rule of thumb is to have your allocation rate below 10MB/s. However, pause times are very small. High allocation rates show up under CPU time, which mostly means that there are CPU gains to be had—as you found out. However, after the fixes, the allocation rates remain high, and this means there are a lot of allocations happening (can you fix this?). The GC pause-time of a few milliseconds is a testament to the self-tuning and efficient GC that the .NET Framework runtime provides. Thus, the .NET Framework GC is automatically taking care of memory management.

Exploring JIT Statistics

To improve startup time, one of the first items to look into is the time needed for JIT compiling of methods. If the time taken is significant (for example, most of the time used during startup of your app is consumed by JIT compilation), the application might benefit from native image generation (NGen), which eliminates JIT compilation time by precompiling the assembly and saving it on disk. That is, the assembly is JIT-compiled and saved on disk, eliminating the need for JIT compilation for subsequent executions. Before going down the NGen route, though, you may also want to consider deferring some of the methods being JIT compiled to a later point in the program so the JIT compilation times don't affect startup. For more information, please see the article, "The Performance Benefits of NGen" (msdn.microsoft.com/magazine/cc163610).

The sample application CsvToXml.exe didn't have a significant startup cost, and allowing it to JIT compile all methods every time is fine. The JIT compilation statistics also tell you that the number of methods that were JIT compiled was 17 (suggesting that all the methods called were JIT compiled), and the total JIT compilation time was 23 ms. Neither of these is a performance issue with this application, but for larger applications where JIT compilation time is a factor, using NGen should eliminate any problems. Normally, JIT compilation time becomes a factor when an application starts JIT compiling hundreds or thousands of methods. In such cases, NGen is the solution to eliminate JIT compilation costs.

More guidance on improving startup is available in other *MSDN Magazine* articles, and the ETW events can help identify and fix bottlenecks. Several other JIT events are available as well, including

Figure 6 **ETW Events in the .NET Framework 4**

Event Category Name	Description
Runtime Information ETW Event	Captures information about the runtime, including the SKU, version number, the manner in which the runtime was activated, the command-line parameters it was started with, the GUID (if applicable) and other relevant information.
Exception Thrown ETW Event	Captures information about exceptions that are thrown.
Contention ETW Events	Captures information about contention for monitor locks or native locks that the runtime uses.
Thread Pool ETW Events	Captures information about worker thread pools and I/O thread pools.
Loader ETW Events	Captures information about loading and unloading application domains, assemblies and modules.
Method ETW Events	Captures information about CLR methods for symbol resolution.
GC ETW Events	Captures information pertaining to GC.
JIT Tracing ETW Events	Captures information about JIT inlining and tail calls.
Interop ETW Events	Captures information about Microsoft intermediate language (MSIL) stub generation and caching.
Application Domain Resource Monitoring (ARM) ETW Events	Captures detailed diagnostic information about the state of an application domain.
Security ETW Events	Captures information about strong name and Authenticode verification.
Stack ETW Event	Captures information that's used with other events to generate stack traces after an event is raised.

JIT inline events that can provide insights into why a method wasn't inlined.

CLR ETW Events in the .NET Framework 4

The CLR team wrote a blog post on tracking down DLL loads and determining whether a particular DLL needs to be loaded during startup. The process of determining whether a DLL load needs to happen during startup becomes simpler with ETW events. By using the ETW Module Load events available in the .NET Framework 4, we know which modules are loaded and why. There are also events for module unloads and so on.

There are several other events in the .NET Framework 4 that make diagnosing your managed application simpler than ever. **Figure 6** summarizes these events. All the events that were triggered during execution can be dumped with the PerfMonitor runPrint command. The CLR team has also run down events that allow you to attach and detach ETW profiling, and the team plans to keep adding more ETW events to make the process of debugging managed applications simpler in future releases.

You'll find two files with the suffix PerfMonitorOutput in the execution directory; these are the ETW log files. You'll also see files with the suffix kernel, signifying they contain the OS events. The data collected by PerfMonitor is the same data that XPerf uses, so you can use PerfMonitor to simplify data collection and simple reporting and XPerf for more advanced analysis of the same data. The PerfMonitor merge command converts the ETW files to an XPerf-readable format.

Wrapping Up

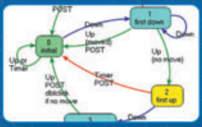
Performance investigation using ETW is simple, yet powerful. Various free, low-overhead ETW-based tools are available that allow debugging of managed code efficiently. I've just skimmed the surface of the ETW events that are available in the .NET Framework runtime. My aim was to get you started debugging your managed application using ETW events and tooling. Downloading PerfMonitor and using the MSDN documentation of ETW events in CLR, along with the CLR Perf Blog, will jump-start your performance investigations of your managed applications.

A special thanks to Vance Morrison, partner architect for CLR Performance, for his guidance and assistance with this article. ■

SUBRAMANIAN RAMASWAMY *is the program manager for CLR Performance at Microsoft. He holds a Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology.*

GoDiagram

Add powerful diagramming capabilities to your applications in less time than you ever imagined with GoDiagram components.



The first and still the best. We were the first to create diagram controls for .NET and we continue to lead the industry.

Fully customizable interactive diagram components save countless hours of programming enabling you to build applications in a fraction of the time.

Our new WPF and Silverlight products fully support XAML, including data-binding, templates, and styling.

For .NET WinForms, ASP.NET, WPF and Silverlight

Specializing in diagramming products for programmers for 15 years!

Powerful, flexible, and easy to use.

Find out for yourself with our **FREE** Trial Download with full support at: ***www.godiagram.com***

Re-Introducing the Windows Azure AppFabric Access Control Service

Vittorio Bertocci and Wade Wegner

If you're looking for a service that makes it easier to authenticate and authorize users within your Web sites and services, you should take another look at the Windows Azure AppFabric Access Control service (ACS for short), as some significant updates are in the works (at the time of this writing).

Opening up your application to be accessed by users belonging to different organizations—while maintaining high security standards—has always been a challenge. That problem has traditionally been associated with business and enterprise scenarios, where users typically live in directories. The rise of the social Web as an important arena for online activities makes it increasingly attractive to make your application accessible to users from the likes of Windows Live ID, Facebook, Yahoo and Google.

This article is based on a prerelease version of the Windows Azure AppFabric Access Control service. All information is subject to change.

This article discusses:

- Outsourcing authentication of a Web site to the ACS
- Configuring an ACS project
- Choosing identity providers
- Adding rules
- Testing authentication flow
- The structures and features of the ACS

Technologies discussed:

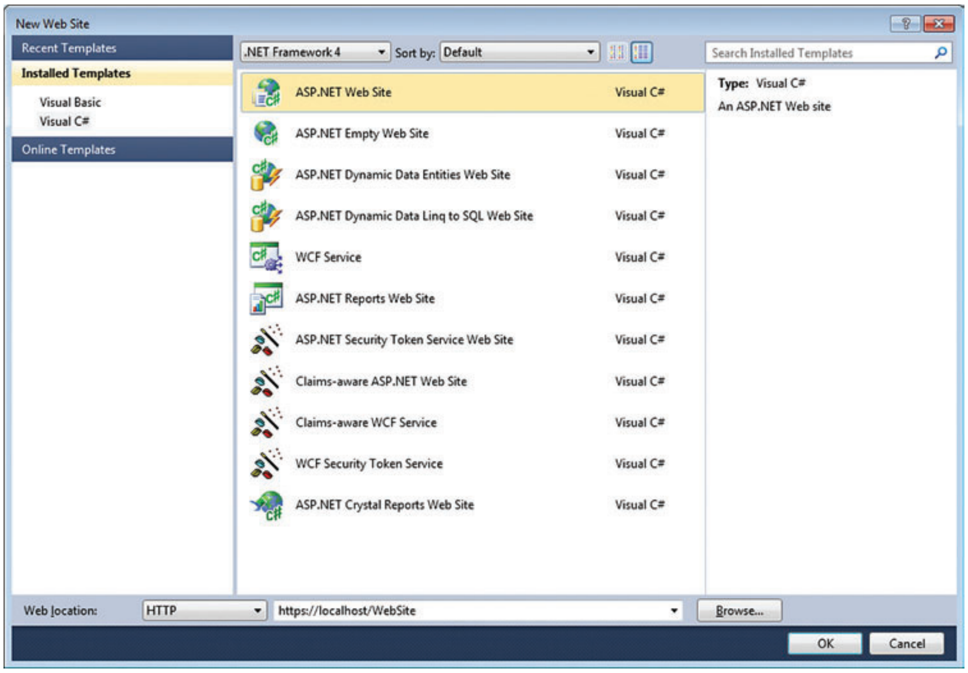
Windows Azure

With the emergence of open standards, the situation is improving; however, as of today, implementing these standards directly in your applications while juggling the authentication protocols used by all those different entities is a big challenge. Perhaps the worst thing about implementing these things yourself is that you're never done: Protocols evolve, new standards emerge and you're often forced to go back and upgrade complicated, cryptography-ridden authentication code.

The ACS greatly simplifies these challenges. In a nutshell, the ACS can act as an intermediary between your application and the user repositories (*identity providers*, or *IP*) that store the accounts you want to work with. The ACS will take care of the low-level details of engaging each IP with its appropriate protocol, protecting your application code from having to take into account the details of every transaction type. The ACS supports numerous protocols such as OpenID, OAuth WRAP, OAuth 2.0, WS-Trust and WS-Federation. This allows you to take advantage of many IPs.

Outsourcing authentication (and some of the authorization) from your solution to the ACS is easy. All you have to do is leverage Windows Identity Foundation (WIF)—the extension to the Microsoft .NET Framework that enhances applications with advanced identity and access capabilities—and walk through a short Visual Studio wizard. You can usually do this without having to see a single line of code!

Does this sound Greek to you? Don't worry, you're not alone; as it often happens with identity and security, it's harder to explain something than actually do it. Let's pick one common usage of the ACS, outsourcing authentication of your Web site to multiple Web IPs, and walk through the steps it entails.



New Web Site

Recent Templates | Installed Templates | Visual Basic | Visual C# | Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

ASP.NET Web Site	Visual C#
ASP.NET Empty Web Site	Visual C#
ASP.NET Dynamic Data Entities Web Site	Visual C#
ASP.NET Dynamic Data Linq to SQL Web Site	Visual C#
WCF Service	Visual C#
ASP.NET Reports Web Site	Visual C#
ASP.NET Security Token Service Web Site	Visual C#
Claims-aware ASP.NET Web Site	Visual C#
Claims-aware WCF Service	Visual C#
WCF Security Token Service	Visual C#
ASP.NET Crystal Reports Web Site	Visual C#

Type: Visual C#
An ASP.NET Web site

Web location: HTTP | https://localhost/WebSite | Browse...

OK | Cancel

Figure 1 **Selecting an ASP.NET Web Site with HTTP as the Location**

Outsourcing Authentication of a Web Site to the ACS

Let's start by taking a vanilla Web site and enabling users to log in using a Google account.

Before we get started, let's make sure we have the prerequisites covered. Here's what we need:

- Visual Studio 2010
- Windows Identity Foundation Runtime
- Windows Identity Foundation SDK and one of the following: Windows 7, Windows Server 2008, Windows Server 2008 R2 or Windows Vista SP1

Although it's not a hard requirement, having IIS on the machine will help; if you don't have IIS installed, you'll have to adjust the steps of the walkthrough here and there.

While Visual Studio requires no introduction, it will probably help to expand a bit on WIF (pronounced "dub-IF"), and why it's a prerequisite. (For a thorough explanation of WIF, see "Programming Windows Identity Foundation" [Microsoft Press, 2010]).

WIF is an extension to the .NET Framework that provides you with a new model for dealing with authentication and user identity, a model that decouples your application from the details of how authentication takes place. Traditional authentication systems (such as the ASP.NET Membership provider APIs) force you to cope with the details of how authentication takes place. This requires you to use low-level APIs to deal with low-level constructs such as passwords, certificates and the like. WIF changes all this by offering a handy abstraction that allows you to specify an external entity to handle user authentication. With Forms-based authentication, you specify a given page—typically login.aspx—where requests are redirected whenever the caller is not yet authenticated. With WIF, you can enlist an external entity—an IP—to be invoked whenever a user needs authentication. The ways in which the IP is chosen at design time and engaged at run time follow well-known protocols. WIF takes care of discovering which protocols should be used and enforcing communication policies accordingly. Once again, this is much easier to show than to explain.

Create the Initial Solution

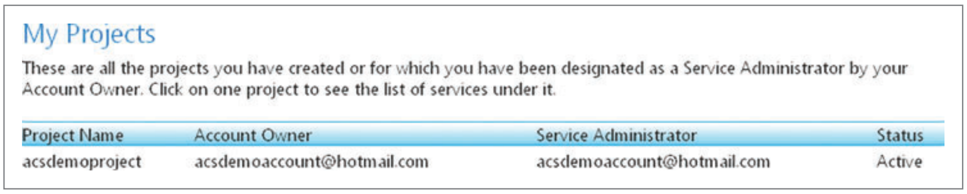
Open up Visual Studio. Create a new Web site by selecting File | New | Web Site. Let's create a new ASP.NET Web site—but first, be sure that you've selected the Web location as "HTTP" and configured the URL so that it's running in IIS (see **Figure 1**).This will ensure a smooth run when using the WIF tools. If you have HTTPS available on your Web server, it's a good idea to use it; although not strictly necessary for this walkthrough, it's highly recommended on production systems and will save you some warnings from WIF.

When you hit F5, you'll see that you have a basic ASP.NET Web site, and by clicking the "Log In" link you'll get prompted to enter a username and password. This is what we're going to change—instead of using a username and password and handling authentication directly in the Web site, we're going to use WIF to outsource authentication to the ACS. The ACS will in turn allow us to open up access to external IPs.

Configure an ACS Project

To begin, we need to create a project in the Windows Azure AppFabric LABS portal. The LABS portal is an environment set up specifically for allowing the community to access early bits. There's no cost associated with AppFabric LABS, but there are also no service-level agreements or guarantees.

Open your browser and go to portal.appfabriclabs.com. You'll be prompted to log in with a Windows Live ID. Once logged in, you'll need to create a new project—click the "create a project" link. You'll have to choose a project name—select something appropriate and click OK. Once complete, you'll see an active project name ("acsdemoproject" in our example)—click it (see **Figure 2**).

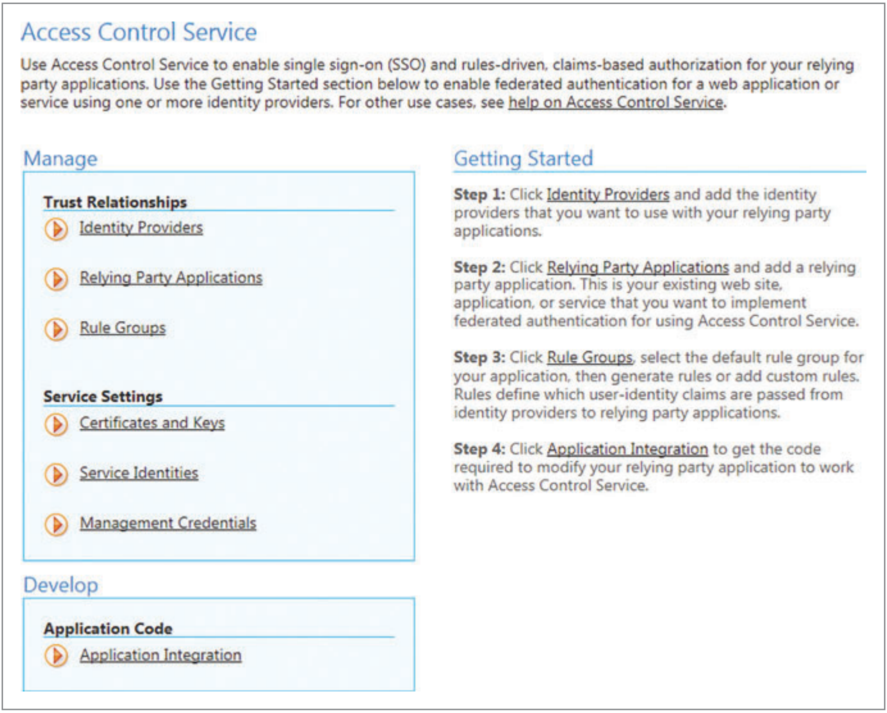


My Projects

These are all the projects you have created or for which you have been designated as a Service Administrator by your Account Owner. Click on one project to see the list of services under it.

Project Name	Account Owner	Service Administrator	Status
acsdemoproject	acsdemoaccount@hotmail.com	acsdemoaccount@hotmail.com	Active

Figure 2 **Creating a Project in the Windows Azure AppFabric LABS Portal**



Access Control Service

Use Access Control Service to enable single sign-on (SSO) and rules-driven, claims-based authorization for your relying party applications. Use the Getting Started section below to enable federated authentication for a web application or service using one or more identity providers. For other use cases, see help on Access Control Service.

Manage

Trust Relationships
- Identity Providers
- Relying Party Applications
- Rule Groups

Service Settings
- Certificates and Keys
- Service Identities
- Management Credentials

Develop

Application Code
- Application Integration

Getting Started

Step 1: Click Identity Providers and add the identity providers that you want to use with your relying party applications.

Step 2: Click Relying Party Applications and add a relying party application. This is your existing web site, application, or service that you want to implement federated authentication for using Access Control Service.

Step 3: Click Rule Groups, select the default rule group for your application, then generate rules or add custom rules. Rules define which user-identity claims are passed from identity providers to relying party applications.

Step 4: Click Application Integration to get the code required to modify your relying party application to work with Access Control Service.

Figure 3 **The Windows Azure AppFabric Access Control Service Management Portal**

Before you can outsource authentication to the ACS, you need to define a service namespace. Think of the service namespace as providing you with your own slice of the AppFabric LABS environment and—for the ACS—the unique component for all the URIs of the resources you'll use when interacting with the ACS from your application. Everything contained within the service namespace is yours to control. Click "Add Service Namespace," specify a name, choose a zone—in LABS you can only select "United States (South/Central)"—and click Create. Note that the URIs used by AppFabric are available on the public Internet and are meant to uniquely identify services; hence you must choose a namespace that hasn't been picked by anybody else.

It'll take a few moments, but after your service namespace activates, you'll be able to click the "Access Control" link to start configuring the ACS for your Web site.

Now you've made it to the management portal, where you can configure the ACS for your Web sites (see **Figure 3**).

Click the "Manage" button to get started. The management portal provides some guided steps to walk you through the process of getting started, and that's just what we're going to do.

Choosing the Identity Providers You Want

Click the "Identity Providers" link. Here we want to configure the various social IPs we want to leverage from within your application. Windows Live ID is present in the list by default; let's add support for Google accounts.

Click the "Add Identity Provider" link, which will show a list of providers. Click the "Add" button next to Google. You can specify a custom image URL for the IP, but go ahead and just click "Save." Just like that, we've added Google as a recognized source of user identities.

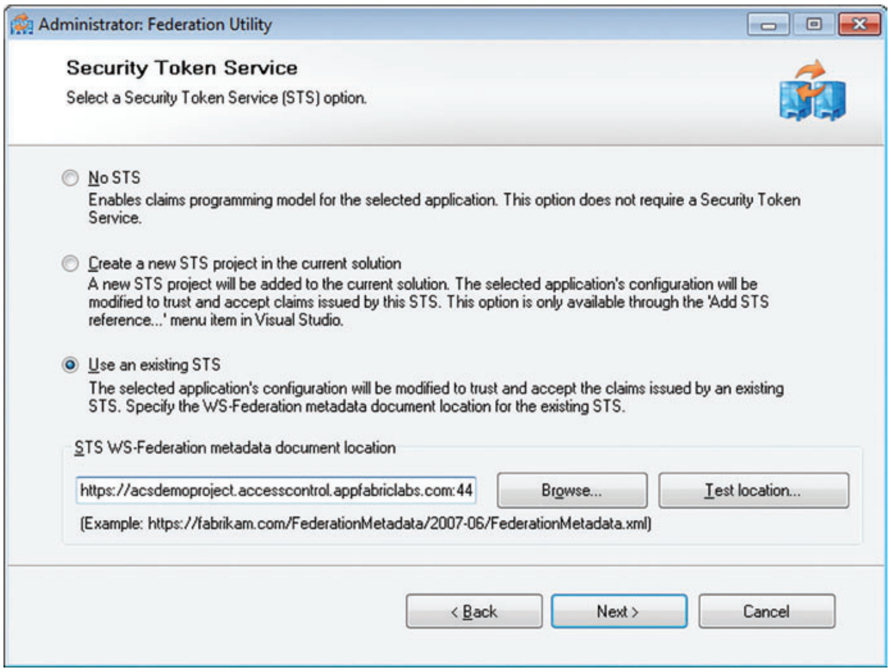
Getting the ACS to Recognize Your Web Site

Now that our IPs have been configured, we need to provide information to the ACS about our Web site. In identity jargon, we often refer to applications as "Relying Parties," an expression that refers to the fact that the application relies on one or more IPs to take care of authentication on their behalf. The ACS UI is consistent with this terminology.

Click the "Relying Party Applications" URL, and then "Add Relying Party Application." Let's specify the following information:

- Name: My Website
- Realm: https://localhost/Website/
- Return URL: https://localhost/Website/
- Token format: SAML 2.0
- Token signing: Use service namespace certificate (typical)

The Token Format field deserves at least a short explanation (we'll spend more time on the topic later in the article). A token is an artifact—typically an XML fragment or something in another serialization format—used by IPs to indicate that a successful authentication operation took



Administrator: Federation Utility

Security Token Service

Select a Security Token Service (STS) option.

( ) No STS
Enables claims programming model for the selected application. This option does not require a Security Token Service.

( ) Create a new STS project in the current solution
A new STS project will be added to the current solution. The selected application's configuration will be modified to trust and accept claims issued by this STS. This option is only available through the 'Add STS reference...' menu item in Visual Studio.

[✓ Use an existing STS]
The selected application's configuration will be modified to trust and accept the claims issued by an existing STS. Specify the WS-Federation metadata document location for the existing STS.

STS WS-Federation metadata document location: https://acsdemoproject.accesscontrol.appfabriclabs.com:44

(Example: https://fabrikam.com/FederationMetadata/2007-06/FederationMetadata.xml)

Figure 4 **Starting the Federation Utility Wizard in Visual Studio**

place. Once a user authenticates, using whatever system the IP chooses, the user's browser will be redirected to the ACS carrying a token that certifies the user has been recognized. The token format and protocol used will be determined according to the IP. The ACS will examine the token and, if it finds it satisfactory (more about this later), will emit a new token of its own and send it back to your application. The settings you change in this step determine which token format you wish the ACS to use for communicating back to your application. The ACS is capable of emitting three types of tokens—SAML2.0, SAML1.1 and SWT—representing different trade-offs between expressive power, security, applicability for certain client types and so on. Just pick SAML2.0 here; the details aren't imperative at this point.

It's important that the realm corresponds to the URL for the Web site we created earlier. Once the authentication with the IP of choice takes place, the ACS will redirect the user back to your Web site using the URL you specify here. Note that, by default, "Create New Rule Group" is selected—we'll leverage this in the next step. Click "Save" once you're done and return to the management portal.

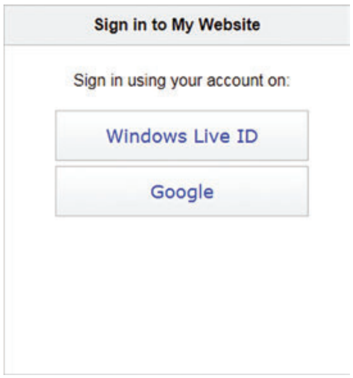


Figure 5 **The Home Realm Discovery Page**

The ACS can act as an intermediary between your application and the user repositories that store the accounts you want to work with.

Adding Rules

Rules are interesting constructs that give you fine-grained control over user identity information. The scenario we're enabling right now, however, doesn't require the explicit use of rules in order to enable sign-on from multiple IPs. Therefore, we'll postpone all explanations about what rules are to a later section in the article, where they'll actually come in handy; here we'll just go with the default settings.

Click the "Rule Groups" link. You should see the rule group created when we added the relying party application ("Default Rule Group for My Website"). Select this rule group, click the "Generate Rules" link, confirm that both Google and Windows Live ID are selected, and then click the "Generate" button—that's all you need to do in regard to rules in this scenario.

Collecting the WS-Federation Metadata Address

At this point, we're finished configuring the ACS. However, before we jump back to Visual Studio, let's grab some information from the Application Integration page. Click the "Application Integration" link and copy the "WS-Federation Metadata" URL—we're going to use this with WIF to set up our Web site to leverage.

Without going into too much detail, the WS-Federation Metadata document is a machine-readable description of how the ACS handles authentication. Your application will need it in order to be configured to outsource authentication to the ACS.

Configuring the Web Site to Use the ACS

Return to Visual Studio and your Web site. We now want to leverage WIF to outsource authentication to the ACS, which will in turn enable Google accounts to access our application. In the Solution Explorer, right-click the Web site project and select "Add STS Reference." This will launch the Federation Utility wizard, which will configure the Web site to use WIF as the authentication mechanism and the ACS as the authenticating authority. STS stands for "Security Token Service," which indicates a special kind of Web service or Web page that offers an entry point for requesting tokens; usually every IP or token issuer uses one.

You can just click "next" most of the time; the steps in which you'll have to enter information are precious few. Advance to the "Security Token Service" step, and specify "Use an existing STS." Paste the federation metadata URL you copied from the ACS portal (see **Figure 4**).

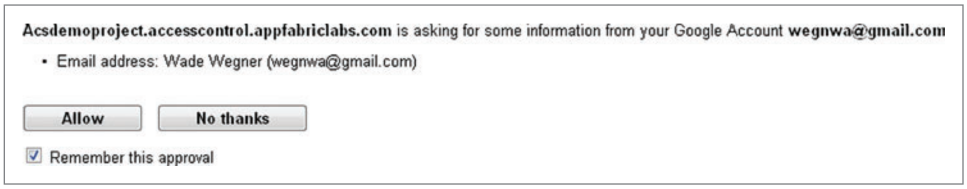
From there, leave the defaults, click through to the end and select Finish. The wizard will add all the required WIF assemblies, add some files to your Web site and (most importantly) update your web.config with the information required to engage with the ACS at authentication time.

Testing the Authentication Flow

It's finally time to give your newly secured Web site a spin! Hit F5. You'll immediately be redirected to the Home Realm Discovery page, which offers the user the chance to pick among the IPs we configured earlier in the ACS management portal (see **Figure 5**).

After you select Google and enter your Google account credentials, you'll see an approval page that requires you to allow the ACS project access—this is important to understand, as it's not your Web site that's requesting permission, but instead the ACS (see **Figure 6**).

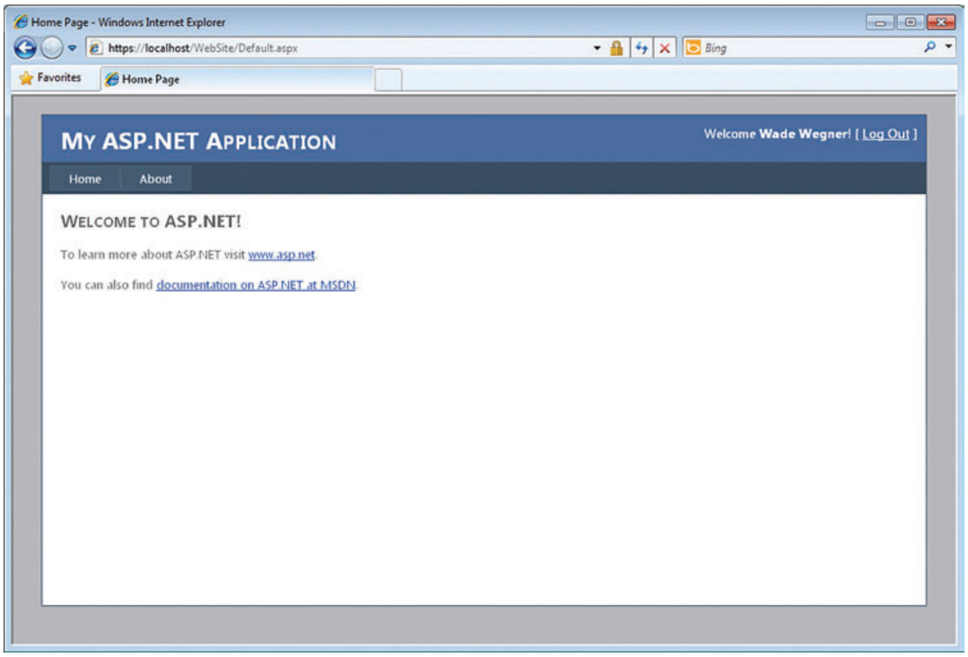
Once you've allowed the ACS the access it requires, you'll get redirected back to the Web site (see **Figure 7**). That's it—you're logged in!



Acsdemoproject.accesscontrol.appfabriclabs.com is asking for some information from your Google Account **wegnwa@gmail.com**

- Email address: Wade Wegner (wegnwa@gmail.com)

Figure 6 **The Windows Azure AppFabric Access Control Service Asking for Permission to Get Information from Google**


Home Page - Windows Internet Explorer
https://localhost/WebSite/Default.aspx
Bing
Favorites
Home Page
My ASP.NET Application
Welcome Wade Wegner! [Log Out]
Home
About

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

Figure 7 **Success! Logging in to the Web Site**

If you want to verify that the same experience would work with Windows Live ID, the other IP configured in your namespace, all you need to do is close the browser, hit F5 again and at the Home Realm Discovery page pick Windows Live ID instead of Google.

If you have any experience in enabling authentication protocols on Web sites, you know that, traditionally, adding an IP means studying its protocols and API, writing fairly challenging code and testing, testing, testing before getting it right. And every additional IP requires the same, plus the extra complication of understanding from the request which protocol is being used.

Here, we didn't need to do any of that; in fact, you may have noticed that we didn't write a single line of code. If we want to add extra identity providers, all we'll need to do is go through a couple of screens on the ACS management portal, with no impact whatsoever on the application itself. If the IPs evolve their protocols, the ACS will change its code to accommodate the new conditions, and our application won't even know anything changed at all.

The ACS: Structure and Features

Now that you've had a chance to experience firsthand the power of the ACS, you're ready for a brief overview of what the ACS really is and what makes it tick. This will require a bit of theory, but you'll discover that you already learned most of what you need to know while walking through the scenario described earlier.

The ACS operates according to the principles of *claims-based identity*. The main idea behind claims-based identity is that every entity in an identity transaction plays one or more canonical roles, taken from a short list of four: subject, identity provider (IP), relying party (RP) and federation provider (FP). In the walkthrough, you've seen all those in action. The interaction among those entities boils down to requesting, obtaining and forwarding security tokens, as shown in **Figure 8**.

The subject is the role played by the user—that is, the entity that needs to be authenticated. The IP is the entity that stores the account for the subject: username, credentials, attributes and so on. The IP uses one or more STSes for exposing its authentication capabilities and for issuing tokens. The RP is the application that the subject wants to use. Those three roles are enough for describing the most basic case: the subject obtains a token from an IP that the RP trusts, uses that token with the RP and the authentication is done.

One thing we didn't cover during the walkthrough is that the tokens aren't just representing the successful outcome of the authentication operation, but they're also used to transport attributes describing the user: name, e-mail address roles and anything else that the RP needs to know and that the IP makes available. If you recall the properties of signed security tokens, you'll see how those attributes can't be tampered with and are cryptographically guaranteed to come from a specific IP. That means that one RP can choose to consider valid the attributes it receives according to how much it trusts the IP that originated them. Think of real-life situations in which you need to prove something—for example, that you actually live at a certain address. Companies often ask you to provide a utility bill, mainly because they trust the utility company more than they trust you. The information is the same (the address), but the IP that produced it makes all the difference.

When an attribute is issued as part of a security token, we call that attribute a *claim*. This concept is so important that it gives the name to the entire approach, and practically everything the ACS

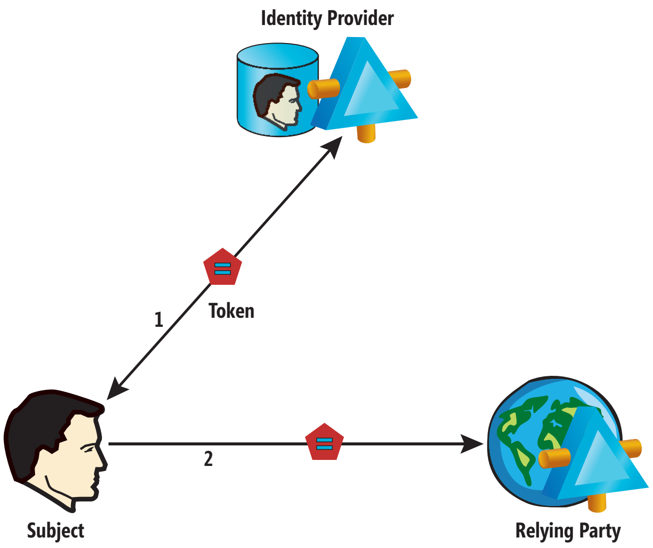


Figure 8 **Requesting, Obtaining and Forwarding Security Tokens**

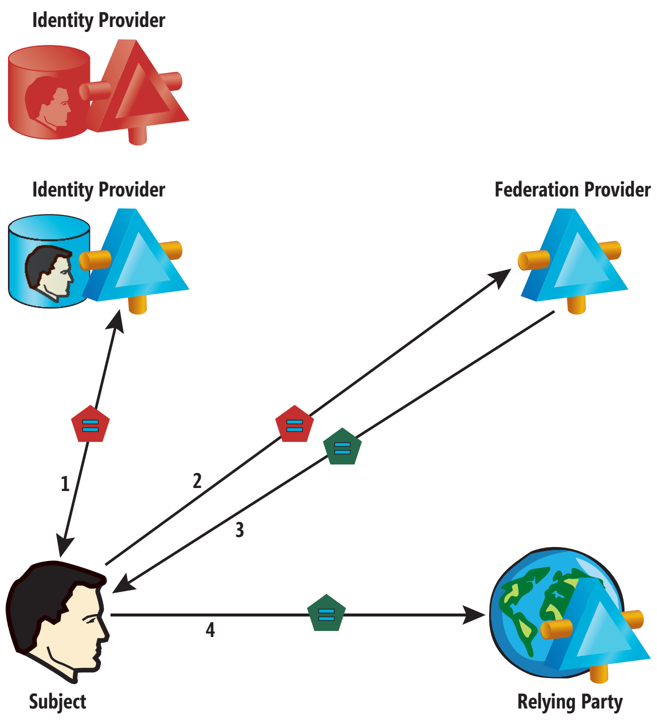


Figure 9 **The Federation Provider as an Intermediary**

does revolves around claims. We just need to get another concept out of the way and then we'll go in the details.

Although you could use the subject-RP-IP roles for modeling every system, in practice it's not very handy. If one RP trusts multiple IPs, as was the case in our scenario, the model would require the RP to maintain multiple relationships, handle different protocols and so on. This is where the fourth role, the FP, comes into play. An FP is an *intermediary* between one or more RPs and one or more IPs, as shown in **Figure 9**.

The FP trusts multiple IPs, behaving like an application and expecting tokens from the IPs. In turn, the RP trusts the FP; to that purpose the FP exposes its own STS, which issues tokens for the RP. The FP takes care of the details of engaging with the various IPs, while always presenting to the RP the same façade, so IPs can be on-boarded and de-provisioned without affecting the RP. The FP can also transform the claims coming from different IPs to make them more useful for the RP. It can normalize different incoming claim types, add extra claims such as roles or permissions, and so on.

As you may have guessed by now, the ACS plays the role of the FP, as illustrated in **Figure 10**.

When you create a service namespace, you get your very own full-featured FP in the cloud. Out of the box, that FP includes four different STS endpoints, all offering different protocols that are suitable for different application types: WS-Federation for signing in to Web sites; WS-Trust for invoking SOAP Web services; OAuth WRAP and OAuth 2 for REST Web services; and Web APIs in general. Those are the endpoints you use to configure your application to outsource authentication.

The ACS is already pre-configured to trust various Web IPs, as we've seen, and it facilitates the experience of choosing among them by providing pages or embeddable code for them. In addition to that, the ACS is able to establish trust with commercial IPs such as Active Directory Federation Services 2.0 (AD FS 2.0), which expose STS endpoints themselves. In practice, the ACS exposes the counterpart of the "Add STS reference" functionality you've seen when configuring your Web site to trust the ACS. Using AD FS 2.0 as an IP is extremely interesting, as it allows you to reuse user accounts whenever you want, including those in Windows Azure-hosted applications that would traditionally be valid only on-premises. Another interesting feature of business IPs is that they usually provide much richer claims sets that can be used for adding sophisticated identity-driven logic in the token processing.

The ACS allows you to describe your claims transformation login in the form of rules, a simple but powerful mechanism. For example, you can assign a role to a user as simply as entering something along the lines of "if the incoming name identifier claim has value X, please add an output claim of type role and value Y."

All of the functionality discussed here can be accessed through the management portal you used in the walkthrough; alternatively, there's an OData-based management service that gives you full control on the ACS settings while integrating with your existing processes.

As trite as it may sound, we did barely scratch the surface of what the ACS can do for you. We invite you to check out the hands-on lab in the identity developer training kit and the Windows Azure platform training kit for exploring more scenarios in greater detail. If you want to simplify access management for your Web site, Web service or Web API, the ACS can help! ■

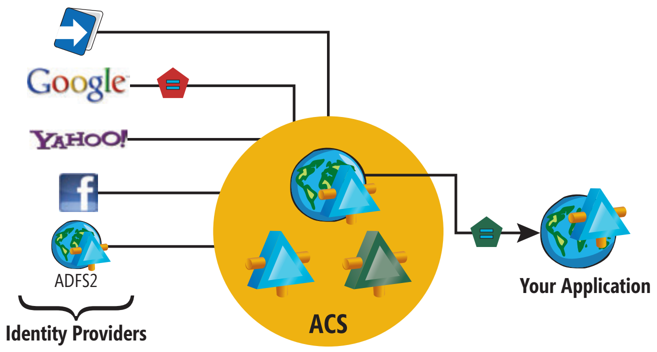


Figure 10 **The Windows Azure AppFabric Access Control Service Playing the Role of Federation Provider**

VITTORIO BERTOCCI *is a senior architect evangelist in the Developer and Platform Evangelism team and a member of the extended engineering team that produces Microsoft claims-based platform components. He's responsible for identity evangelism for the .NET developer community and drove initiatives such as the Identity Developer Training Kit and the IdElement show on Channel 9. He recently wrote "Programming Windows Identity Foundation" (Microsoft Press, 2010).*

WADE WEGNER *is a senior technical evangelist at Microsoft, responsible for influencing and driving Microsoft's technical strategy for the Windows Azure platform. You can reach him through his blog at wadewegner.com or on Twitter at twitter.com/WadeWegner.*

THANKS to the following technical expert for reviewing this article: *Kent Brown*

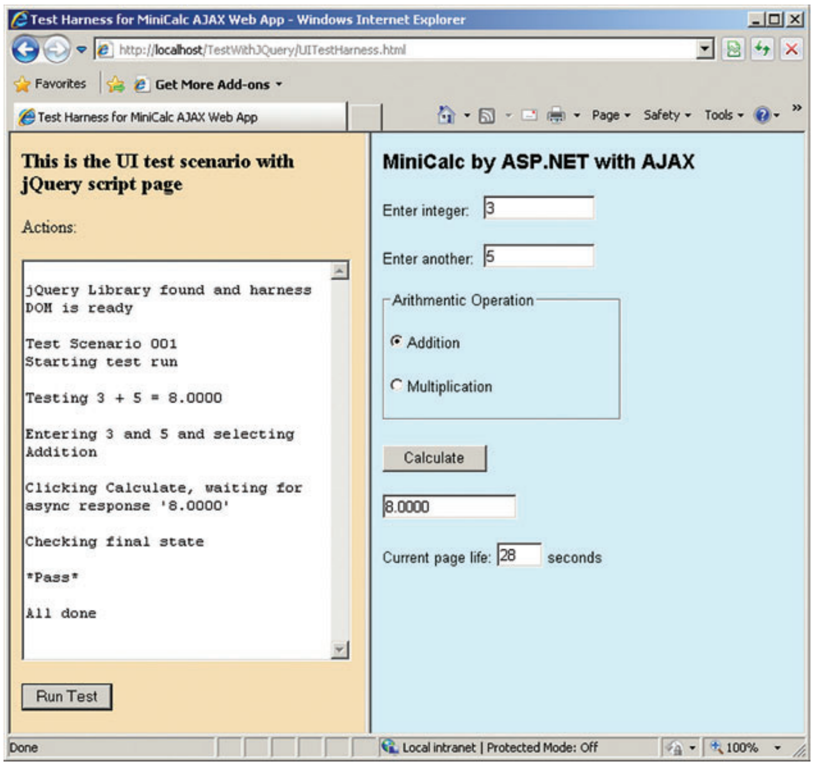
JAMES MCCAFFREY

Web Application UI Testing with jQuery

The jQuery library is an open source collection of JavaScript functions. Although jQuery was created with Web development in mind, the library has several characteristics that make it well-suited for lightweight Web application UI test automation. In this month's column I'll show you how to do just that.

The best way for you to see where I'm headed is to examine the screenshot in **Figure 1**, which shows UI test automation with jQuery in action. The test harness is hosted by Internet Explorer and consists of an HTML page named UITestHarness.html.

The harness page is really just a container with two HTML frame elements. The frame on the right holds the Web application under test, in this case a simple but representative ASP.NET calculator application named MiniCalc. The frame on the left holds an HTML page named TestScenario001.html, which consists of a TextArea element to display progress messages, a Button element to manually launch the automation, and jQuery-based JavaScript functions that manipulate the Web application under test and check the resulting state of the application to determine a pass/fail result.


Test Harness for MiniCalc AJAX Web App - Windows Internet Explorer
http://localhost/TestWithJQuery/UITestHarness.html
Favorites
Get More Add-ons
Test Harness for MiniCalc AJAX Web App
Page
Safety
Tools
Run Test
Calculate
Done
Local intranet | Protected Mode: Off
100%

This is the UI test scenario with jQuery script page

Actions:

```
jQuery Library found and harness
DOM is ready

Test Scenario 001
Starting test run

Testing 3 + 5 = 8.0000

Entering 3 and 5 and selecting
Addition

Clicking Calculate, waiting for
async response '8.0000'

Checking final state

*Pass*

All done
```

MiniCalc by ASP.NET with AJAX

Enter integer: 3

Enter another: 5

Arithmentic Operation

(● Addition) ○ Multiplication

8.0000

Current page life: 28 seconds

Figure 1 **UI Test Automation with jQuery**

The jQuery library is also well-suited for HTTP request-response testing, and I addressed request-response testing with jQuery in the January 2010 Test Run column (msdn.microsoft.com/magazine/ee335793).

This article assumes you have basic familiarity with ASP.NET technology and intermediate JavaScript programming skills, but does not assume you have any experience with the jQuery library. However, even if you're new to ASP.NET and test automation in general, you should still be able to follow this month's column without too much difficulty.

In the sections that follow, I'll first describe the MiniCalc application so you'll know exactly how the implementation of the application under test is related to the UI test automation. Next, I'll walk you through the details of creating lightweight jQuery-based UI test automation as shown in **Figure 1**. I'll wrap up by describing how you can extend the techniques I've presented to meet your own needs, and I'll discuss the advantages and disadvantages of jQuery UI test automation compared to alternative approaches. I think the techniques presented here are interesting and can be a useful addition to your testing, development and management toolsets.

The Application Under Test

Let's take a look at the code for the MiniCalc ASP.NET Web application, which is the target of my jQuery-based UI test automation.

I created the MiniCalc application using Visual Studio 2008. After launching Visual Studio I clicked File | New | Web Site. To avoid the ASP.NET code-behind mechanism and keep all the code for my Web application in a single file, I selected the Empty Web Site option. Next, I selected the HTTP mode option (rather than File mode) from the Location field drop-down and specified the location as:

```
http://localhost/TestWithJQuery/MiniCalc
```

I decided to use C# for the MiniCalc app logic. The test automation techniques presented here will work with ASP.NET Web applications written with C# and Visual Basic as well as Web applications created using technologies such as classic ASP, CGI, PHP, JSP, Ruby and so on.

Code download available at code.msdn.microsoft.com/mag201012TestRun.

I clicked OK on the New Web Site dialog to configure IIS and generate the structure of my Web app. Next, I went to the Solution Explorer window, right-clicked the MiniCalc project name, and selected Add New Item from the context menu. I then selected Web Form from the list of installed templates and accepted the Default.aspx file name. I cleared the "Place code in separate file" option and then clicked the Add button.

Next, I double-clicked on the Default.aspx file name in Solution Explorer to load the template-generated code into the text editor. I deleted all the template code and replaced it with the code shown in **Figure 2**.

To keep the size of my source code small and easy to understand, I omitted normal error checking. The complete source code for the MiniCalc application and the test harness is available from code.msdn.microsoft.com/mag201012TestRun.

To write test automation for Web applications, in most cases you need to know the IDs of the various user controls. As you can see in **Figure 2**, I used TextBox1 and TextBox2 to hold two user integer input values—RadioButton1 and RadioButton2 to select addition or multiplication—and TextBox3 to hold the arithmetic calculation result.

When a user clicks on the Button1 control, the MiniCalc app first goes into a random delay of one to five seconds to simulate server-side processing of some sort, and then computes and displays either a sum or product of the two user input values.

Next, I decided to make the MiniCalc app asynchronous by using AJAX technology. To do that I needed a web.config file for the application, so, rather than create a web.config file manually from scratch, I hit the F5 key to instruct Visual Studio to build and run the app through the debugger. When Visual Studio prompted me for permission to add a web.config file, I clicked OK. Next, I added a ScriptManager server-side control to the MiniCalc application to enable AJAX:

```
<asp:ScriptManager ID="sm1" runat="server" EnablePartialRendering="true" />
```

Then I added the tags necessary to asynchronously update the TextBox3 result element in conjunction with the Button1 click event:

```
<asp:UpdatePanel ID="up1" runat="server">
<ContentTemplate>
<p><asp:TextBox id="TextBox3" width="120"  runat="server" />
</ContentTemplate>
<Triggers>
<asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
</Triggers>
</asp:UpdatePanel>
```

If you examine **Figure 1** closely, you can see that, to emphasize the fact that MiniCalc is an AJAX app, I placed a client-side page life counter in the UI. When an asynchronous request to MiniCalc returns, only TextBox3 is updated and the page life counter is not reset. The pageLife text box is defined as:

```
<input type="text" id="pageLife" size="1"/>
```

The associated client-side JavaScript is:

```
<script language="javascript">
  var count = 0;
  function updatePageLife() {
    ++count;
    var tb = document.getElementById("pageLife");
    tb.value = parseInt(count);
    window.setTimeout(updatePageLife, 1000);
  }
</script>
```

The counter is started up by the application onload event:

```
<body bgColor="#ccffff" onload="updatePageLife();">
```

Figure 2 **MiniCalc Web Application Under Test Source**

```
<%@ Page Language="C#" %>
<script runat="server">
  static Random rand = null;

  private void Page_Load(object sender, EventArgs e)
  {
    if (!IsPostBack)
      rand = new Random(0);
  }

  private void Button1_Click(object sender, System.EventArgs e)
  {
    int randDelay = rand.Next(1, 6); // [1-5]
    System.Threading.Thread.Sleep(randDelay * 1000);
    int x = int.Parse(TextBox1.Text);
    int y = int.Parse(TextBox2.Text);
    if (RadioButton1.Checked)
      TextBox3.Text = (x + y).ToString("F4");
    else if (RadioButton2.Checked)
      TextBox3.Text = (x * y).ToString("F4");
  }
</script>
<html>
  (client-side JavaScript and UI elements here)
</html>
```

Recall that the window.setTimeout function calls its function argument only once after the specified delay in milliseconds, so timing algorithms typically use recursive calls, as in this code.

Web Application UI Testing with jQuery

Now that you've seen the Web application under test, let's dive right into the UI test automation code. The main test harness is simply an ordinary HTML page with two frame elements:

```
<html>
<!-- UITestHarness.html -->
<head>
  <title>Test Harness for MiniCalc AJAX Web App</title>
</head>
  <frameset cols="45%,*" onload="leftFrame.appLoaded=true">
    <frame src="http://localhost/TestWithJQuery/TestScenario001.html"
       name="leftFrame" >
    <frame src="http://localhost/TestWithJQuery/MiniCalc/Default.aspx"
       name="rightFrame">
  </frameset>
</html>
```

The frame named rightFrame hosts the Web application under test as is, without any modifications or test instrumentation. The frame named leftFrame hosts an HTML page named TestScenario001.html, which contains all the jQuery test automation code. Notice that when the frameset element onload event fires, a variable in the leftFrame page, named appLoaded, is set to true. This variable will be used to make sure that the test automation does not begin before the Web application under test is completely loaded into the test harness. The structure of the test scenario code is listed in **Figure 3**.

The test script begins by referencing the jQuery library:

```
<script src='http://localhost/TestWithJQuery/jquery-1.3.2.js'>
```

Here I point to a local copy of the jQuery library that I had downloaded from the jQuery Project Web site (jquery.com) and copied to the MiniCalc application root directory. I used jQuery version 1.3.2. The library is under constant development, so there will likely be a newer version available by the time you read this article. For more information about referencing the jQuery library in your code, see "Getting the jQuery Library" on p. 72.

Next, I use a standard jQuery idiom to determine if my automation has access to the jQuery library:

Getting the jQuery Library

You have a few options for the location of the jQuery library used by your application. As mentioned, you can download the latest version from **jquery.com** and use it from your local file system. The jQuery site has both development (uncompressed) and production (minified—white space removed—for a smaller footprint) downloads available. Just select the package you want and save the .js file to your project directory.

If your application host has an active Internet connection, an even easier option is to point to the most current version of jQuery from an online content delivery network (CDN). There are a number of sources you can use (including your own hosted version), but two highly available CDNs are the Microsoft AJAX Content Delivery Network (**asp.net/ajaxlibrary/cdn.ashx**) and the Google Libraries API (**code.google.com/apis/libraries**).

For example, you could use the minified version of jQuery from the Microsoft Ajax CDN with the following script tag:

```
<script
  src="http://ajax.microsoft.com/ajax/jquery/jquery-1.3.2.min.js"
  type="text/javascript">
</script>
```

Scott Guthrie has a useful blog post on using the Microsoft Ajax CDN for both jQuery and ASP.NET AJAX at **tinyurl.com/q7rf4w**.

In general, when using jQuery for test automation, using a local, unpacked copy of the library in the test harness is more reliable than using a remote or packed copy. For production applications, however, you'll want to use a reliable hosted library.

```
$(document).ready(function() {
  logRemark("jQuery Library found and harness DOM is ready\n");
} );
```

The jQuery ready function fires as soon as the containing document DOM is fully loaded into the test host memory and all DOM elements are available. If the jQuery library is not accessible—which typically happens when you specify an incorrect path to the library—an "Object expected" error will be thrown.

The ready function accepts an anonymous function as its single parameter. Anonymous functions are used frequently in test automation for both jQuery and JavaScript. You can think of an anonymous function as a function that's defined on the fly using the function keyword.

Here's an example for a function called logRemark:

```
function logRemark(comment) {
  var currComment = $("#comments").val();
  var newComment = currComment + "\n" + comment;
  $("#comments").val(newComment);
}
```

In this situation I define a function that simply invokes a program-defined logging function called logRemark to display a message to the test harness that jQuery is available. I could also have used the intrinsic JavaScript alert function.

I begin by using the jQuery selector and chaining syntax to get the current text in the textarea with ID "comments." The $ notation is a shortcut alias for the jQuery meta-class. The # syntax is used to select an HTML element by ID, and the val function can act as both a value setter and getter (a property in object-oriented programming terminology). I append the comment parameter and a newline character to the existing comment text, and then use jQuery syntax to update the TextArea element.

Next, I set up a few test automation global variables:

```
var testScenarioID = "Test Scenario 001";
var maxTries = 20;
var numTries;
var polling = 500;
var appLoaded = false;
var started = false;
```

Because my automation deals with an asynchronous application, I don't use arbitrary time delays. Instead, I use a sequence of short (defined by variable polling) delays, checking repeatedly (variable numTries) to see if the value of some HTML element satisfies a Boolean condition, up to a maximum number of attempts (variable maxTries). In this test scenario I delay a maximum of 20 attempts at 500 ms delay per attempt for a total of 10 seconds. The appLoaded variable is used to determine when the Web app under test is fully loaded into the test harness. The started variable is used to coordinate test harness execution.

To manually start the automation you can click on the Run Test button:

```
<input type="button" value="Run Test" onclick="runTest();" />
```

The launch function shown in **Figure 3** is used for full test automation, as I'll explain shortly. The runTest function acts as the main coordinating function for the test automation:

```
function runTest() {
  waitUntilAppLoaded();
  started = true;
  try {
    logRemark(testScenarioID);
    logRemark("Testing 3 + 5 = 8.0000\n");
    step1();
  }
  catch(ex) {
    logRemark("Fatal error: " + ex);
  }
}
```

The runTest function begins by calling the waitUntilAppLoaded function, which is defined as:

```
function waitUntilAppLoaded() {
  if (appLoaded == true) return true;
  else window.setTimeout(waitUntilAppLoaded, 100);
}
```

Anonymous functions are used frequently in test automation for both jQuery and JavaScript.

Recall that the test scenario initializes variable appLoaded to false and that the harness frameset onload event sets appLoaded to true. Here I use the intrinsic setTimeout function to repeatedly pause for 100 ms until the value of appLoaded becomes true. Note that this approach could delay forever. To prevent this possibility, you may want to add a global counter and return false after some maximum number of delays.

After setting the global start variable, runTest displays some comments and invokes a step1 function in an exception handler wrapper. The harness structure I present here is only one possibility, and you can modify the harness organization to suit your programming style and test environment. With my structure I consider a test scenario as a sequence of state changes, each of which is represented by a stepX function.

Figure 3 **Structure of UI Test Automation Page**

```
<html>
<!-- TestScenario001.html -->
<head>
  <script src='http://localhost/TestWithJQuery/jquery-1.3.2.js'></script>
  <script type="text/javascript">
    $(document).ready(function() {
      logRemark("jQuery Library found and harness DOM is ready\n");
    } );

    var testScenarioID = "Test Scenario 001";
    var maxTries = 20;
    var numTries;
    var polling = 500; // milliseconds
    var appLoaded = false;
    var started = false;

    function launch() {
      if (!started)
        runTest();
    }

    function waitUntilAppLoaded() {
      // Code
    }

    function runTest() {
      // Start automation
    }

    function step1() {
      // Manipulate state
    }

    function clickCalculate() {
      // Click the Calculate button
    }

    function checkControl(controlID, controlVal) {
      // Determine if control has specified value
    }

    function step2() {
      // Manipulate state
    }

    function callAndWait(action, checkControlFunc, controlID, controlVal,
      callbackFunc, pollTime) {
      // The heart of the automation
    }

    function doWait(checkControlFunc, controlID, controlVal,
      callbackFunc, pollTime) {
      // Wait until Web app responds
    }

    function finish() {
      // Determine pass/fail result
    }

    function logRemark(comment) {
      // Utility logging function
    }
  </script>

</head>
<body bgcolor="#F5DEB3">
  <h3>This is the UI test scenario with jQuery script page</h3>
  <p>Actions:</p><p><textarea id="comments" rows="22" cols="34">
  </textarea></p>
  <input type="button" value="Run Test" onclick="runTest();" />
</body>
</html>
```

The step1 function manipulates the state of the Web application under test by simulating user input, as shown in **Figure 4**.

The jQuery syntax for accessing and manipulating HTML elements is consistent, elegant and, for the most part, browser-independent. Notice that to access the Web app loaded in the rightFrame element from code in the leftFrame element, I must use the parent keyword. Also notice that I must use the jQuery find filter.

When manipulating the TextBox1 and TextBox2 elements, I make the assumption that the Web app under test is fully loaded into the rightFrame element. This assumption may not be reasonable for applications with long load times, and in such situations you can place the jQuery selector code in a window.setTimeout delay loop, testing the target object against the built-in "undefined" value.

Because the MiniCalc application under test is an AJAX application, my harness cannot simply invoke the Calculate button click event, because the test harness code would continue execution without waiting for the application's asynchronous response. So, I use a program-defined callAndWait function:

```
function callAndWait(action, checkControlFunc, controlID,
  controlVal, callbackFunc, pollTime) {
    numTries = 0;
    action();
    window.setTimeout(function(){doWait(
      checkControlFunc, controlID, controlVal,
      callbackFunc, pollTime);}, pollTime);
}
```

The callAndWait function will invoke a function (the action parameter), go into a delay loop and pause a short amount of time (variable pollTime), and check to see if some application state is true by calling parameter function checkControlFunc with arguments of parameter controlID and controlVal. When checkControlFunc returns true, or a maximum number of delays have been executed, control will be transferred to parameter function callbackFunc.

The callAndWait function works hand-in-hand with a program-defined doWait function:

```
function doWait(checkControlFunc, controlID,
  controlVal, callbackFunc, pollTime) {
  ++numTries;

  if (numTries > maxTries) finish();
  else  if (checkControlFunc(controlID, controlVal))
    callbackFunc();
  else window.setTimeout(function(){
    doWait(checkControlFunc, controlID,
    controlVal, callbackFunc, pollTime);}, pollTime);
}
```

The doWait function is recursive and exits when checkControlFunc returns true or local counter numTries exceeds global variable maxTries. So, this calls a function named clickCalculate, goes into a delay loop, pauses polling for 500 ms and calls the function checkControl with arguments of TextBox3 and 8.0000 until checkControl returns true or the delay loop has executed 20 times (specified by maxTries):

```
callAndWait(clickCalculate, checkControl, "TextBox3",
  "8.0000", step2, polling);
```

If checkControl returns true, control is transferred to function step2. The clickCalulate function uses jQuery selection and chaining:

```
function clickCalculate() {
  var btn1 = $(parent.rightFrame.document).find('#Button1');
  if (btn1 == null || btn1.val() == undefined)
    throw "Did not find btn1";
  btn1.click();
}
```

The main reason for defining an action wrapper function like this is so that the function can be conveniently passed by name to the callAndWait function. The checkControl function is straightforward:

```
function checkControl(controlID, controlVal) {
  var ctrl = $(parent.rightFrame.document).find('#' + controlID);
  if (ctrl == null || ctrl.val() == undefined || ctrl.val() == "")
    return false;
  else
    return (ctrl.val() == controlVal);
}
```

First I use jQuery syntax to get a reference to the control specified by parameter controlID. If the value of the control is not yet available, I immediately return to the delay loop. Once the control value is ready I can check to see whether it's equal to some expected value given by parameter controlVal.

After calling as many stepX functions as I care to call, I transfer control to a finish function. That function first determines how it was reached:

```
if (numTries > maxTries) {
  logRemark("\nnumTries has exceeded maxTries");
  logRemark("\n*FAIL*");
}
else ....
```

If the value of the global numTries variable exceeds the value of maxTries, then I know that the Web application under test hasn't responded within the time allowed. Here I arbitrarily decide that this is a test case failure rather than some form of an undetermined result. If numTries hasn't exceeded maxTries I begin checking the final state of the app under test:

```
logRemark("\nChecking final state");
var tb1 = $(parent.rightFrame.document).find('#TextBox1');
var tb2 = $(parent.rightFrame.document).find('#TextBox2');
var tb3 = $(parent.rightFrame.document).find('#TextBox3');
```

Here I get references to the three textbox controls. Exactly which elements of the Web app under test you decide to check will depend on the details of your particular application. Next, I examine the value of each textbox control to see if each has an expected value:

```
var result = "pass";
if (tb1.val() != "3") result = "fail";
if (tb2.val() != "5") result = "fail";
if (tb3.val() != "8.0000") result = "fail";
```

My test scenario script has all test case input and expected values hard-coded. The test automation I present is best suited for lightweight, quick testing situations where hard-coded test data is simple and effective.

The finish function wraps up the test run by displaying a pass or fail result:

```
if (result == 'pass')
  logRemark("\n*Pass*");
else
  logRemark("\n*FAIL*");
```

As with the test case input data, this approach is lightweight and you may want to write test results to an external file on the test host or Web server, or perhaps send test results via SMTP to an e-mail address.

Wrapping Up

The harness described here is semi-automated in the sense that you must click on a button control to launch the test. You can fully automate the harness by adding a start-wrapper function:

```
function launch() {
  if (!started)
    runTest();
}
```

Add an attribute of onload="leftFrame.launch();" to the frameset element in the harness page. Each load of the Web application in the harness will trigger an onload event, so I use the global "start" variable to prevent the test automation from restarting. Interestingly, even though the HTML Frame element doesn't support an onload event, you can in fact place an onload attribute in the harness frame element, and the event will bubble up to its parent frameset element.

Now you can create a .bat file with commands such as:

```
iexplore http://localhost/TestWithJQuery/UITestHarness001.html
iexplore http://localhost/TestWithJQuery/UITestHarness002.html
```

When the .bat file executes—perhaps via a Windows Task Scheduler—the harness will load, and your automation will launch automatically. Another way you might want to extend the test system I've presented here is to place the program-defined functions into a jQuery plug-in.

When writing lightweight Web application UI test automation you have several alternatives to the jQuery-based approach I've presented here. One of the primary advantages of using the jQuery library compared to using raw JavaScript is that jQuery works across multiple browsers such as Internet Explorer, Firefox and Safari. Another significant advantage is that by using jQuery to write test automation, you can actively build your knowledge of using jQuery for Web development tasks.

Using jQuery does have disadvantages compared to alternative approaches. The use of jQuery entails an external dependency to some extent, and script-based test automation tends to be more difficult to manage than non-script test automation. Compared to using a test framework such as Selenium or Watir, writing jQuery-based automation gives you more flexibility, but you must write code at a lower level of abstraction.

As usual, I'll remind you that no one particular test automation approach is best suited for all situations, but jQuery-based Web application UI test automation can be an efficient and effective technique in many software development scenarios. ■

Figure 4 **Simulating Input with the step1 Function**

```
function step1() {
  logRemark(
    "Entering 3 and 5 and selecting Addition");
  var tb1 =
    $(parent.rightFrame.document).find('#TextBox1');
  tb1.val('3');

  var tb2 =
    $(parent.rightFrame.document).find('#TextBox2');
  tb2.val('5');

  var rb1 =
    $(parent.rightFrame.document).find('#RadioButton1');
  rb1.attr("checked", true);

  logRemark(
    "\nClicking Calculate, waiting for async response '8.0000'");
  asyncCall(clickCalculate, checkTextBox3, "8.0000",
    step2, polling);
}
```

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's worked on several Microsoft products, including Internet Explorer and MSN Search. Dr. McCaffrey is the author of ".NET Test Automation Recipes" (Apress, 2006), and can be reached at jammc@microsoft.com.*

THANKS to the following technical experts for reviewing this article: *Scott Hanselman and Matthew Osborn*

Speed • Reliability • Precision • Agility

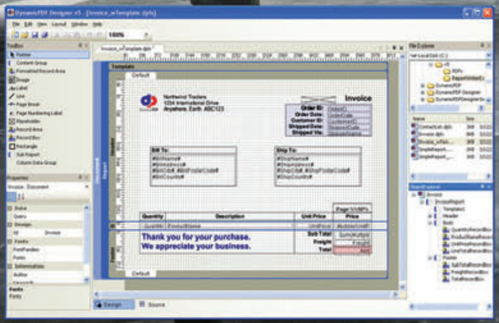
DynamicPDF…Proven .NET Components for Real-Time PDFs

DynamicPDF

• Easy-to-use • Highly efficient
• Industry leading support • Huge feature set

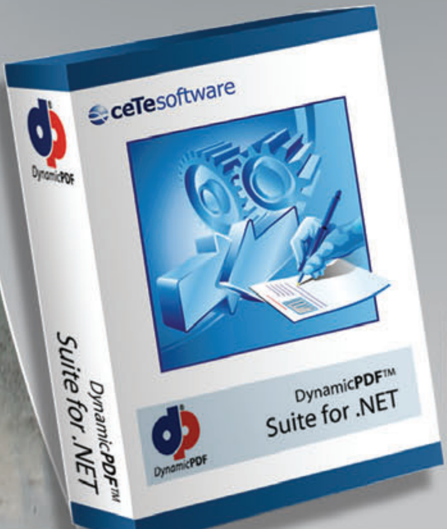
Try it FREE today!

Experience our fully functional, never expiring evaluation and community editions.


Layout reports in DynamicPDF Designer with its Visual Studio look and feel.

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.



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.

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.

ceTesoftware
INFINITE POSSIBILITIES

www.cete.com
info@cete.com

800.631.5006
+1 410.772.8620

Multiparadigmatic .NET, Part 4: Object Orientation

In the previous article, we explored commonalities and variability as expressed through procedural programming, and discovered several interesting "sliders" by which variability can be introduced into designs. In particular, two design approaches emerged out of the procedural line of thought: name-and-behavior variability, and algorithm variability.

As the complexity of programs and their requirements grew, developers found themselves struggling to keep all of the various subsystems straight. Procedural abstractions, we discovered, didn't "scale" as well as we might've hoped. With the advent of the GUI, a new style of programming began to emerge, one that many readers who learned the "plain old SDK" style of building Windows apps from the Windows 3 SDK and Charles Petzold's classic "Programming Windows" (Microsoft Press, 1998) will recognize instantly. Ostensibly procedural in nature, this style followed a particularly interesting pattern. Each procedure in a closely clustered knot of related functionality centered around a "handle" parameter, most often taking it as a first (or only) parameter, or returning it from a Create call or the like: CreateWindow, FindWindow, ShowWindow and more, all centered around a window handle (HWND), for example.

Inheritance gave developers a new axis on which to analyze commonality and variability.

What developers didn't realize at the time was that this was actually a new way of programming, a new paradigm that would make itself felt within just a few years. Hindsight, of course, makes it obvious that this was object-oriented programming, and most readers of this column will be well-versed with its precepts and ideas already. Given that, why would we decide to spend precious column inches on the subject? The answer is that no discussion of multiparadigm design would be complete without incorporating objects within its purview.

Object Fundamentals

Object orientation is, in many ways, an exercise in inheritance. Implementation inheritance has dominated much of the object design discussion, with advocates suggesting that proper abstractions are built by identifying the entities in the system—the "nouns," as it were—and where commonality emerges, elevating that commonality into a base class, thus creating an "IS-A" relationship. A Student IS-A Person, an Instructor IS-A Person, a Person IS-A Object and so on. Inheritance thus gave developers a new axis on which to analyze commonality and variability.

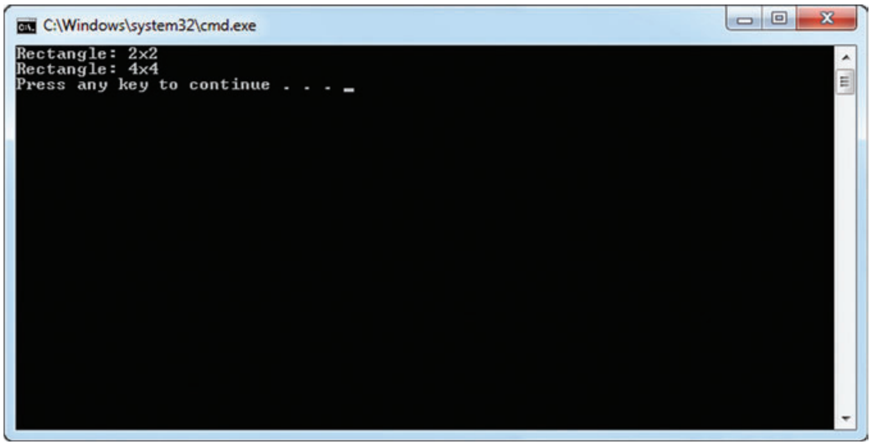
In the days of C++, the implementation-inheritance approach stood alone, but as time and experience progressed, interface inheritance emerged as an alternative. In essence, the introduction of interface inheritance into the designer's toolbox allowed for a lighter-weight inheritance relationship, declaring that a type IS-A different type, but without the behavior or structure of the parent type. Interfaces thus provide a mechanism for "grouping" types along an inheritance axis without enforcing any particular restrictions on their implementation.

Consider, for example, the canonical object-oriented example, that of a hierarchy of geometric shapes that can be drawn (if only figuratively) to screen:

```
class Rectangle
{
  public int Height { get; set; }
  public int Width { get; set; }
  public void Draw() { Console.WriteLine("Rectangle: {0}x{1}", Height,
Width); }
}

class Circle
{
  public int Radius { get; set; }
  public void Draw() { Console.WriteLine("Circle: {0}r", Radius); }
}
```

Figure 1 Deriving a Square

```
class Rectangle : Shape
{
  public virtual int Height { get; set; }
  public virtual int Width { get; set; }
  public override void Draw() {
    Console.WriteLine("Rectangle: {0}x{1}", Height, Width); }
}

class Square : Rectangle
{
  private int height;
  private int width;
  public override int Height {
    get { return height; }
    set { Height = value; Width = Height; }
  }
  public override int Width {
    get { return width; }
    set { Width = value; Height = Width; }
  }
}
```



```
Rectangle: 2x2
Rectangle: 4x4
Press any key to continue . . .
```

Figure 2 **A Surprising Result with Grow Code**

The commonality between the classes suggests that a superclass is in order here, to avoid repeating that commonality in every drawable geometric shape:

```
abstract class Shape
{
  public abstract void Draw();
}

class Rectangle : Shape
{
  public int Height { get; set; }
  public int Width { get; set; }
  public override void Draw() {
    Console.WriteLine("Rectangle: {0}x{1}", Height, Width); }
}

class Circle : Shape
{
  public int Radius { get; set; }
  public override void Draw() { Console.WriteLine("Circle: {0}r",
Radius); }
  }
```

So far, so good—most developers would take no issue with what's been done thus far. Unfortunately, a problem lies in wait for the unwary.

Liskov Rides Again

The catch here is known as the Liskov Substitution Principle: any type that inherits from another must be completely substitutable for that other. Or, to use the words that originally described the principle, "Let q(x) be a property provable about objects x of type T. Then q(y) should be true for objects y of type S where S is a subtype of T."

What that means in practice is that any particular derivation of Rectangle, such as a Square class, must ensure that it obeys the same behavioral guarantees provided by the base. Because a Square is essentially a Rectangle with the guarantee that both the Height and Width are always the same, it seems reasonable to write Square like the example in **Figure 1**.

Notice how Height and Width properties are now virtual so as to avoid any kind of accidental shadowing or slicing behavior when overriding them in the Square class. So far so good.

Next, a Square gets passed in to a method that takes a Rectangle and "grows" it (what graphics geeks sometimes call a "transform"):

```
class Program
{
  static void Grow(Rectangle r)
  {
    r.Width = r.Width + 1;
    r.Height = r.Height + 1;
  }

  static void Main(string[] args)
  {
    Square s = new Square();
    s.Draw();
    Grow(s);
    s.Draw();
  }
}
```

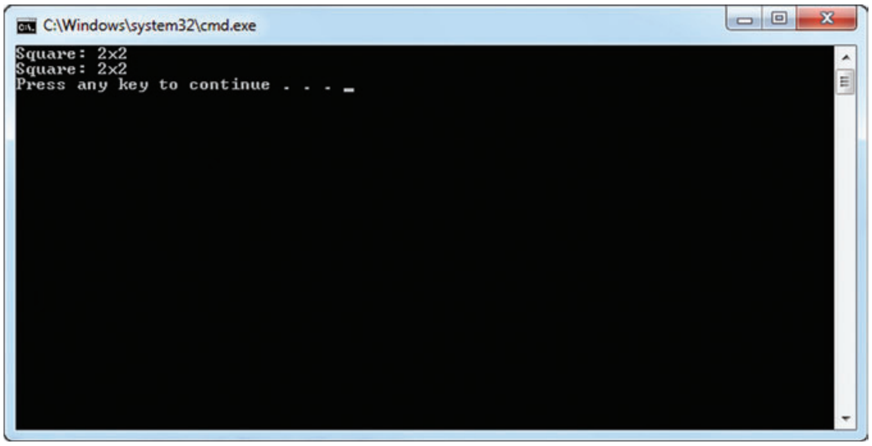
Figure 2 shows the net result of calling this code, which is not what you might expect.

Figure 3 **Conversion Operation**

```
class Square : Shape
{
  public int Edge { get; set; }
  public Rectangle AsRectangle() {
    return new Rectangle { Height = this.Edge, Width = this.Edge };
  }
  public override void Draw() { Console.WriteLine("Square: {0}x{1}",
Edge, Edge); }
}

class Program
{
  static void Grow(Rectangle r)
  {
    r.Width = r.Width + 1;
    r.Height = r.Height + 1;
  }

  static void Main(string[] args)
  {
    Square s = new Square() { Edge = 2 };
    s.Draw();
    Grow(s.AsRectangle());
    s.Draw();
  }
}
```

Figure 4 **The C# Conversion Operator Facility**

```
class Square : Shape
{
  public int Edge { get; set; }
  public static implicit operator Rectangle(Square s) {
    return new Rectangle { Height = s.Edge, Width = s.Edge };
  }
  public override void Draw() { Console.WriteLine("Square: {0}x{1}",
Edge, Edge); }
}

class Program
{
  static void Grow(Rectangle r)
  {
    r.Width = r.Width + 1;
    r.Height = r.Height + 1;
  }

  static void Main(string[] args)
  {
    Square s = new Square() { Edge = 2 };
    s.Draw();
    Grow(s);
    s.Draw();
  }
}
```



```
Square: 2x2
Square: 2x2
Press any key to continue . . . _
```

Figure 5 **The Result of Using the C# Conversion Operator Facility**

The problem here, as careful readers may have already surmised, is that each property implementation assumes it's being called in isolation, and thus has to act independently to ensure the Height==Width constraint around Square at all times. The Grow code, however, assumes that a Rectangle is being passed in, and remains entirely ignorant of the fact that it's a Square coming in (as intended!), and acts in a manner entirely appropriate for Rectangles.

The core of the problem? Squares aren't rectangles. They have a lot of similarity, granted, but at the end of the day, the constraints of a square don't hold for rectangles (which is also true, by the way, for ellipses and circles), and trying to model one in terms of the other is based on a fallacy. It's tempting to inherit Square from Rectangle, particularly because it lets us reuse some code, but it's a false premise. In fact, I'll even go so far as to suggest that one should never use inheritance to foster reuse until the Liskov Substitution Principle for those two types has been proven to be true.

This example isn't new—Robert "Uncle Bob" Martin (bit.ly/4F2R6t) discussed Liskov and this exact example back in the mid-90s when talking to C++ developers. Some problems like this can be solved partly by using interfaces to describe the relationships, but that doesn't help this particular case, because Height and Width remain separate properties.

Is there a solution in this case? Not really, not while keeping the Square-as-derived-from-Rectangle relationship in place. The best answer comes from making Square a direct descendent of Shape, and abandoning the inheritance approach entirely:

```
class Square : Shape
{
  public int Edge { get; set; }
    public override void Draw() { Console.WriteLine("Square: {0}x{1}",
Edge, Edge); }
}

class Program
{
    static void Main(string[] args)
    {
      Square s = new Square() { Edge = 2 };
      s.Draw();
      Grow(s);
      s.Draw();
    }
}
```

Of course, now we have the problem that Square can't be passed in to Grow at all, and it does seem like there's a potential code-reuse relationship there. We can solve this in one respect by providing a view of the Square as a Rectangle using a conversion operation, as shown in **Figure 3**.

It works—but it's a bit awkward. We might also use the C# conversion operator facility to make it easier to convert Squares to Rectangles, as shown in **Figure 4**.

This approach, while perhaps strikingly different from what's expected, offers the same client perspective as before, but without the problems of the earlier implementation, as **Figure 5** shows.

In fact, we have a different problem—where before the Grow method modified the Rectangle being passed in, now it appears that it's doing nothing, largely because it's modifying a copy of the Square, not the original Square itself. We could fix this by having the conversion operator return a new subclass of Rectangle that holds a secret reference back to this Square instance, so that modifications to the Height and Width properties will in turn come back and modify the Square's Edge ... but then we're back to the original problem!

No Happy Ending Here

In Hollywood, movies have to end in a fashion commensurate with audience expectations or face being rejected at the box office. I'm not a moviemaker, so I feel no compulsion to present the readers of this column with a happy ending in every case. This is one of those cases: trying to keep the original code in place and make it all work just creates deeper and deeper hacks. Solutions might be to move the Grow or Transform method directly on to the Shape hierarchy or just make the Grow method return the modified object rather than modify the object passed in (which is something we'll talk about in another column), but in short, we can't keep the original code in place and keep everything working correctly.

All of this is designed to showcase precisely one thing: object-oriented developers are comfortable with modeling commonality and variability with inheritance, perhaps too much so. Remember, if you choose to use the inheritance axis to capture commonality, you have to ensure this commonality holds across the entire hierarchy if subtle bugs like this are to be avoided.

Remember, too, that inheritance is always a positive variability (adding new fields or behaviors), and that modeling negative variability in inheritance (which is what Square tried to do) is almost always a recipe for disaster along Liskovian lines. Ensure that all inheritance-based relationships involve a positive commonality, and things should be good. Happy coding! ■

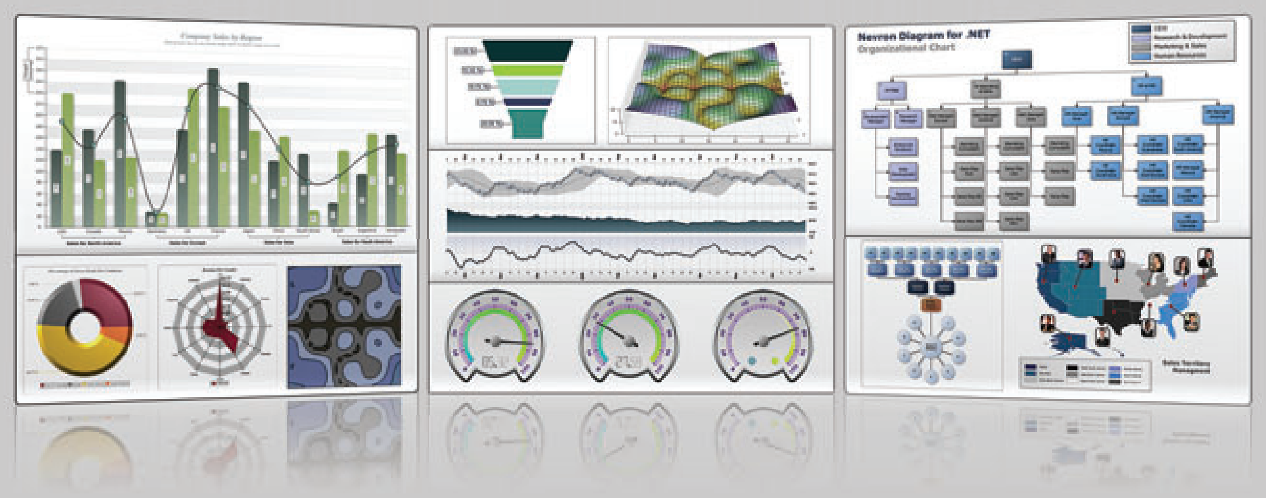
TED NEWARD *is a principal with Neward & Associates, an independent firm specializing in enterprise .NET Framework and Java platform systems. He's written more than 100 articles, is a C# MVP and INETA speaker and has authored and coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He also consults and mentors regularly. Reach him at ted@tedneward.com with questions or consulting requests, and read his blog at blogs.tedneward.com.*

THANKS to the following technical expert for reviewing this article: *Anthony Green*

NEVRON
visualize your success

Let us help you visualize your success

Nevron provides the essential components for the creation of advanced digital dashboards, scientific and financial applications, diagrams and MMI interfaces for a variety of .NET centric technologies.



Nevron components integrate seamlessly in web and desktop applications, SQL Server Reporting Services 2005/2008 reports and SharePoint 2007/2010 portals and deliver an unmatched set of enterprise-grade features. That is why Nevron is the trusted vendor by many Fortune 500 companies for their most demanding data visualization needs.

Developers

Nevron .NET Vision incorporates components that help you create enterprise grade digital dashboards, scorecards, diagrams, maps, MMI interfaces and much more.

- Chart for .NET
- Diagram for .NET
- Gauge for .NET
- Map for .NET
- User Interface for .NET

IT Professionals

Nevron Reporting Services Vision instantly enhances your SQL Server Reporting Services 2005/2008 reports with the industry leading data visualization technology.

- Chart for SSRS
- Gauge for SSRS

Nevron SharePoint Vision instantly converts your SharePoint pages into advanced dashboards and reports, that unite powerful data analysis with industry leading data visualization.

- Chart for SharePoint
- Gauge for SharePoint

MAKE SURE THAT YOUR DATA IS MAKING THE VISUAL STATEMENT IT DESERVES BY DOWNLOADING YOUR FREE EVALUATION COPY FROM WWW.NEVRON.COM TODAY.

Microsoft .NET ASP.net AJAX Microsoft SharePoint 2007/2010 Microsoft SQL Server 2005/2008 Microsoft Visual Studio PARTNER

www.nevron.com | sales@nevron.com | 1888 201 6088

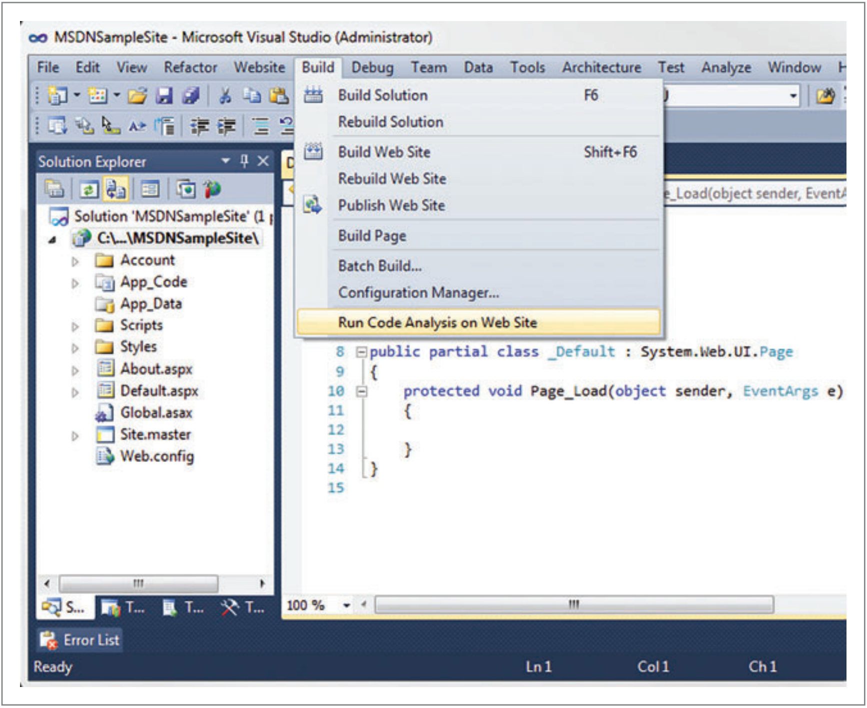
SACHA FAUST

Improving ASP.NET Security with Visual Studio 2010 Code Analysis

Anyone doing ASP.NET development probably admits, openly or not, to introducing or stumbling upon a security issue at some point during their career. Developers are often pressured to deliver code as quickly as possible, and the complexity of the platform and vast number of configuration options often leaves the application in a less than desirable security state. In addition, the configuration requirements for debugging and production are different, which can often introduce debugging settings in production, causing a variety of issues.

Over the years, the ASP.NET platform has matured and better documentation has been made available through MSDN and community blogs, but knowing which feature or configuration setting to use is often troublesome. Even with good knowledge of the security functionality, mistakes can happen that could result in security vulnerabilities in your application.

Peer code review is a useful process and a good way to catch issues early. Still, not everyone has the time or budget—or knowledgeable peers at hand—for such review.



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

    }
}
```

Figure 1 **Running Code Analysis on a Sample Web Site**

Since the introduction of code analysis in Visual Studio 2005, developers have been able to automatically analyze their code to see if it complies with a series of best practices ranging from design, maintainability, performance and security. So far, code analysis has been a great tool, but it hasn't focused on providing best security practice guidance for ASP.NET—until now.

Peer code review is a useful process and a good way to catch issues early.

In this article I'll introduce you to the new ASP.NET code analysis rules that can be used with Visual Studio code analysis as well as with the standalone FxCop application to improve the security of your ASP.NET applications.

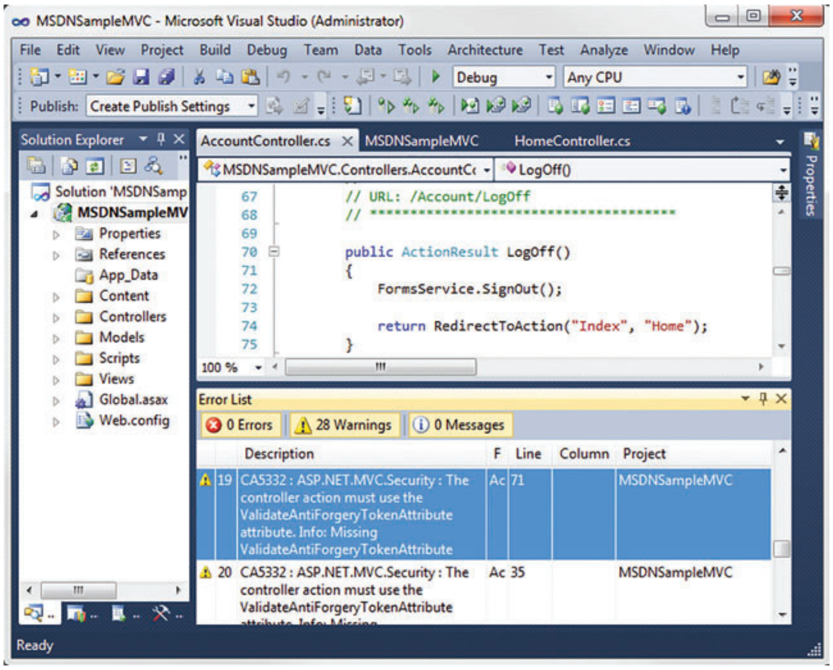
Overview

You can download the ASP.NET security code analysis rules package for Visual Studio 2010 and FxCop version 10.0 from go.microsoft.com/?linkid=9750555. The installation contains three new rules packages:

- **ASP.NET.Security:** This category focuses on security best practices related to how System.Web.Ui.Page properties are initialized.
- **ASP.NET.MVC.Security:** This category focuses on security best practices related to how ASP.NET MVC is used.
- **ASP.NET.Security.Configuration:** This category focuses on security best practices related to configuration elements under the web.config files.

Once the rules package is installed, you can start reviewing the security of your Web application automatically by clicking on the Run Code Analysis on Web Site button under the Build menu (see **Figure 1**). The analysis will review each Page class and web.config file of your application against a series of security best practices for ASP.NET.

For example, one widespread security vulnerability in Web applications is cross-site request forgery, which allows an attacker to execute commands as another user. The common mitigation for this vulnerability is to use the Page.ViewStateUserKey property (bit.ly/cTSHM0). You can also employ the AntiForgeryToken in ASP.NET MVC (bit.ly/ciiQIP). Both techniques prevent a malicious replay attack on your application. The code analysis will make sure that the appropriate mitigation is being used in your application.

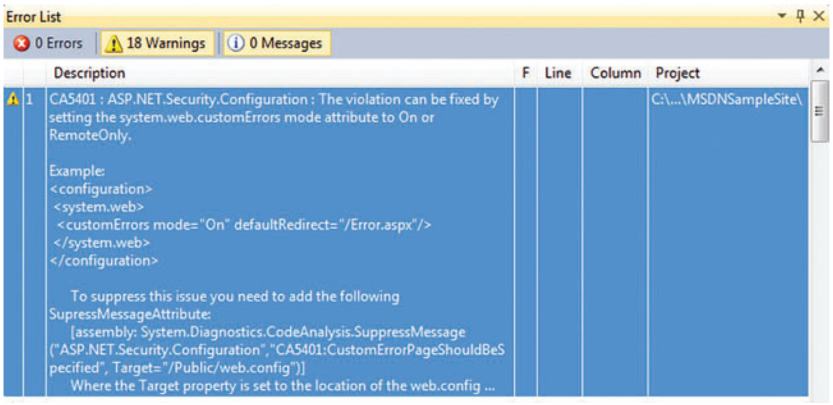


```
// URL: /Account/LogOff
// **************************************

public ActionResult LogOff()
{
    FormsService.SignOut();

    return RedirectToAction("Index", "Home");
}
```

Error List: 0 Errors | 28 Warnings | 0 Messages

	Description	F	Line	Column	Project
19	CA5332 : ASP.NET.MVC.Security : The controller action must use the ValidateAntiForgeryTokenAttribute attribute. Info: Missing ValidateAntiForgeryTokenAttribute	Ac	71		MSDNSampleMVC
20	CA5332 : ASP.NET.MVC.Security : The controller action must use the ValidateAntiForgeryTokenAttribute attribute. Info: Missing	Ac	35		MSDNSampleMVC

Figure 2 **Violations Are Listed in the Error List Warnings Tab**

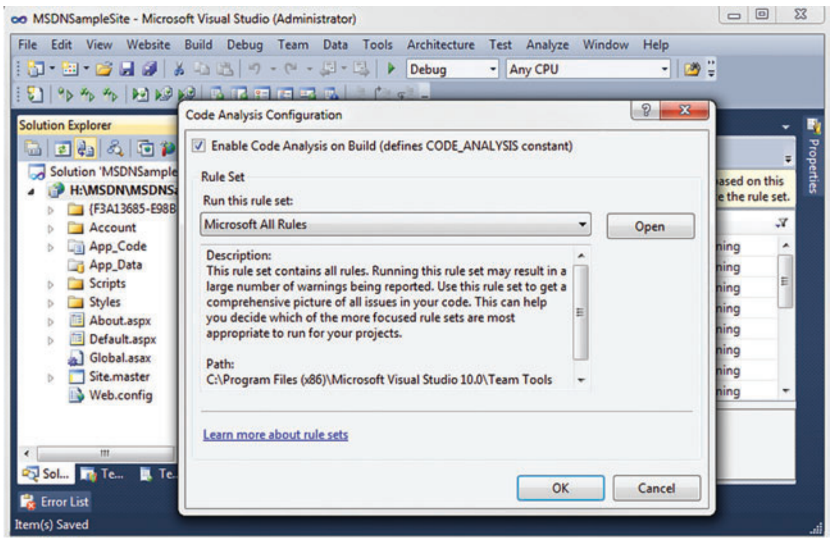


Error List: 0 Errors | 18 Warnings | 0 Messages

	Description	F	Line	Column	Project
1	CA5401 : ASP.NET.Security.Configuration : The violation can be fixed by setting the system.web.customErrors mode attribute to On or RemoteOnly.				C:\...\MSDNSampleSite\

```
Example:
<configuration>
 <system.web>
  <customErrors mode="On" defaultRedirect="/Error.aspx"/>
 </system.web>
</configuration>

    To suppress this issue you need to add the following
SupressMessageAttribute:
    [assembly: System.Diagnostics.CodeAnalysis.SuppressMessage
("ASP.NET.Security.Configuration","CA5401:CustomErrorPageShouldBeS
pecified", Target="/Public/web.config")]
    Where the Target property is set to the location of the web.config ...
```

Figure 3 **Detailed Information in the Warnings Section**



Code Analysis Configuration

[✓] Enable Code Analysis on Build (defines CODE_ANALYSIS constant)

Rule Set
Run this rule set: Microsoft All Rules

Description:
This rule set contains all rules. Running this rule set may result in a large number of warnings being reported. Use this rule set to get a comprehensive picture of all issues in your code. This can help you decide which of the more focused rule sets are most appropriate to run for your projects.

Path:
C:\Program Files (x86)\Microsoft Visual Studio 10.0\Team Tools

Learn more about rule sets

Figure 4 **Enabling Code Analysis During Build**

Each rule includes a clear indication of what needs to be fixed and how to fix it.

A common bit of feedback I've heard from developers running code analysis for the first time is the overwhelming number of warnings returned (see **Figure 2**). It's easy to feel like you're on your own to figure out how to fix all of them.

To eliminate some of the burden of fixing each warning, each rule includes a clear indication of what needs to be fixed and how to fix it, along with some references if you need more information before applying the change (see **Figure 3**).

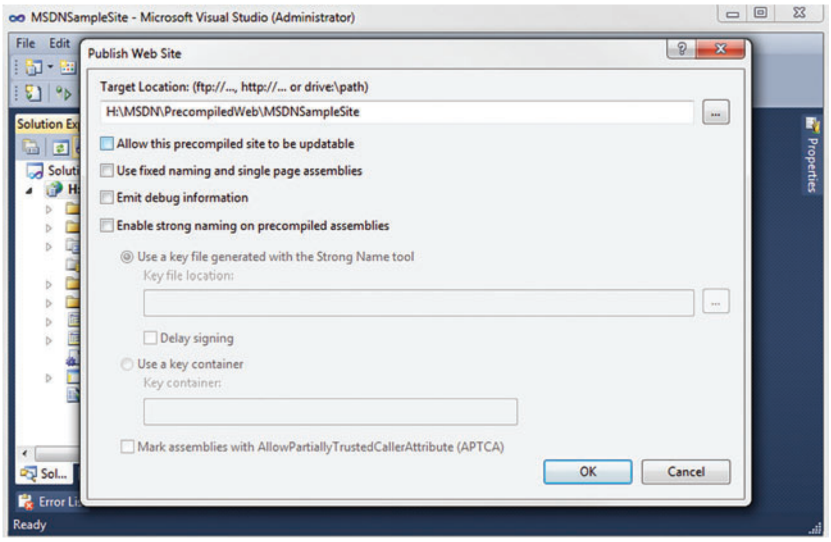
Code analysis can also be configured to run after each build by clicking Website | Configure Code Analysis for Website and then checking the "Enable Code Analysis on Build (defines CODE_ANALYSIS constant)" option (see **Figure 4**).

Code Analysis with FxCop

The code analysis feature is only available in Visual Studio Premium and Ultimate versions. However, you can also use the standalone FxCop tool to perform ASP.NET code analysis. FxCop is available as part of the Windows SDK. The Windows SDK 7.1 release is available from bit.ly/dzCizq.

When using the standard FxCop tool, a little bit more work is required to perform the analysis. I'll walk you through the steps to get it working.

Normally when you compile your Web project, the page markup—the page code not included in a code-behind file—is not compiled and is left intact in the Web root of your application. When the first user requests the page, the markup is compiled into separate assemblies. This allows a site to be updatable without requiring everything else to be recompiled. (For details about the ASP.NET page compilation process, see msdn.microsoft.com/library/ms366723.)


MSDNSampleSite - Microsoft Visual Studio (Administrator)

Publish Web Site

Target Location: (ftp://..., http://... or drive:\path)

H:\MSDN\PrecompiledWeb\MSDNSampleSite

- [ ] Allow this precompiled site to be updatable
- [ ] Use fixed naming and single page assemblies
- [ ] Emit debug information
- [ ] Enable strong naming on precompiled assemblies

Use a key file generated with the Strong Name tool

Key file location:

- [ ] Delay signing

Use a key container

Key container:

- [ ] Mark assemblies with AllowPartiallyTrustedCallerAttribute (APTCA)

Figure 5 **Publishing Web Site with Precompilation**

Because not all code is automatically compiled, some of the code isn't visible during analysis and important security issues could be missed. To make sure that all code is available during analysis, you need to force the precompilation of all pages. Precompilation can be achieved by using the Publish Web Site tool, which can be started by clicking Build | Public Web Site. The tool allows you to configure how the Web site will be published, and this is where precompilation can be enabled. Simply uncheck the "Allow this precompiled site to be updatable" option and click OK (see **Figure 5**). This will result in a fully compiled site ready for analysis.

Now that you have a fully compiled site, unleash FxCop on it.

Now that you have a fully compiled site, unleash FxCop on it.

ASP.NET analysis requires functionality that is only available in the command-line version of FxCop, so open a command prompt and navigate to the FxCop installation. This will most likely be one of the following, depending on whether you're running a 32- or 64-bit version of Windows:

```
C:\Program Files (x86)\Microsoft FxCop 10.0
C:\Program Files\Microsoft FxCop 10.0
```

From the FxCop folder you can run Fxcopcmd.exe to start your code analysis. For an ASP.NET Web site you simply need to use a command like this:

```
fxcopcmd.exe /file:"H:\MSDN\PrecompiledWeb\
  MSDNSampleSite\bin" /rule:
  AspNetConfigurationSecurityRules.dll
  /rule:AspNetMvcSecurityRules.dll
  /rule:ASPNetSecurityRules.dll /aspnet /console
```

Let's walk through this so you understand the options I'm using.

The /file option indicates what assemblies are to be analyzed. In this example, my precompiled site assemblies are under H:\MSDN\PrecompiledWeb\MSDNSampleSite\bin.

The /rule option indicates which rules to use during analysis. For the purpose of this example, I'm only using three ASP.NET security rules: AspNetConfigurationSecurityRules.dll, AspNetMvcSecurityRules.dll and ASPNetSecurityRules.dll.

The /aspnet option enables ASP.NET analysis, and the /console option directs analysis output to the command window. You can see the results in **Figure 6**. More information about the Fxcopcmd and its various options can be found at msdn.microsoft.com/library/bb429474(VS.80).

Wrapping Up

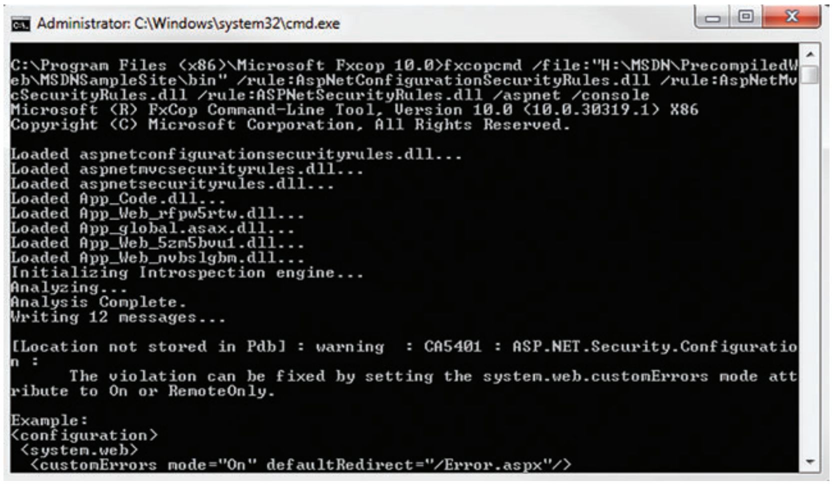
Making ASP.NET sites more secure can be a difficult task, but the ASP.NET security code analysis rules does a lot of the work for you by identifying some significant threats. As you've seen in this article, the process is simple and can be configured to run each time you build, giving you early identification of issues.

I recommend deploying the rules to each developer machine and also adding them as part of a Team Foundation Sever or other repository check-in policy. This enables individual developers to verify their code at build time and also enforces the policy so that no code can be checked that doesn't meet best practices.

You can also implement your own custom code analysis rules. If you're interested in going down this route, there's some excellent information in a blog post by Duke Kamstra on the Code Analysis Team Blog (bit.ly/bIpP38). You can also find a useful walkthrough of the process on Tatham Oddie's blog (bit.ly/5tFrMw). ■

Sacha Faust *is a developer on the Microsoft Office 365 platform team. You can follow Faust at blogs.msdn.com/sfaust.*

Thanks to the following technical expert for reviewing this article: *Bryan Sullivan*


Administrator: C:\Windows\system32\cmd.exe

```
C:\Program Files (x86)\Microsoft Fxcop 10.0>fxcopcmd /file:"H:\MSDN\PrecompiledW
eb\MSDNSampleSite\bin" /rule:AspNetConfigurationSecurityRules.dll /rule:AspNetMv
cSecurityRules.dll /rule:ASPNetSecurityRules.dll /aspnet /console
Microsoft (R) FxCop Command-Line Tool, Version 10.0 (10.0.30319.1) X86
Copyright (C) Microsoft Corporation, All Rights Reserved.

Loaded aspnetconfigurationsecurityrules.dll...
Loaded aspnetmvcsecurityrules.dll...
Loaded aspnetsecurityrules.dll...
Loaded App_Code.dll...
Loaded App_Web_rfpw5rtw.dll...
Loaded App_global.asax.dll...
Loaded App_Web_5zm5bvu1.dll...
Loaded App_Web_nvbslgbm.dll...
Initializing Introspection engine...
Analyzing...
Analysis Complete.
Writing 12 messages...

[Location not stored in Pdb] : warning  : CA5401 : ASP.NET.Security.Configuratio
n :
      The violation can be fixed by setting the system.web.customErrors mode att
ribute to On or RemoteOnly.

Example:
<configuration>
 <system.web>
  <customErrors mode="On" defaultRedirect="/Error.aspx"/>
```

Figure 6 **Running ASP.NET Rules Using Fxcopcmd.exe**

Know it all.

We've got all the answers.

- Thousands of code samples
- Over 29,000 articles
- Practical answers in active forums
- MFC/Microsoft® Visual C++®, Visual C#®, Visual Basic® .NET, ASP.NET, SQL Server®, Windows® Phone, and more!
- QuickAnswers
- Over 7.4 million members
- Start your FREE membership TODAY!

THE CODE PROJECT™

WWW.CODEPROJECT.COM

CHARLES PETZOLD

Silverlight, Windows Phone 7 and the Multi-Touch Thumb

For many Silverlight programmers, the most exciting news about Windows Phone 7 is its support for Silverlight as one of its two programming interfaces. (The other one is XNA.) Not only can Silverlight programmers leverage their existing knowledge and skills in writing new applications for the phone, but they should be able to build Silverlight programs for the Web and the phone that share code.

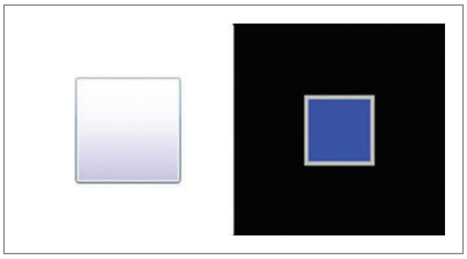


Figure 1 **The Silverlight and Windows Phone Thumb Controls**

Of course, sharing code—particularly UI code—is rarely as easy as it first seems. The version of Silverlight used in the phone is called Silverlight for Windows Phone, and it's mostly a stripped-down implementation of Silverlight 3. When contemplating a shared-code application, you'll want to take a close look at the documentation: For each Silverlight class, the online documentation indicates which environments support that class. Within each class, lists of properties, methods and events use icons to indicate Windows Phone 7 support.

A Silverlight application for the Web gets user input through the keyboard, mouse and perhaps multi-touch. In a Windows Phone 7 program, multi-touch is the primary means of input. There's no mouse, and while there might be a hardware keyboard on the phone, Silverlight programs can rely only on the existence of a virtual keyboard—the Software Input Panel, or SIP—and only through the TextBox control.

Moving from the mouse to multi-touch will require some thought.

If your existing Silverlight programs never directly obtain keyboard or mouse input and rely entirely on controls, you won't have to worry about the conversion to multi-touch. Also, if your programs contain their own mouse logic, you can actually retain that logic when porting the program to the phone.

On the phone, primary touch events are converted to mouse events, so your existing mouse logic should work fine. (A primary touch event is the entire activity of a finger that first touches the screen when no other fingers are in contact with the screen.)

Moving from the mouse to multi-touch will require some thought: Both Silverlight for the Web and Silverlight for Windows Phone support the static Touch.FrameReported event, but this event is a rather low-level interface to multi-touch. I focused on this event in my article "Finger Style: Exploring Multi-Touch Support in Silverlight" in the March 2010 issue (msdn.microsoft.com/magazine/ee336026).

Silverlight for Windows Phone supports a subset of the Manipulation events that originated in the Surface SDK and have since become part of Windows Presentation Foundation (WPF). It's an example of how multi-touch is becoming more mainstream in steps. The phone supports only the translation and scaling functions, not rotation, and does not implement inertia, although sufficient information is available to implement inertia on your own. These Manipulation events are *not* yet supported in the Web version of Silverlight.

In summary, if you want to share code between Silverlight for the Web and Silverlight for Windows Phone, you'll be sticking either with mouse events or with Touch.FrameReported.

Consider the Thumb

However, there's another option: If you need only the translation support of the Manipulation events, and you don't want to worry about whether the input is coming from the mouse or touch, there is a control that provides this support in a very pure form. This control is the Thumb.

It's possible that you've never actually encountered the Thumb. The Thumb control is hidden away in the System.Windows.Controls.Primitives namespace and is primarily intended for ScrollBar and Slider templates. But you can also use it for other chores, and I've recently come to think of the Thumb as a high-level implementation of the translation feature of the Manipulation events.

Now, the Thumb isn't a truly "multi"-touch control—it supports only one touch at a time. However, exploring the Thumb in some detail will give you an opportunity to experiment with supporting touch computing along with sharing code between a Silverlight application and a Windows Phone 7 application.

Code download available at code.msdn.microsoft.com/mag201012UIFrontiers.

The Thumb defines three events:

- DragStarted is fired when the user first touches the control with a finger or mouse.
- DragDelta indicates movement of the mouse or finger relative to the screen.
- DragCompleted indicates the mouse or finger has lifted.

The DragDelta event is accompanied by event arguments with the properties HorizontalChange and VerticalChange that indicate the mouse or finger movement since the last event. You'll generally handle this event by adding the incremental changes to the X and Y properties of a TranslateTransform set to a RenderTransform property of some draggable element, or the Canvas.Left and Canvas.Top attached properties.

In its default state, the Thumb is rather plain. As with other controls, the HorizontalAlignment and VerticalAlignment properties are set to Stretch so the Thumb normally fills the area allowed for it. Otherwise, the Silverlight Thumb is just four pixels square. In Silverlight for Windows Phone, the Thumb is 48 pixels square, but visually it's really just 24 pixels square with a 12-pixel wide transparent border on all four sides.

At the very least, you'll probably want to set an explicit Height and Width on the Thumb. **Figure 1** shows the Silverlight and Windows Phone 7 versions side by side, with the default light-on-dark color theme of the phone. For both I've set the Height and Width to 72 and Background to Blue, which in the Silverlight version becomes a gradient that changes when the Thumb is pressed. Neither Thumb pays attention to the Foreground property.

Very often you'll want not only to resize the Thumb, but also to apply a ControlTemplate that redefines the control's visuals. This ControlTemplate can be extremely simple.

Sharing Controls

Suppose you want a simple control that lets the user drag bitmaps around the screen. A very easy approach is to put both an Image element and a Thumb in a single-cell Grid, with the Thumb the same size as the Image and overlaying it. If the ControlTemplate for the Thumb is just a transparent Rectangle, the Thumb is invisible but it still fires drag events.

Let's try to create such a control usable in both regular Silverlight and Windows Phone 7 projects. I'll assume you have the Windows Phone 7 Developer Tools installed (create.msdn.com). These tools allow you to create Windows Phone 7 projects from Visual Studio.

Begin by creating a regular Silverlight 4 project called DragImage. The resulting DragImage solution contains the customary DragImage project (which is the Silverlight program itself) and a DragImage.Web project (which hosts the Silverlight program in an HTML or ASP.NET page).

Next, add a new project of type Windows Phone Application to the solution. Call this project DragImage.Phone. (It's likely you won't want that name showing up in the program list of the phone or the phone emulator, so you can change the display name in the Title attribute of the App tag in the WMAppManifest.xml file.)

By right-clicking either the DragImage.Web project or the DragImage.Phone project, you'll get a context menu from which you can select Set as StartUp Project and run either the regular Silverlight program or the Windows Phone 7 program. A toolbar drop-down in Visual Studio lets you deploy the phone program to either an actual phone device or the phone emulator. (Visual Studio won't build the projects if this drop-down is set for Windows Phone 7 Device and no phone is attached.)

Figure 2 **DraggableImage.xaml**

```
<UserControl x:Class="DragImage.DraggableImage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Name="ctrl">

  <Grid x:Name="LayoutRoot">
    <Grid HorizontalAlignment="Left"
          VerticalAlignment="Top">
      <Image Name="image" Stretch="None"
             Source="{Binding ElementName=ctrl, Path=Source}" />
      <Thumb DragDelta="OnThumbDragDelta">
        <Thumb.Template>
          <ControlTemplate>
            <Rectangle Fill="Transparent" />
          </ControlTemplate>
        </Thumb.Template>
      </Thumb>
      <Grid.RenderTransform>
        <TranslateTransform x:Name="translate" />
      </Grid.RenderTransform>
    </Grid>
  </Grid>
</UserControl>
```

In the DragImage project (the regular Silverlight project), add a new item of type Silverlight User Control. Call it DraggableImage. As usual, Visual Studio creates DraggableImage.xaml and DraggableImage.xaml.cs files for this control.

Figure 2 shows DraggableImage.xaml with the visual tree of the control. The standard outer Grid named LayoutRoot will occupy the full dimensions of the control's container; the inner Grid is aligned at the upper-left corner, but there's a TranslateTransform assigned to its RenderTransform property to move it within the outer Grid. This inner Grid holds an Image element with a Thumb control on top with its Template property set to a visual tree containing only a transparent Rectangle.

Notice that the Source property of the Image element is bound to the Source property of the control itself. That property is defined in the DraggableImage.xaml.cs file shown in **Figure 3**. That file also processes the DragDelta event from the Thumb by changing the X and Y properties of the TranslateTransform.

To share that control with the Windows Phone 7 project, right-click the DragImage.Phone project and select Add | Existing Item to bring up the Add Existing Item dialog box. Navigate to the DragImage project directory. Select DraggableImage.xaml and DraggableImage.xaml.cs, but don't click the Add button. Instead, click the little arrow to the right of the Add button and select Add as Link. The files show up in the DragImage.Phone project with little arrows on the icons indicating that the files are shared between the two projects.

Now you can make changes to the DraggableImage files and both projects will use the revised versions.

To test it out, you'll need a bitmap. Store the bitmap in an Images directory within each of the projects. (You don't need to make copies of the bitmap; you can add the bitmap to the Images directory using a link.)

Figure 3 **DraggableImage.xaml.cs**

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;

namespace DragImage {
  public partial class DraggableImage : UserControl {
    public static readonly DependencyProperty SourceProperty =
      DependencyProperty.Register("Source",
      typeof(ImageSource),
      typeof(DraggableImage),
      new PropertyMetadata(null));

    public DraggableImage() {
      InitializeComponent();
    }

    public ImageSource Source {
      set { SetValue(SourceProperty, value); }
      get { return (ImageSource)GetValue(SourceProperty); }
    }

    void OnThumbDragDelta(object sender, DragDeltaEventArgs args) {
      translate.X += args.HorizontalChange;
      translate.Y += args.VerticalChange;
    }
  }
}
```

There should be two MainPage.xaml files floating around. One is from the regular Silverlight project and the other is from the Windows Phone 7 project. In MainPage.xaml for the Silverlight project, add an XML namespace binding called (traditionally) “local”:

```
xmlns:local="clr-namespace:DragImage"
Now you can add DraggableImage to the page:
<Grid x:Name="LayoutRoot" Background="White">
  <local:DraggableImage
    Source="Images/BuzzAldrinOnTheMoon.png" />
</Grid>
```

The MainPage class for the Windows Phone 7 project is in a namespace called DragImage.Phone, but the shared DraggableImage class is in the namespace DragImage. You’ll need an XML namespace binding for the DragImage namespace, which you can call “shared”:

```
xmlns:shared="clr-namespace:DragImage"
```

Now you can add DraggableImage to the content area of the page:

```
<Grid x:Name="ContentPanel"
  Grid.Row="1" Margin="12,0,12,0">
  <shared:DraggableImage
    Source="Images/BuzzAldrinOnTheMoon.png" />
</Grid>
```

Figure 4 **The Thumb Style from TextTransformer.xaml**

```
<Style x:Key="thumbStyle" TargetType="Thumb">
  <Setter Property="HorizontalAlignment"
          Value="Left" />
  <Setter Property="VerticalAlignment"
          Value="Top" />
  <Setter Property="Width"
          Value="{StaticResource ThumbSize}" />
  <Setter Property="Height"
          Value="{StaticResource ThumbSize}" />
  <Setter Property="RenderTransform">
    <Setter.Value>
      <TranslateTransform
        X="{StaticResource HalfThumbOffset}"
        Y="{StaticResource HalfThumbOffset}" />
    </Setter.Value>
  </Setter>
</Style>
```

That’s probably the simplest way you can share a control between two Silverlight projects, one for the Web and one for Windows Phone 7. Because the control uses the Thumb, both programs work with the mouse or touch.

The downloadable code for the DragImage solution also includes a project named DragImage.Wpf, which is a WPF program that also uses this control. In the general case, however, sharing controls between Silverlight and WPF is harder than sharing controls between Silverlight and Windows Phone 7.

Color and Resolution

Aside from mouse and touch input, when attempting to share code between Silverlight and Windows Phone 7, you’ll need to deal with two other issues: color and video resolution.

On the desktop, Silverlight displays black text on a white background. (However, a Silverlight program could use the SystemColors class in order to display the Windows colors selected by the user.) By default, Windows Phone 7 displays white text on a black background except if the user changes the color theme to display black on white. Windows Phone 7 provides handy, predefined resource keys, such as PhoneForegroundBrush and PhoneBackgroundBrush, to help a program use the selected color scheme.

Any code or markup shared between Silverlight and Windows Phone 7 that uses explicit colors will have to figure out some way to determine the platform on which it’s running to get the correct colors.

All Silverlight coordinates are in units of pixels, and that rule applies to the phone as well.

The video resolution problem is a little trickier. All Silverlight coordinates are in units of pixels, and that rule applies to the phone as well. The average desktop video display probably has a resolution somewhere in the vicinity of 100 dots per inch (DPI). (For example, suppose a 21-inch video display handles 1600 × 1200 pixels, or 2000 pixels diagonally. That’s a resolution of 105 DPI.) By default, Windows assumes that the display resolution is 96 DPI, although the user can change that to make the screen easier to read.

A Windows Phone 7 device has a screen that’s 480 × 800 pixels with a diagonal of 933 pixels. Yet the screen measures only 3.5 inches diagonally, which means the resolution is about 264 DPI, some 2.75 times the resolution of the desktop display.

This means that shared elements of a particular size that look fine on the desktop are going to be too small on the phone. However, the viewing distance of the phone is usually shorter than for desktop displays, so the elements don’t have to be increased by a full 2.75 times to be visible on the phone.

How big should the Thumb be for touch purposes? One criterion I’ve read indicates that touch targets should be 9 millimeters (or 0.25 inches) wide and high. On a desktop display with a resolution of 96 pixels to the inch, that’s 34 pixels—but on the phone it’s 93 pixels.

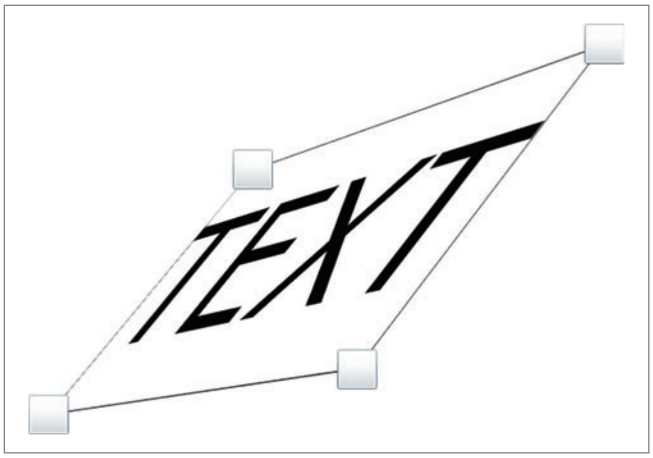


Figure 5 **The TextTransform Program in the Browser**

On the other hand, the standard button on a Windows Phone 7 device is only 72 pixels tall, and that seems adequate. Perhaps the best approach is to experiment until you find something that's easy to use but isn't too clunky.

Making Adjustments

Traditionally, programs adjusted themselves for different platforms using preprocessor directives for conditional compilation. A Silverlight program defines the conditional compilation symbol SILVERLIGHT, and a Windows Phone 7 program defines both SILVERLIGHT and PHONE. (You can see these by selecting the Build tab on the project Properties page.) That means you can have code that looks something like this:

```
#if PHONE
  // Code for Windows Phone 7
#else
  // Code for regular Silverlight
#endif
```

Or, you can differentiate at run time by accessing the Environment.OSVersion object. If the Platform property is PlatformID.WinCE and the Version.Major property is 7 or greater, your code is running on a Windows Phone 7 device (or perhaps Windows Phone 8 or 9).

In theory, it's possible to define conditional sections of XAML files using the AlternateContent and Choice tags defined in the markup-compatibility (mc) namespace, but these tags don't seem to be supported in Silverlight.

But XAML can contain data bindings, and these bindings can reference different objects depending on the platform. XAML can also have StaticResource references that retrieve different objects for different platforms. It is this approach I used in the TextTransform program.

I created the TextTransform solution the same way I created the DragImage solution. The solution has three projects: TextTransform (Silverlight program), TextTransform.Web (Web project to host the Silverlight program) and TextTransform.Phone (Windows Phone 7).

In the Silverlight project, I then created a TextTransformer control that derives from UserControl. This control is shared between the Silverlight project and the Windows Phone 7 project. The TextTransformer control contains a hardcoded text string (the word "TEXT") surrounded by a Border with four Thumb controls at the corners. Moving a Thumb causes a non-affine transform to be applied to the Border and TextBlock. (It only works correctly if the quadrilateral formed by the Border has no concave corners.)

The TextTransformer.xaml file doesn't create a new template for the Thumb, but it does define a Style as shown in **Figure 4**.

Notice the references to ThumbSize and HalfThumbOffset. Although the TextBlock displaying the text gets the correct Foreground property through property inheritance, the Border must be explicitly colored with the same foreground color:

```
<Border Name="border"
        BorderBrush="{StaticResource ForegroundBrush}"
        BorderThickness="1">
```

Where are these resources defined? They're defined in App.xaml. The regular Silverlight project includes a Resources collection in its App.xaml file that contains the following:

```
<Application.Resources>
  <SolidColorBrush x:Key="BackgroundBrush" Color="White" />
  <SolidColorBrush x:Key="ForegroundBrush" Color="Black" />
  <system:Double x:Key="ThumbSize">36</system:Double>
  <system:Double x:Key="HalfThumbOffset">-18</system:Double>
</Application.Resources>
```

The App.xaml file for the Windows Phone 7 program references the predefined resources for the two brushes and defines larger ThumbSize and HalfThumbOffset values:

```
<Application.Resources>
  <SolidColorBrush x:Key="BackgroundBrush"
     Color="{StaticResource PhoneBackgroundColor}" />
  <SolidColorBrush x:Key="ForegroundBrush"
     Color="{StaticResource PhoneForegroundColor}" />
  <system:Double x:Key="ThumbSize">96</system:Double>
  <system:Double x:Key="HalfThumbOffset">-48</system:Double>
</Application.Resources>
```

Figure 5 shows the program running in the browser and **Figure 6** shows the program running on the Windows Phone 7 emulator. The emulator is displayed at 50 percent of full size to compensate for the higher pixel density on the phone.

These techniques suggest that sharing code between the desktop and phone has become a reality. If you want to delve a bit deeper into this subject, the Surface Toolkit for Windows Touch includes a SurfaceThumb control for WPF developers. This is just like the normal Thumb control, but it adds support for true multi-touch and events for when the thumb is flicked. For more information, see the Surface Toolkit for Windows Touch beta page at msdn.microsoft.com/library/ee957351. ■

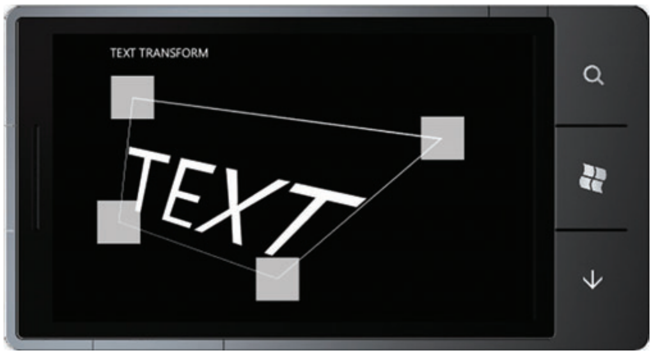


Figure 6 **The TextTransform Program on the Phone Emulator**

CHARLES PETZOLD *is a longtime contributing editor to* MSDN Magazine. *His new book, "Programming Windows Phone 7," is available as a free download at bit.ly/cpebookpdf.*

THANKS to the following technical experts for reviewing this article: *Doug Kramer and Robert Levy*

The Secret to a Successful Windows Phone 7 App

Killer applications sell the hardware and the OS that runs them. The classic example is Lotus 1-2-3, for which users bought the original IBM PC. The killer apps for dumb cell phones are voice and text messaging. They've been fantastically successful, with a mobile phone in the pocket of more than half the world's population.

The killer app for smartphones is more elusive. Worldwide smartphone sales in 2009 were 172 million units, about 14 percent of total phone sales of 1.2 billion units.

Microsoft has just released Windows Phone 7. Many reviewers dismiss it as too little, too late. But I think Microsoft's stolid, un-hip image (very different from 20 years ago) will play well to the much larger audience now considering the move to smartphones, provided that app developers recognize the composition of that audience and adjust their offerings to it.

Anyone who owns a smartphone today is, by definition, an early adopter. They bought an iPhone or Android because they enjoy the technology for its own sake, and for displaying status within their geek peer group. They consider the iPhone app store amazingly cool because it contains more than 100 apps that make fart noises, which they enjoy comparing and contrasting in bars with their friends.

Only male geeks can pacify themselves by sucking on the corner of a plastic phone.

The first app programmers resembled, and often were, their early adopter customers. They had only to build apps that they themselves liked in order to be successful. I'd bet that somewhere in the app is an Easter egg crediting the original expeller of the fart sounds and the brave anosmic souls that recorded them. But that's not the killer app that will catapult smartphones from early adopters into the mainstream.

The next wave of smartphone adoption will come from users who value technology not for itself, but only for making their lives easier. This wave is primarily controlled by women, either on their own or as telecommunication managers for their families. They have different technology-usage patterns and goals than male users, as I wrote in my August column, "Mars and Venus" (msdn.microsoft.com/magazine/ff898402). A killer app to them is very different from a killer app for the predominantly male early adopter audience.

My wife rolls her eyes at the farting apps. (My daughters, 10 and 7, think they're way cool, especially when networked with the companion lighter app on another iPhone. But they don't control the purse strings.) She absolutely loathes the iRevolver Russian Roulette app, and is underwhelmed by the iBeer drinking app—after working her job and schlepping the kids around all day, she needs the real thing. Only male geeks can pacify themselves by sucking on the corner of a plastic phone.

What is the overriding factor in the life of today's female smartphone purchaser? She's busy. She works a demanding job, then takes care of her kids, her pets, her parents and her in-laws, herself and her husband—very much in that order. She needs apps that deliver groceries because she doesn't have time to stop at the supermarket; apps that schedule appointments and track medical data with the pediatrician (or geriatrician or obstetrician or vet); apps that tell her where her kids are and how late her husband's train is running; apps that play soothing music while she waits for the kids' gymnastics practice to end, or drown out the caterwauling at their violin lessons. In a word: tools, not the geek toys that drove the early adopters.

The next wave of customers will demand completely different apps. Developers will succeed in satisfying these customers if and only if they follow Platt's First, Last and Only Law of User Experience Design: "Know Thy User, for He Is Not Thee." The cool app that so impressed the first wave will leave the much-larger second wave cold. ■

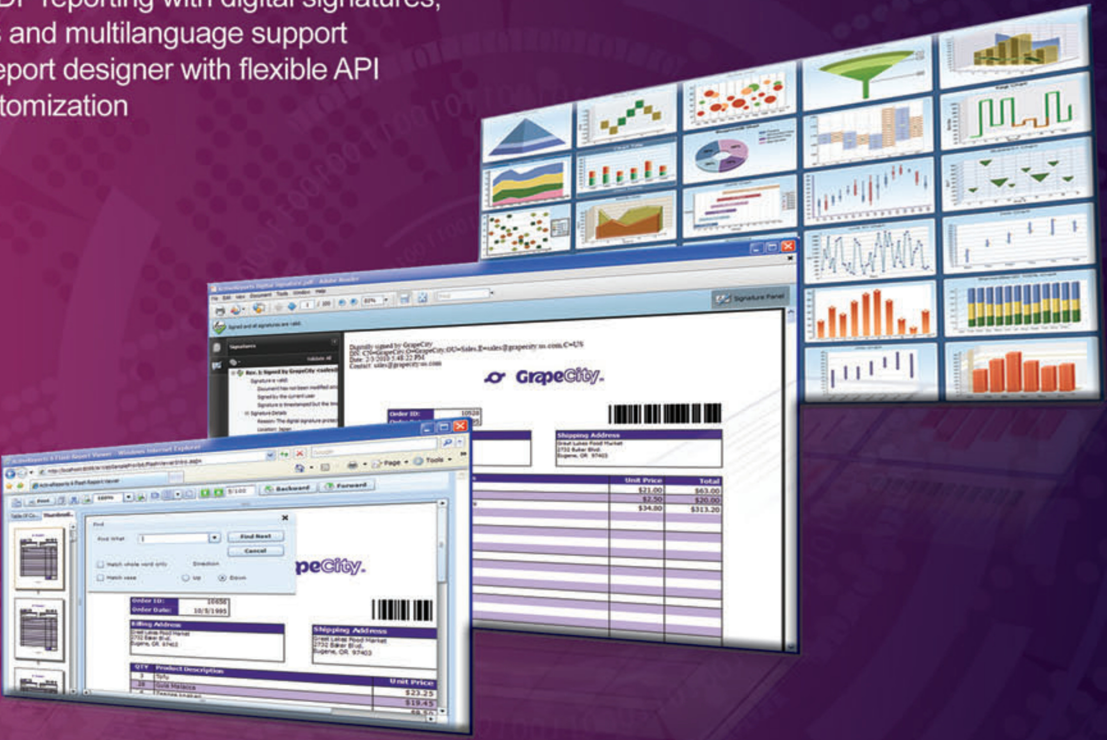
DAVID S. PLATT *teaches Programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

WE ARE

REPORTING

The defacto standard reporting tool for Microsoft Visual Studio.NET

- Unmatched customization, performance and quality
- Rich collection of report designing and rendering components
- Medium Trust environment support in ASP.NET
- Wide range of cross-platform export and preview formats
- Powerful PDF reporting with digital signatures, timestamps and multilanguage support
- End-user report designer with flexible API and full customization



ACTIVEREPORTS

Visual Studio Readers Merit 2009 · Readers' Choice 2009 asp.netPRO · Awards 2009-10 ComponentSource · Awards 2009-10 ComponentSource · Awards 2009-10 ComponentSource · SD Times 2010 100 · Members Choice Winner The Code Project 2010

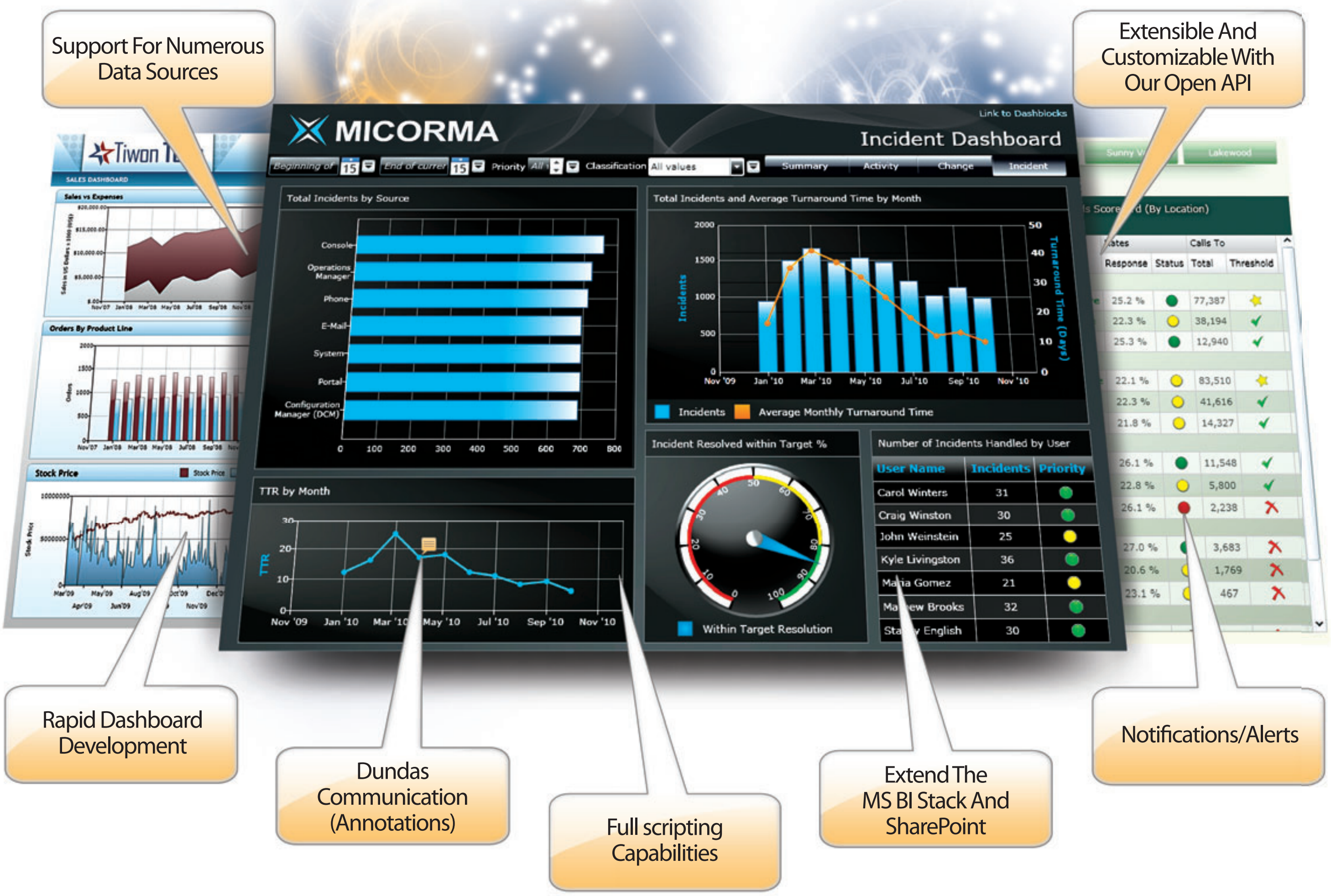
ACTIVEREPORTS SPREAD DATA DYNAMICS REPORTS ACTIVEANALYSIS

GrapeCity PowerTools
Report · Analyze · Excel

www.GCPowerTools.com

Dundas Dashboard

The Dashboard Framework for Developers like you


Support For Numerous Data Sources
Extensible And Customizable With Our Open API
Rapid Dashboard Development
Dundas Communication (Annotations)
Full scripting Capabilities
Extend The MS BI Stack And SharePoint
Notifications/Alerts
MICORMA
Incident Dashboard
Link to Dashblocks
Summary
Activity
Change
Incident
Total Incidents by Source
Total Incidents and Average Turnaround Time by Month
Incidents
Average Monthly Turnaround Time
Incident Resolved within Target %
Within Target Resolution
TTR by Month
Tiwon
Sales vs Expenses
Orders By Product Line
Stock Price

Number of Incidents Handled by User

User Name	Incidents	Priority
Carol Winters	31	🟢
Craig Winston	30	🟢
John Weinstein	25	🟡
Kyle Livingston	36	🟢
Ma ia Gomez	21	🟡
Ma ew Brooks	32	🟢
Sta y English	30	🟢

Response	Status	Total	Threshold
25.2 %	🟢	77,387	★
22.3 %	🟡	38,194	✓
25.3 %	🟢	12,940	✓
22.1 %	🟡	83,510	★
22.3 %	🟡	41,616	✓
21.8 %	🟡	14,327	✓
26.1 %	🟢	11,548	✓
22.8 %	🟡	5,800	✓
26.1 %	🔴	2,238	✗
27.0 %	🟢	3,683	✗
20.6 %	🟡	1,769	✗
23.1 %	🟡	467	✗

V2.5 Now Available

Dundas Dashboard was built with developers and IT staff in mind. Whether it's our open API, simple web integration or powerful scripting capabilities, technologists have all the tools and options they need for getting their dashboard projects up and running quickly and easily.

Powered by Microsoft® Silverlight®

Dundas Data Visualization

www.dundas.com
(416) 467-5100 • (800) 463-1492