



msdn[®] magazine

WINDOWS AZURE

Introducing the Windows Azure

AppFabric Caching Service

Karandeep Anand and Wade Wegner 26

CQRS on Windows Azure

Mark Seemann 36

Parsing Log Files with F#, MapReduce and Windows Azure

Noah Gift 44

PLUS:

Visual Studio TFS Team Project and Collection Guidance

Willy-Peter Schaub and Mike Schimmel 50

Use Bee Colony Algorithms to Solve Impossible Problems

James McCaffrey 56

Introduction to WebMatrix

Clark Sell 72

COLUMNS

TOOLBOX

F# Tools and Resources
Terrence Dorsey page 8

CUTTING EDGE

Give Your Classes
a Software Contract
Dino Esposito page 12

DATA POINTS

Composing WPF DataGrid
Column Templates for a
Better User Experience
Julie Lerman page 20

MOBILE MATTERS

Windows Phone Navigation,
Part 2: Advanced Recipes
Yochay Kiriathy and
Jaime Rodriguez page 82

UI FRONTIERS

Lissajous Animations in Silverlight
Charles Petzold page 88

DON'T GET ME STARTED

The Cat Butt Factor
David Platt page 96

DESIGN DEVELOP EXPERIENCE



Quince

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



NetAdvantage®

for Silverlight Data Visualization
for WPF Data Visualization

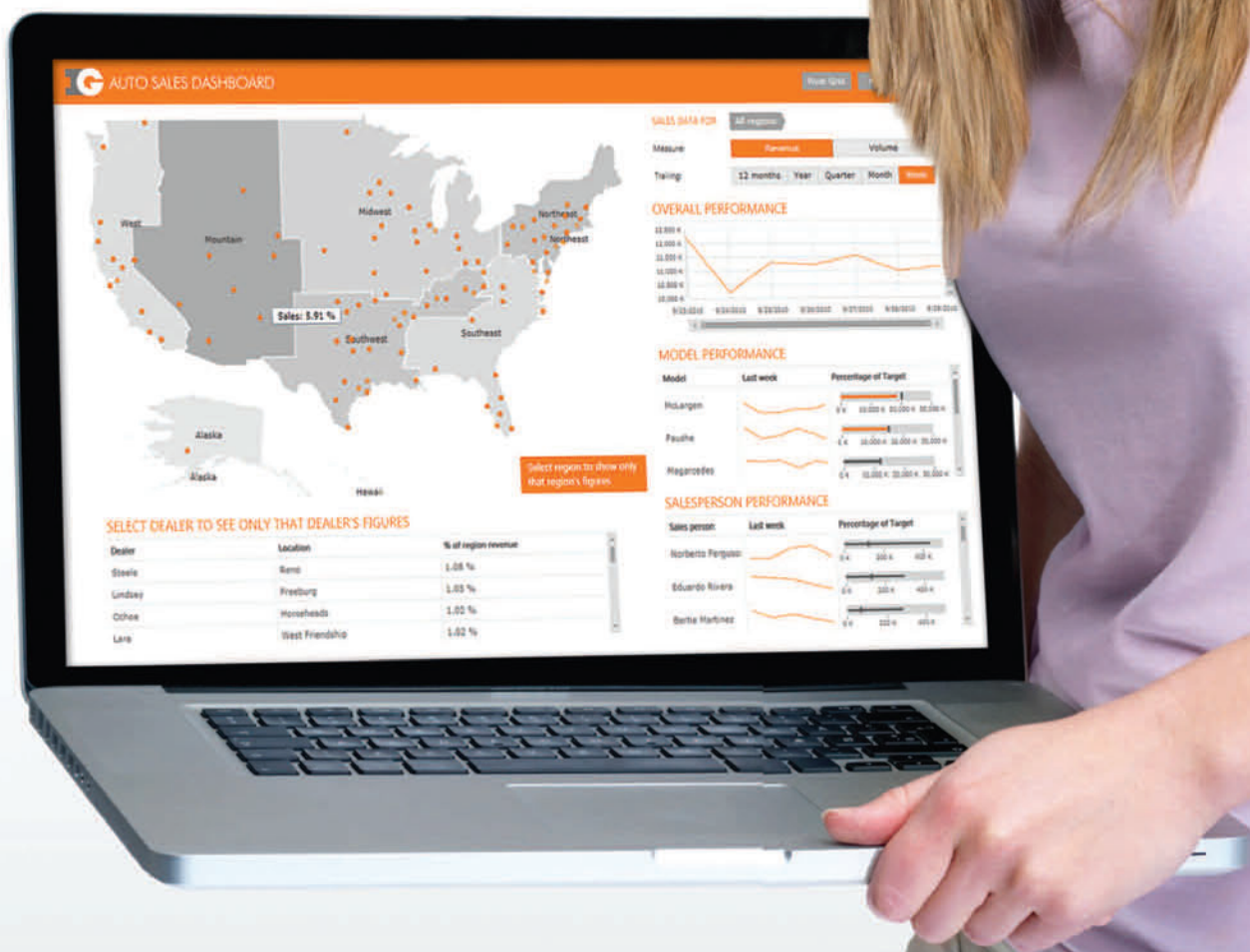
Then use NetAdvantage® UI controls, like the map control used here, to bring the application to life quickly & easily.



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](#)





The innovative mind thinks in Windows Azure.

Windows Azure is the cloud-based development platform that's creating a new mindset for developers. Build and run applications in the cloud. Launch applications in minutes instead of hours. Code in multiple languages and technologies, including .NET, PHP and Java. Start to innovate, unencumbered by redundancy, bandwidth or server configuration constraints. **That's Cloud Power.**

Start thinking in Windows Azure.

Try Windows Azure for free* at Microsoft.com/cloud/windowsazure



TO TRY WINDOWS AZURE,
SNAP OR TEXT MSDN
TO 70700.** Get the free
mobile app at <http://gettag.mobi>
**Standard messaging and data charges apply.



*Credit card required for trial offer. Usage in excess of 750 extra small compute hours per month is charged at normal rates. Free compute hours are available for the trial only until June 30, 2011 (11:59 p.m. UTC).



dtSearch®

Instantly Search Terabytes of Text

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

InfoWorld

"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 developer case studies, see www.dtSearch.com

Highlights hits in a wide range of data, using dtSearch's own file parsers and converters

- Supports MS Office through 2010 (Word, Excel, PowerPoint, Access), OpenOffice, ZIP, HTML, XML/XSL, PDF and more
- Supports Exchange, Outlook, Thunderbird and other popular email types, including nested and ZIP attachments
- Spider supports static and dynamic web data like ASP.NET, MS SharePoint, CMS, PHP, etc.
- API for SQL-type data, including BLOB data

25+ full-text & fielded data search options

- Federated searching
- Special forensics search options
- Advanced data classification objects

APIs for C++, Java and .NET through 4.x

- Native 64-bit and 32-bit Win / Linux APIs; .NET Spider API
- Content extraction only licenses available

Desktop with Spider

Web with Spider

Network with Spider

Engine for Win & .NET

Publish (portable media)

Engine for Linux

Ask about fully-functional evaluations!

The Smart Choice for Text Retrieval® since 1991

www.dtSearch.com • 1-800-IT-FINDS



msdn®

magazine

APRIL 2011 VOLUME 26 NUMBER 4

LUCINDA ROWLEY Director

KIT GEORGE 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

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

Redmond Media Group

Henry Allain President, Redmond Media Group

Matt Morollo Vice President, Publishing

Doug Barney Vice President, Editorial Director

Michele Imgrund Director, Marketing

Tracy Cook Online Marketing Director

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

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

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 XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

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

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

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

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

Reprints: For single article reprints (in minimum quantities of 250-500), e-reprints, 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



Printed in the USA

Your best source for software development tools!

programmer's paradise®



LEADTOOLS Document Imaging Suite SDK v17.0 by LEAD Technologies

- Libraries for C/C++, .NET, Silverlight, Windows Phone, WPF, WCF & WF
- High Performance OCR, ICR, MICR & OMR
- 1D & 2D Barcodes (Read/Write)
- Forms Recognition/Processing
- PDF, PDF/A and XPS
- Document Cleanup
- Advanced Compression (CCITT G3/G4, JBIG2, MRC, ABIC, ABC)
- High-Speed Scanning
- Print Capture and Document Writers

Paradise #
LO5 03301A02

\$4,018.99

programmers.com/lead



Embarcadero RAD Studio XE by Embarcadero

Embarcadero® RAD Studio XE is a comprehensive application development suite and the fastest way to visually build GUI-intensive, data-driven applications for Windows, .NET, PHP and the Web. RAD Studio includes Delphi®, C++Builder®, Delphi Prism™, and RadPHP™. The suite provides powerful compiled, managed and dynamic language support, heterogeneous database connectivity, rich visual component frameworks and a vast third-party ecosystem that enable you to deliver applications up to 5x faster across multiple Windows, Web, and database platforms!

NEW RELEASE!

Paradise #
CGI 15401A01

\$1,383.99

programmers.com/embarcadero



ActiveReports 6 by GrapeCity PowerTools

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

Professional Ed.
Paradise #
DO3 04301A01

\$1,310.99

programmers.com/grapecity



New Intel Visual Fortran Compiler by Intel

Intel® Visual Fortran Composer XE 2011 includes the latest generation of Intel® Fortran compilers, Intel® Visual Fortran Compiler XE 12.0 for Windows. Intel® Fortran Composer XE is available for Linux and Mac OS X. This package delivers advanced capabilities for development of application parallelism and winning performance for the full range of Intel® processor-based platforms and other compatible platforms. It includes the compiler's breadth of advanced optimization, multithreading, and processor support, as well as automatic processor dispatch, vectorization, and loop unrolling.

for Windows Single (SSR)
Paradise #
I23 86101E03

\$263.99

programmers.com/intel

UltraEdit

The #1 Best Selling Text Editor in the World

by IDM

UltraEdit is the world's standard in text editors. Millions use UltraEdit as the ideal text/hex/programmers editor on any platform — Windows, Mac, or Linux!

Features include syntax highlighting for nearly any programming language; powerful Find, Replace, Find in Files, and Replace in Files; FTP support, sort, column mode, hex, macros/scripting, large file handling (4+ GB), projects, templates, Unicode, and more.



Named User
1-24 Users
Paradise #
I84 01201A01

\$59.95

programmers.com/idm

Enterprise Architect Corporate Edition

Visualize, Document and Control Your Software Project

by Sparx Systems

Enterprise Architect is a comprehensive, integrated UML 2.1 modeling suite providing key benefits at each stage of system development. Enterprise Architect 7.5 supports UML, SysML, BPMN and other open standards to analyze, design, test and construct reliable, well understood systems. Additional plug-ins are also available for Zachman Framework, MODAF, DoDAF and TOGAF, and to integrate with Eclipse and Visual Studio 2005/2008.



1-4 Licenses
Paradise #
SP6 03101A02

\$182.99

programmers.com/sparxsystems

Mindjet® MindManager version 9 for Windows®

Every Successful Project Starts with a Good Plan.

by Mindjet®

Mindjet MindManager® is information mapping software that gives business professionals a better way to conquer information overload, brainstorm concepts, develop strategies, simplify project planning, and communicate results. MindManager® maps provide an intuitive visual framework for planning successful projects.



1 User
Paradise #
F15 17401A01

\$293.98

programmers.com/mindjet

Microsoft SQL Server Developer Edition 2008 R2 by Microsoft

SQL Server 2008 Developer enables developers to build and test applications that run on SQL Server on 32-bit, ia64, and x64 platforms. SQL Server 2008 Developer includes all of the functionality of Enterprise Edition, but is licensed only for development, test, and demo use. The license for SQL Server 2008 Developer entitles one developer to use the software on as many systems as necessary. For rapid deployment into production, instances of SQL Server 2008 Developer can easily be upgraded to SQL Server 2008 Enterprise without reinstallation.



2-bit/x64
IA64 DVD
Paradise #
M47 31101A04

\$40.99

programmers.com/microsoft

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

TX Text Control 16.0

Word Processing Components

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

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

New Version Released!



Professional Edition
Paradise #
T79 12101A01

\$1,109.99

Download a demo today.

programmers.com/textcontrol

TuneUp Utilities™ 2011 by TuneUp Distribution

The latest version of TuneUp Utilities™ has been designed with the need of small-to-medium businesses as well as government and educational institutions in mind. TuneUp Utilities™ will allow you to reduce costly system downtime, extend system lifetime and allow your staff to work more efficiently.

With more than 30 intuitive tools, you will benefit from:

- Increased performance and system response time
- Solutions to a large number of PC problems
- Greater system reliability
- Optimal maintenance
- Recovered disk space



Single User
Paradise #
TU7 01201A01

\$34.52

programmers.com/tuneup

Win an iPad!

Place an Order for Software (or Hardware) with Programmer's Paradise and You'll be Entered for a Drawing to Win an iPad Wi-Fi 32GB.



Just Use the **Offer Code TRWD04** When You Place Your Order Online or with Your Programmer's Paradise Representative.

866-719-1528

Prices subject to change. Not responsible for typographical errors.

programmersparadise.com



Cancel This!

Do you develop Web sites that actively push users away? “Of course not!” you respond. “What kind of addlepated fool would do that?”

Maybe, uh, you—if you do one or more of these things, either at the behest of your boss(es), or because you think it will help drive sales.

The thing that drove me away from a sports-related Web site last week was the simple act of trying to cancel my “premium content” subscription. So I logged in to my account and looked up Settings. I looked under my Subscriptions. I found my subscription.

Then I looked around for how to cancel it. Nada. There was a veritable buffet of tabs: Profiles, Messages, Groups, Blogs and more. I decided to click on Edit Profile. Sensible enough, right? My Profile should include information about my subscription, and a way to cancel.

El zippo—nothing useful there.

OK, where to next? Hmm, let's try Member Services. Surprise—more links! They included yet another Profile section, among other goodies. I didn't have any luck in the previous Profile section, so maybe this one had some deeper level of information, including how to cancel my account.

Nope. Disappointed yet again.

Aha! Account Information. That had to be it, especially as one of the categories was Payment Methods. It did have lots of account information, including Payment Methods, which listed some old credit cards and a new one. Yikes—I didn't know that old information was still there, so I deactivated all my old cards.

I figured if I deactivated all my cards, my subscription would be automatically cancelled. But no: I couldn't deactivate the current credit card. Because, of course, I needed some way to pay for the subscription I no longer wanted.

So, feeling a bit like Theseus, I wandered further into the labyrinth. I did searches on “cancel my subscription” and other phrases.

If you're guessing that I didn't find what I'm looking for, give yourself a one-handed clap.

Finally, I located a customer service number. Note that I didn't find any number, link or written description of any kind that mentioned canceling a subscription or account; I just stumbled across a phone number.

I called. After a bit of shuffling around, I talked to someone who cancelled my subscription. Thinking that this was ridiculous, I asked

if I missed something on the Web site—was there a way to cancel my subscription online that I didn't see? Was I just dumb (always a possibility with me)?

“No,” said the friendly voice on the other side. “There's no way to cancel your subscription on the Web site.”

I thought I must have misheard. So, you can sign up on the site—in fact, signing up and providing a credit-card number is as easy as taking a bite of cheesecake—but you can't cancel in the same way?

Feeling a bit like Theseus,
I wandered further into the labyrinth.

That's right, I was told. Then she tried to sell me on an even cheaper version of my subscription. How thoughtful of her.

To her credit, the lady was very professional and polite. So I responded, just as politely (I hope), to pass along my complaint about this sneaky, money-grubbing Web site and the company that produces it. She said she would.

And maybe she even did. Of course, I'm sure my complaint was placed in the “round file” equivalent of their e-mail system.

So, after a long while of clicking around in a vain effort to find a way to delete my subscription, I called a hard-to-locate phone number, hoping that there might be a way to cancel. I lucked into the right number, and canceled. But, again, *nowhere on the Web site* was I told how to cancel my subscription. Nary a word.

Although I'm certain this company would give some long-winded rationale for this Web site design, there is, of course, just one reason for it: They hope you get tired of looking, and give up the idea of canceling because it's not worth it. Throw up enough barriers, and hope the poor schlub gets tired of scaling or going around them.

Word of advice: Don't do this to your users. Don't treat them like suckers. Don't try to bleed them in this cynical manner. All you'll do is lose a customer forever.

The Golden Rule comes to mind.

Keith Ward

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

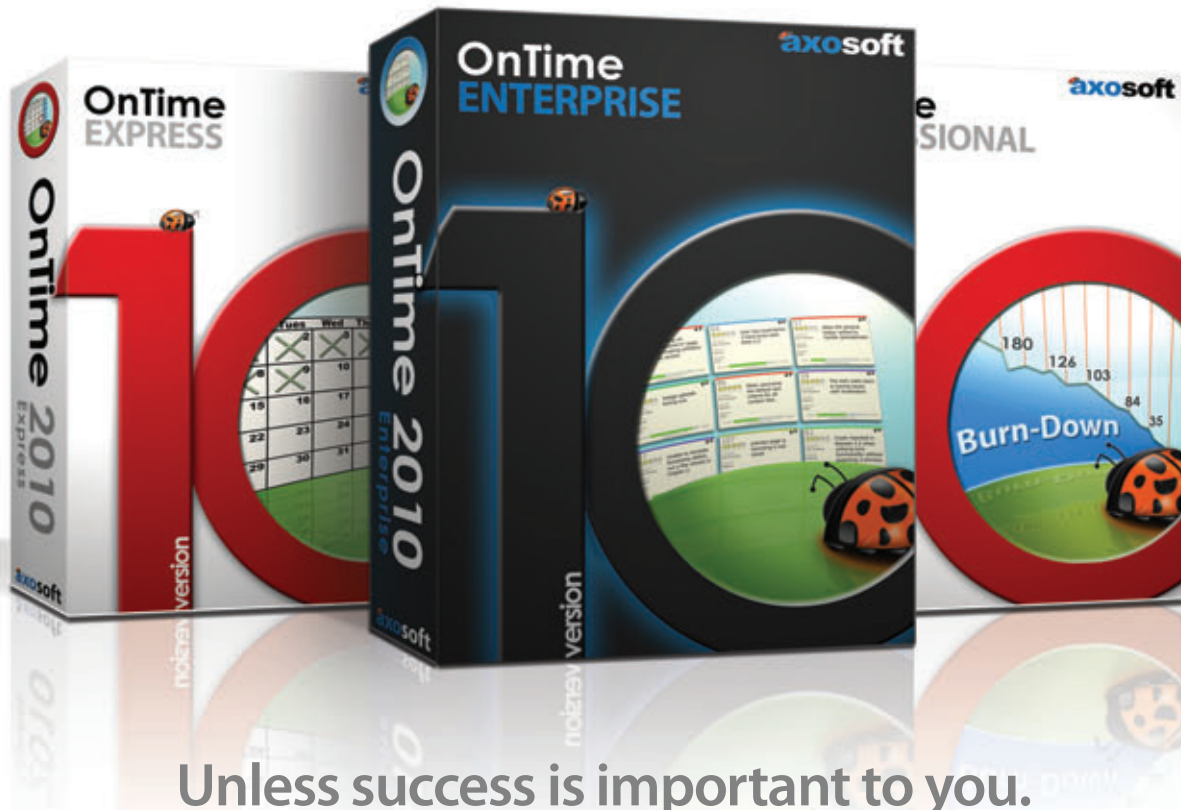
© 2011 Microsoft Corporation. All rights reserved.

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

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

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

It's not really that helpful...



Unless success is important to you.

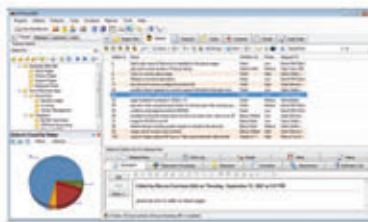
Project management and bug tracking for Agile & Scrum dev teams.

Hosted



OnTime Now! Managing an Agile or Scrum dev team using a cloud-based solution has never been easier or more cost effective. Start a Free 30-Day Trial in seconds and see why Axosoft is a leading provider.

Windows



OnTime for Windows is installed in your own server environment. It uses a .NET & SQL back-end, integrates easily with Visual Studio, and is a total joy to use in the rich Windows client. Includes free SDK / APIs.

Web



OnTime Web provides you with an anywhere-anytime solution, whether you go with OnTime hosted or installed. This is what a web app is supposed to be like -- loaded with features and fully capable.

Free 1-user License • Free Download • Free Team Trial • Free Web Demo

Some of our awards:



 **axosoft.com**
SOFTWARE FOR SOFTWARE DEVELOPMENT™
800.653.0024



F# Tools and Resources

F# is a new functional, type-safe programming language for the Microsoft .NET Framework. Developed at Microsoft Research, it's based on Objective Caml, or OCaml, which is in turn based on ML. The creator of F#, Don Syme, calls it "a synthesis between type-safe, scalable, math-oriented scripting and programming for .NET."

It may be a relatively young language, but F# has quickly developed a cult following within the programming community. Perhaps that's because F# lets you build on familiarity with the .NET Framework and Visual Studio. Perhaps it's because F# combines functional, imperative and object-oriented programming techniques in a single language. Or perhaps it's because F# is a strongly typed language that also supports the flexibility of compile-time typing.

Whichever aspects of the language appeal to you, here's a guide to the tools and resources that will help you get the most out of F#.

Getting Started

If you're using Visual Studio 2010 and the .NET Framework 4, you've already installed all the tools you need to start playing with F# 2.0. You can also download a standalone installer for the **F# 2.0 tools** (bit.ly/fGVQvI) if you want to use F# in a previous version of Visual Studio or in other development environments.

Your next stop for learning about F# should be the **Microsoft F# Developer Center** (msdn.microsoft.com/fsharp). Here you'll find documentation, links to examples, blog posts from F# experts and more.

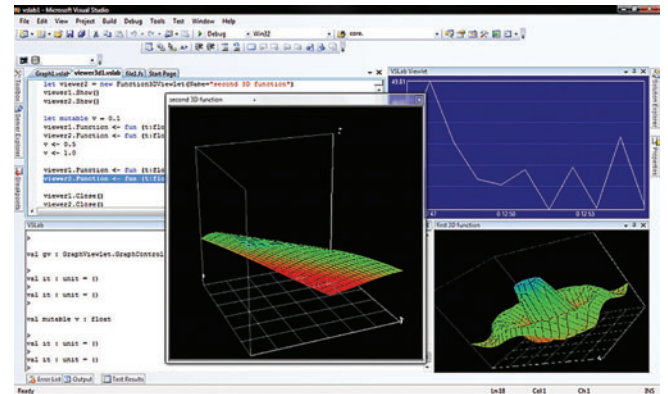
Your core resource is going to be the **F# Language Reference** (msdn.microsoft.com/library/dd233181) in the MSDN Library. This has all the language details, along with sample code.

Development Environments

As mentioned earlier, Visual Studio 2010 has deep support for F# right out of the box. But if you want to experiment with the language on other machines, or even other OSes, head over to the **Try F#** Web site (tryfs.net), where you can interactively code using F# in a browser-based interpreter.

Those without Visual Studio can grab a trial download as a standalone app or virtual machine from Microsoft (msdn.microsoft.com/vstudio/bb984878). Aside from Visual Studio, there are a number of other free and commercial IDEs that directly support F# development. These include **SharpDevelop** (sharpdevelop.net/OpenSource/SD), **xacc.ide** (xacc.wordpress.com) and **MonoDevelop** (monodevelop.com).

A unique aspect of MonoDevelop is that it enables you to set up your development environment on Windows, Mac OS X or Linux and target .NET Framework-based applications to those platforms in addition to Android, iOS and Windows Phone 7. **Functional Variations** has details on getting the F# features running; see "Installing and using F# in MonoDevelop" (functional-variations.net/monodevelop). **Robert Pickering** also has a guide to using F# on Mac OS X and Linux with Mono (strangelights.com/fsharp/MonoLinux.aspx) on his F# Resources blog.



VSLab Math Visualizations in Visual Studio

And if you're a die-hard Emacs user, don't despair: **Laurent Le Brun** is working on F# IntelliSense in Emacs (bit.ly/f3pd8b). Watch his blog for details.

Tools, Templates and Libraries

There are a variety of tools available to make F# easier and more powerful. Your first download should probably be the **F# PowerPack** (fsharp.powerpack.codeplex.com) from the F# development team. The PowerPack includes source for the F# compiler and code library, plus a number of additional tools and libraries. Among them are a Matrix library, tools for lexing and parsing, LINQ-based data access and tools for documenting F# libraries.

FAKE (github.com/forki/FAKE) is a build-automation system for F# designed with inspiration from tools like make and rake. FAKE lets you leverage both functional programming techniques and direct access to .NET assemblies while also integrating directly with the editing and debugging features of F#-aware IDEs like Visual Studio and SharpDevelop.

Head over to CodePlex to find a couple of handy testing tools:

TickSpec (tickspec.codeplex.com) is a Behavior-Driven Development framework that lets you describe behaviors in Gherkin (given, when, then), and then execute the behaviors against C# and F# methods.

FsUnit (fsunit.codeplex.com) is a unit-testing library. Simply drop the FsUnit.fs file into your project, then start writing test fixtures and tests.

And **FsCheck** (fscheck.codeplex.com) is a randomized testing framework based on the Haskell QuickCheck project that lets you write your test specifications in F#, C# or Visual Basic.

If you're interested in Windows Phone 7 development, go to the Visual Studio Gallery and grab the **F# Library for Windows Phone (XNA) template** (bit.ly/h5sg9h) and **F# and C# Win Phone App (Silverlight) template** (bit.ly/fraF4S) for Visual Studio.

Speaking of Visual Studio again, here are two great projects that will make working with F# much easier: **F# Project Extender**

DESIGN

Design Applications That Help Run the Business



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
t@infragistics

(fsprojectextender.codeplex.com) helps organize the files in F# projects in Solution Explorer without affecting compilation order; **F# Refactor** (fsharprefactor.codeplex.com) is developing a toolset for refactoring your F# code (this one's not quite ready for production work yet, and I'm sure they'd appreciate your contribution; give something back to the community that made all the great tools listed here).

The Web in F#

Websharper (websharper.com) is a framework for writing Web applications in F#. WebSharper 2.1 beta 5 brings F# compatibility extensions for a number of popular Web libraries including Google Maps, Google Visualization, jQuery, Bing Maps, Modernizr, InfoVis and Protovis.

Among his many contributions to the F# community, Tomas Petricek developed **F# Web Tools** (tomasp.net/projects/fsharpwebtools.aspx), which helps you author client/server/database Web apps with F# and LINQ. Be sure to watch Petricek's blog (tomasp.net/blog) for F# news, tips and tricks. Petricek was a member of the Microsoft Research team that created F# and thus has a unique insight into the language.

Remember we said "contributions"? Well, Petricek also created **F# Snippets** (fssnip.net), a site where you can share and borrow F# code snippets for hundreds of different uses. The site features full syntax highlighting and type information.

Fun Stuff

It's not all serious stuff in the world of F#. Sometimes you build something just to have fun. I think **IronJS** (github.com/fholm/IronJS) falls into that bucket. IronJS is a fast implementation of JavaScript running on the DLR and written in F#. Check out the code and play with it.

A bit more useful for math enthusiasts and number crunchers, **VSLab** (vslab.codeplex.com) provides an F#-based interactive visualization environment similar to Matlab within Visual Studio.

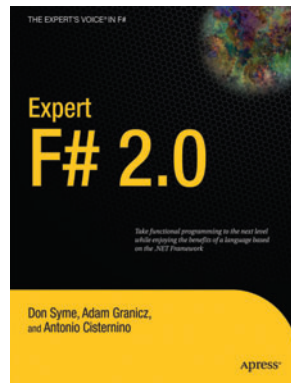
To help you learn F#, Chris Marinos started an ongoing project called **F# Koans** (bit.ly/hovkxs). Inspired by the Ruby Koans approach to learning the Ruby language, F# Koans teaches you the fundamentals and deeper truths of F# through testing. Each koan solution presents a runtime error. Your goal is to make the error go away. By fixing the errors you gain knowledge of the F# language and functional programming. Check out Marinos' blog for more F# tidbits at chrismarinos.com.

Reading Is Fundamental

Many of us rely on good old-fashioned books—and increasingly, ebooks—for learning new skills and as resources we refer to for language details and for implementation techniques. Here are a few you might turn to when adding F# to your quiver of skills.

"Expert F# 2.0" (Apress, 2010) (tinyurl.com/4ddksgm), by Don Syme, Adam Granicz and Antonio Cisternino, is an authoritative guide to the language by its inventor and two leading members of the F# development community. You'll learn how to use F# with .NET technologies including ASP.NET, LINQ, Windows Presentation Foundation and Silverlight. You can get "Expert F# 2.0" in print and ebook formats, including a Kindle edition.

Another reference for developers is **"Programming F#"** (O'Reilly, 2009) (tinyurl.com/4dhl2z9), by Chris Smith, who worked on the F#



Expert F# 2.0

development team at Microsoft. Smith walks you through using F# to solve problems with functional, imperative and object-oriented programming techniques so you can see the full capabilities of the language. There's also an overview of the F# libraries. "Programming F#" is available in print and a variety of ebook formats, including a Kindle edition.

Flying Frog Consultancy (ffconsultancy.com) specializes in functional programming and technical computing. The company offers a number of interesting publications on F#, including **"Visual F# 2010 for Technical Computing," "F# for Numerics," "F# for Visualization"** and the monthly **"F# Journal."** It also has a selection of other books on OCaml and data analysis techniques.

Deeply geeky stuff! Don't forget to check out its **F# News blog** (fsharpnews.blogspot.com) for news, analysis, examples and job listings for developers with experience in F# and functional programming.

Community Resources

From the outside it might seem as if F# is a niche technology, but there's a huge, enthusiastic community growing around it. To join in, start by hanging around some of these popular sites on the Web:

Community for F# (communityforfsharp.net) meets via Live Meeting about once every month for presentations from members around the world. Past meetings are archived on the site.

F# Central (fsharpcentral.com) provides a roundup of weekly F# community news. If you want to keep up with the latest articles, releases, training and job opportunities, point your browser or RSS reader right here.

hubFS: THE place for F# (cs.hubfs.net) hosts a news feed, blogs and discussion forums for F# programmers.

In addition, here are some key people in the F# community whose blogs you may want to peruse for tips and tricks:

Gordon Hogenson (blogs.msdn.com/gordonhogenson) is the technical writer who wrote most of the MSDN documentation for F#.

Luke Hoban (blogs.msdn.com/lukeh) was a project manager for the F# team. Although he's now working on the JavaScript team, he still loves F# and has some great information on his blog.

Richard Minerich (richardminerich.com) does a fantastic job of sharing blog posts and news about F# and other interesting topics. You can follow him on Twitter too, for the real-time F# experience (twitter.com/rickasaurus).

Dan Mohl (bloggemdano.blogspot.com) is an F# Insider and Microsoft MVP who blogs extensively about F# and Microsoft. Mohl has been keeping up on F#-related NuGet packages, so turn to him if you want the latest news.

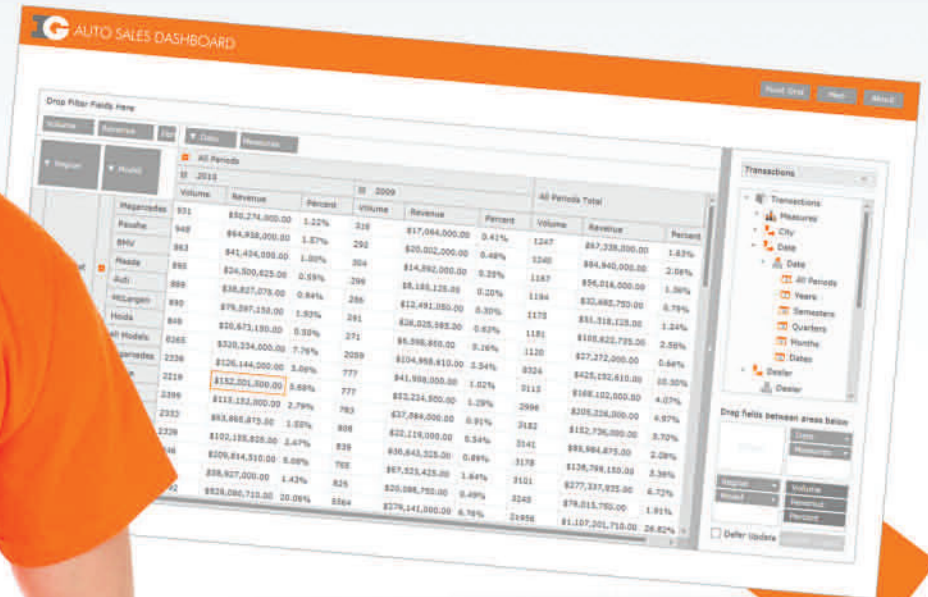
Don Syme (blogs.msdn.com/dsyme), as the creator of F#, uses his blog to share some great insights into how the language works and what you can expect in future releases. You'll also find breaking news about F# here. ■

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

THANKS to the following technical experts for reviewing this article:
Chris Marinos and Richard Minerich

DEVELOP

Rich Business Intelligence Applications in WPF and Silverlight



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
t@infragistics



Give Your Classes a Software Contract

An old but good practice of software development recommends that you place at the top of each method—before any significant behavior takes place—a barrier of conditional statements. Each conditional statement checks a different condition that input values must verify. If the condition isn't verified, the code throws an exception. This pattern is often referred to as If-Then-Throw.

But is If-Then-Throw all that we need to write effective and correct code? Is it sufficient in all cases?

The notion that it may *not* be sufficient in all cases isn't a new one. Design by Contract (DbC) is a methodology introduced several years ago by Bertrand Meyer based on the idea that each piece of software has a contract in which it formally describes what it expects and what it provides. The If-Then-Throw pattern nearly covers the first part of the contract; it lacks entirely the second part. DbC isn't natively supported in any mainstream programming language. However, frameworks exist to let you taste flavors of DbC in commonly used languages such as Java, Perl, Ruby, JavaScript and, of course, the Microsoft .NET Framework languages. In .NET, you do DbC via the Code Contracts library added to the .NET Framework 4, located in the mscorlib assembly. Note that the library is available to Silverlight 4 applications but not to Windows Phone applications.

I believe that nearly every developer would agree in principle that a contract-first approach to development is a great thing. But I don't

think so many are actively using Code Contracts in .NET 4 applications, now that Microsoft has made software contracts available and integrated in Visual Studio. This article focuses on the benefits of a contract-first approach for code maintainability and ease of development. You can hopefully use arguments in this article to sell Code Contracts to your boss for your next project. In future installments of this column, I'll drill down into aspects such as configuration, runtime tools and programming features such as inheritance.

Reasoning About a Simple Calculator Class

Code Contracts are a state of mind; you shouldn't put them aside until you're called to design a big application that requires a super architecture and employment of many cutting-edge technologies. Keep in mind that—when poorly managed—even the most powerful technology can cause problems. Code Contracts are useful for just about any type of application, as long as you have a good grasp of them. So let's start with a simple class—a classic Calculator class such as this:

```
public class Calculator
{
    public Int32 Sum(Int32 x, Int32 y)
    {
        return x + y;
    }

    public Int32 Divide(Int32 x, Int32 y)
    {
        return x / y;
    }
}
```

You'll probably agree that the code here isn't realistic, as it lacks at least one important piece: a check to see if you're attempting to divide by zero. As we write a better version of it, let's also assume that we have an additional issue to deal with: The calculator doesn't support negative values. **Figure 1** has an updated version of the code that adds a few If-Then-Throw statements.

So far, we can state that our class either starts processing its input data or, in the case of invalid input, it just throws before doing anything. What about the results generated by the class? What facts do we know about them? Looking at the specifications, we should expect that both methods return a value not less than zero. How can we enforce that and fail if it doesn't happen? We need a third version of the code, as shown in **Figure 2**.

Both methods now are articulated in three distinct phases: check of the input values, performance of the operation and check of the output. Checks on input and output serve two different purposes. Input checks flag bugs in your caller's code. Output checks look for bugs in your own code. Do you really need checks on the out-

Figure 1 The Calculator Class Implementing the If-Then-Throw Pattern

```
public class Calculator
{
    public Int32 Sum(Int32 x, Int32 y)
    {
        // Check input values
        if (x < 0 || y < 0)
            throw new ArgumentException();

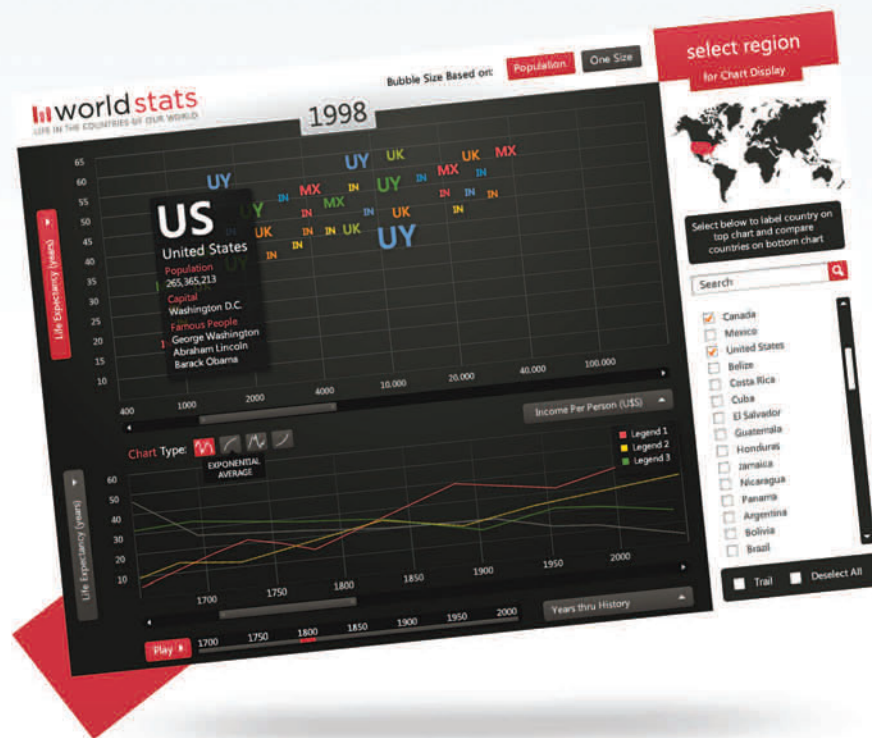
        // Perform the operation
        return x + y;
    }

    public Int32 Divide(Int32 x, Int32 y)
    {
        // Check input values
        if (x < 0 || y < 0)
            throw new ArgumentException();
        if (y == 0)
            throw new ArgumentException();

        // Perform the operation
        return x / y;
    }
}
```

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/motion 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
t@infragistics

Figure 2 The Calculator Class Checking Preconditions and Postconditions

```
public class Calculator
{
    public Int32 Sum(Int32 x, Int32 y)
    {
        // Check input values
        if (x < 0 || y < 0)
            throw new ArgumentException();

        // Perform the operation
        Int32 result = x + y;

        // Check output
        if (result < 0)
            throw new ArgumentException();

        return result;
    }

    public Int32 Divide(Int32 x, Int32 y)
    {
        // Check input values
        if (x < 0 || y < 0)
            throw new ArgumentException();
        if (y == 0)
            throw new ArgumentException();

        // Perform the operation
        Int32 result = x / y;

        // Check output
        if (result < 0)
            throw new ArgumentException();

        return result;
    }
}
```

put? I admit that check conditions can be verified via assertions in some unit tests. In that case, you don't strictly need such checks buried in the runtime code. However, having checks in the code makes the class self-describing and makes it clear what it can and can't do—much like the terms of a contracted service.

If you compare the source code of **Figure 2** with the simple class we started with, you'll see that the source grew by quite a few lines—and this is a simple class with few requirements to meet. Let's take it one step further.

Code Contracts are useful
for just about any type of
application, as long as you have
a good grasp of them.

In **Figure 2**, the three steps we identified (check input, operation and check output) run sequentially. What if the performance of the operation is complex enough to accommodate additional exit points? What if some of these exit points refer to error situations where other results are expected? Things can really get complicated. To illustrate the point, however, it suffices that we add a shortcut exit to one of the methods, as shown in **Figure 3**.

In the sample code (and it *is* just an example), method Sum attempts a shortcut if the two values are equal—multiplying instead

of summing. The code used to check output values, however, must be replicated for each early exit path in the code.

The bottom line is that nobody can reasonably think of taking a contract-first approach to software development without some serious tooling, or at least a specific helper framework. Checking preliminary conditions is relatively easy and cheap to do; dealing manually with post-execution conditions makes the entire code base unwieldy and error prone. Not to mention a few other ancillary aspects of contracts that would make the source code of classes a real mess for developers, such as checking conditions when input parameters are collections and ensuring that the class is always in a known valid state whenever a method or a property is called.

Enter Code Contracts

In the .NET Framework 4, Code Contracts is a framework that provides a much more convenient syntax to express a class contract. In particular, Code Contracts support three types of contracts: preconditions, postconditions and invariants. Preconditions indicate the preliminary conditions that should be verified for a method to execute safely. Postconditions express the conditions that should be verified once the method has executed either correctly or because of a thrown exception. Finally, an invariant describes a condition that's always true during the lifetime of any class instance. More precisely, an invariant indicates a condition that must hold after every possible interaction between the class and a client—that is, after executing public members, including constructors. The conditions expressed as invariants aren't checked and subsequently may be temporarily violated after the invocation of a private member.

The Code Contracts API consists of a list of static methods defined on the class Contract. You use the Requires method to express preconditions and Ensures to express postconditions. **Figure 4** shows how to rewrite the Calculator class using Code Contracts.

A quick comparison of **Figure 3** and **Figure 4** shows the power of an effective API for implementing DbC. Method code is back to a

Figure 3 A Shortcut Exit Duplicates the Code for Postconditions

```
public Int32 Sum(Int32 x, Int32 y)
{
    // Check input values
    if (x < 0 || y < 0)
        throw new ArgumentException();

    // Shortcut exit
    if (x == y)
    {
        // Perform the operation
        var temp = x << 1; // Optimization for 2*x

        // Check output
        if (temp < 0)
            throw new ArgumentException();

        return temp;
    }

    // Perform the operation
    var result = x + y;

    // Check output
    if (result < 0)
        throw new ArgumentException();

    return result;
}
```




Bring your XML development projects to light with the complete set of tools from Altova®



Experience how the Altova MissionKit®, the integrated suite of XML, database, and data integration tools, can simplify even the most advanced XML development projects.

New in Version 2011:

- Instant chart generation for XML, XBRL, and databases
- Schema flattener & schema subset creation
- Report generation in MapForce via StyleVision integration
- Data streaming for ETL
- Ability to auto-generate ASPX Web applications
- Numerous enhancements for database, XML, and XBRL reporting

The Altova MissionKit includes multiple intelligent XML tools – now with cutting edge chart and report generation:

XMLSpy® – industry-leading XML editor

- Support for all XML-based technologies
- Graphical editing views, powerful debuggers, code generation, & more

MapForce® – graphical data mapping & ETL tool

- Drag-and-drop data conversion with code generation

- Support for XML, DBs, EDI, Excel® 2007+, XBRL, flat files & Web services

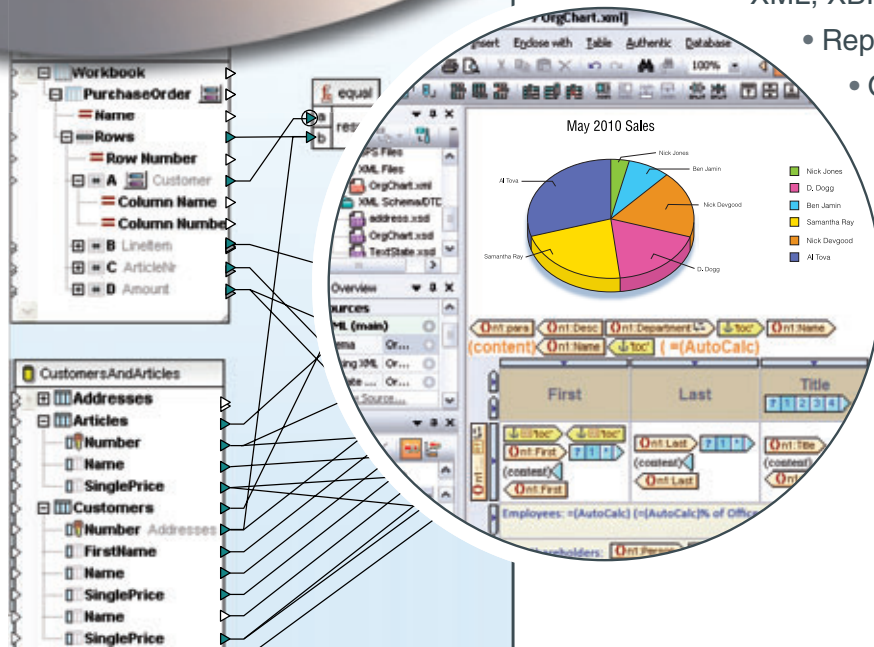
StyleVision® – visual stylesheet & report designer

- Graphical stylesheet and report design for XML, XBRL & databases

- Report designer with chart creation

- Output to HTML, PDF, Word & eForms

Plus up to five additional tools...



Download a 30 day free trial!

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

Figure 4 The Calculator Class Written Using Code Contracts

```
using System.Diagnostics.Contracts;
public class Calculator
{
    public Int32 Sum(Int32 x, Int32 y)
    {
        Contract.Requires<ArgumentOutOfRangeException>(x >= 0 && y >= 0);
        Contract.Ensures(Contract.Result<Int32>() >= 0);

        if (x == y)
            return 2 * x;

        return x + y;
    }

    public Int32 Divide(Int32 x, Int32 y)
    {
        Contract.Requires<ArgumentOutOfRangeException>(x >= 0 && y >= 0);
        Contract.Requires<ArgumentOutOfRangeException>(y > 0);
        Contract.Ensures(Contract.Result<Int32>() >= 0);

        return x / y;
    }
}
```

highly readable form in which you distinguish only two levels: contract information including both preconditions and postconditions and actual behavior. You don't have to mix conditions with behavior, as in **Figure 3**. As a result, readability is greatly improved, and maintaining this code gets much easier for the team. For example, you can quickly and safely add a new precondition or edit postconditions at will—you

Figure 5 Using Contract Abbreviators

```
public class Calculator
{
    public Int32 Sum(Int32 x, Int32 y)
    {
        // Check input values
        ValidateOperands(x, y);
        ValidateResult();

        // Perform the operation
        if (x == y)
            return x << 1;
        return x + y;
    }

    public Int32 Divide(Int32 x, Int32 y)
    {
        // Check input values
        ValidateOperandsForDivision(x, y);
        ValidateResult();

        // Perform the operation
        return x / y;
    }

    [ContractAbbreviator]
    private void ValidateOperands(Int32 x, Int32 y)
    {
        Contract.Requires<ArgumentOutOfRangeException>(x >= 0 && y >= 0);
    }

    [ContractAbbreviator]
    private void ValidateOperandsForDivision(Int32 x, Int32 y)
    {
        Contract.Requires<ArgumentOutOfRangeException>(x >= 0 && y >= 0);
        Contract.Requires<ArgumentOutOfRangeException>(y > 0);
    }

    [ContractAbbreviator]
    private void ValidateResult()
    {
        Contract.Ensures(Contract.Result<Int32>() >= 0);
    }
}
```

intervene in just one place and your changes can be clearly tracked.

Contract information is expressed via plain C# or Visual Basic code. Contract instructions aren't like classic declarative attributes, but they still maintain a strong declarative flavor. Using plain code instead of attributes increases the programming power of developers, as it makes it more natural to express the conditions you have in mind. At the same time, using Code Contracts gives you more guidance when you refactor the code. Code Contracts, in fact, indicate the behavior you should expect from the method. They help maintain coding discipline when you write methods and help keep your code readable even when preconditions and postconditions get numerous. Even though you can express contracts using a high-level syntax such as that in **Figure 4**, when code actually gets compiled, the resulting flow can't be much different from the code outlined in **Figure 3**. Where's the trick, then?

An additional tool integrated in the build process of Visual Studio—the Code Contracts rewriter—does the trick of reshaping the code, understanding the intended purpose of expressed preconditions and postconditions and expanding them into proper code blocks placed where they logically belong. As a developer, you just don't worry about where to place a postcondition and where to duplicate it if, at some point, you edit the code to add another exit point.

Expressing Conditions

You can figure out the exact syntax of preconditions and postconditions from the Code Contracts documentation; an up-to-date PDF can be obtained from the DevLabs site at bit.ly/f4LxHi. I'll briefly summarize it. You use the following method to indicate a required condition and otherwise throw the specified exception:

```
Contract.Requires<TException>(Boolean condition)
```

The method has a few overloads you might want to consider. The method `Ensures` expresses a postcondition:

```
Contract.Ensures(Boolean condition)
```

Code Contracts help you write clean code by forcing you to indicate expected behavior and results for each method.

When it comes to writing a precondition, the expression will usually contain only input parameters and perhaps some other method or property in the same class. If this is the case, you're required to decorate this method with the `Pure` attribute to note that executing the method won't alter the state of the object. Note that Code Contract tools assume property getters are pure.

When you write a postcondition, you may need to gain access to other information, such as the value being returned or the initial value of a local variable. You do this through ad hoc methods such as `Contract.Result<T>` to get the value (of type `T`) being returned from the method, and `Contract.OldValue<T>` to get the value stored in the specified local variable at the beginning of the method

execution. Finally, you also have a chance to verify a condition when an exception is thrown during the execution of the method. In this case, you use the method `Contract.EnsuresOnThrow<TException>`.

Abbreviators

The contract syntax is certainly more compact than using plain code, but it can grow large as well. When this happens, readability is at risk again. A natural remedy is grouping several contract instructions in a subroutine, as shown in **Figure 5**.

The `ContractAbbreviator` attribute instructs the rewriter on how to correctly interpret the decorated methods. Without the attribute to qualify it as a sort of macro to expand, in fact, the `ValidateResult` method (and other `ValidateXxx` methods in **Figure 5**) would contain rather inextricable code. What would, for example, `Contract.Result<T>` refer to, as it's used in a void method? Currently, the `ContractAbbreviator` attribute must be explicitly defined by the developer in the project, as it isn't included in the `mscorlib` assembly. The class is fairly simple:

```
namespace System.Diagnostics.Contracts
{
    [AttributeUsage(AttributeTargets.Method,
        AllowMultiple = false)]
    [Conditional("CONTRACTS_FULL")]
    internal sealed class
        ContractAbbreviatorAttribute :
            System.Attribute
    {
    }
}
```

Clean, Improved Code

Summing up, the Code Contracts API—basically the `Contract` class—is a native part of the .NET Framework 4, as it belongs to the `mscorlib` assembly. Visual Studio 2010 comes with a configuration page in the project properties specific to the configuration of Code Contracts. For each project, you must go there and explicitly enable runtime checking of contracts. You also need to download runtime tools from the DevLabs Web site. Once on the site, you pick up the proper installer for the version of Visual Studio you have. Runtime tools include the Code Contracts rewriter and interface generator, plus the static checker.

Code Contracts help you write clean code by forcing you to indicate expected behavior and results for each method. At the very minimum, this gives guidance when you go to refactor and improve your code. There's a lot more to be discussed about Code Contracts. In particular, in this article, I just quickly mentioned invariants and didn't mention the powerful feature contract

inheritance at all. In future articles, I plan to cover all of this and more. Stay tuned!

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

THANKS to the following technical expert for reviewing this article:
Brian Grunkemeyer

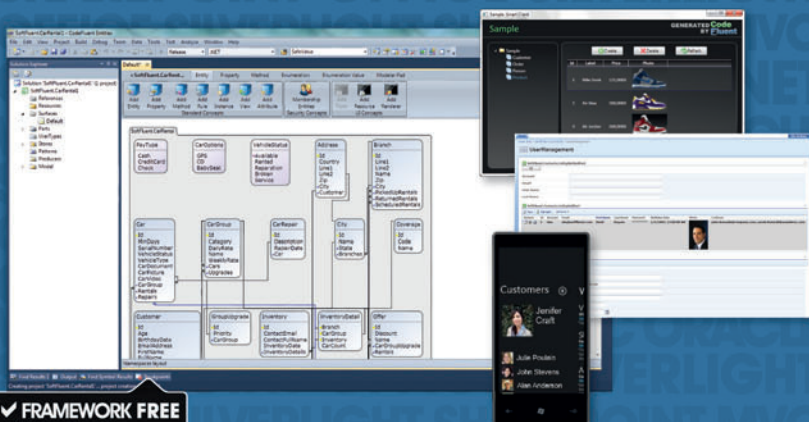
Less Plumbing Code, More Features use CODEFLUENT ENTITIES!

Focus on what makes the difference

Define your business logic, choose your technical targets, and create your custom business rules and behaviors!

Your application deserves rock-solid foundations: let CodeFluent Entities generate them, and keep yourself the fun part!

CodeFluent Entities provides a Visual Studio 2008/2010 integrated environment that helps you master present and future Microsoft .NET development technologies.



- ✓ FRAMEWORK FREE
- ✓ UML FREE
- ✓ ORM FREE
- ✓ TEMPLATE FREE

A model-first tool for continuous generation of all your application layers (user interface, service, business and database) that preserves your custom code.



**DOWNLOAD YOUR LICENSE
TOTALLY FREE FOR PERSONAL USAGE**

www.codefluententities.com/msdn



Contact: info@sofffluent.com
Twitter: twitter.com/sofffluent
US Sales: +1 425 372 3047
Europe Sales: +33 1 75 60 04 45

CodeFluent Entities is a trademark of SoftFluent SAS.
All other product and brand names are trademarks and/or registered trademarks of their respective holders

The Industry's Most Interactive Dashboards...



...are Now Available for Windows Phone 7!



ComponentArt **Data Visualization** for Windows Phone 7

Visit our website to learn about the industry's first complete framework for building mobile dashboards targeting Windows Phone 7. To experience the product in action, simply search for "ComponentArt" in the Windows Phone App Marketplace and install the free ComponentArt Demo application.

www.componentart.com

ComponentArt's latest Data Visualization technology allows you to present, navigate and visualize your data like never before. Built from the ground up to take advantage of the latest versions of Silverlight, WPF, Windows Phone and Visual Studio, ComponentArt Data Visualization for .NET delivers the complete feature set for building next-generation digital dashboards and data analysis applications.

ComponentArt
Build Something Amazing



Composing WPF DataGrid Column Templates for a Better User Experience

Recently I've been doing some work in Windows Presentation Foundation (WPF) for a client. Although I'm a big believer in using third-party tools, I sometimes avoid them in order to find out what challenges lay in wait for developers who, for one reason or another, stick to using only those tools that are part of the Visual Studio installation.

So I crossed my fingers and jumped into the WPF DataGrid. There were some user-experience issues that took me days to solve, even with the aid of Web searches and suggestions in online forums. Breaking my DataGrid columns into pairs of complementary templates turned out to play a big role in solving these problems. Because the solutions weren't obvious, I'll share them here.

The focus of this column will be working with the WPF ComboBox and DatePicker controls that are inside a WPF DataGrid.

The DatePicker and New DataGrid Rows

One challenge that caused me frustration was user interaction with the date columns in my DataGrid. I had created a DataGrid by dragging an object Data Source onto the WPF window. The designer's default behavior is to create a DatePicker for each DateTime value in the object. For example, here's the column created for a DateScheduled field:

```
<DataGridTemplateColumn x:Name=" dateScheduledColumn"
    Header="DateScheduled" Width="100">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <DatePicker
                SelectedDate="{Binding Path=DateScheduled, Mode=TwoWay,
                    ValidatesOnExceptions=true, NotifyOnValidationError=true}" />
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

This default isn't conducive to editing. Existing rows weren't updating when edited. The DatePicker doesn't trigger editing in the DataGrid, which means that the data-binding feature won't push the change through to the underlying object. Adding the UpdateSourceTrigger attribute to the Binding element and setting its value to PropertyChanged solved this particular problem:

```
<DatePicker
    SelectedDate="{Binding Path= DateScheduled, Mode=TwoWay,
        ValidatesOnExceptions=true, NotifyOnValidationError=true,
        UpdateSourceTrigger=PropertyChanged}" />
```

However, with new rows, there's a worse implication of the inability of the DatePicker to trigger the DataGrid edit mode. In a DataGrid, a new row is represented by a NewRowPlaceholder.

When you first edit a cell in a new row, the edit mode triggers an insert in the data source (again, not in the database, but in the underlying in-memory source). Because DatePicker isn't triggering edit mode, this doesn't happen.

I discovered this problem because, coincidentally, a date column was the first column in my row. I was depending on it to trigger the row's edit mode.

Figure 1 shows a new row where the date in the first editable column has been entered.

But after editing the value in the next column, the previous edit value has been lost, as you can see in **Figure 2**.

There were some
user-experience issues that
took me days to solve, even with
the aid of Web searches and
suggestions in online forums.

The key value in the first column has become 0 and the date that was just entered has changed to 1/1/0001. Editing the Task column finally triggered the DataGrid to add a new entity in the source. The ID value becomes an integer—default, 0—and the date value becomes the .NET default minimum date, 1/1/0001. If I had a default date specified for this class, the user's entered date would have changed to the class default rather than the .NET default. Notice that the date in the Date Performed column didn't change to its default. That's because DatePerformed is a nullable property.

So now the user has to go back and fix the Scheduled Date again? I'm sure the user won't be happy with that. I struggled with this problem for a while. I even changed the column to a DataText-BoxColumn instead, but then I had to deal with validation issues that the DatePicker had protected me from.

Finally, Varsha Mahadevan on the WPF team set me on the right path.

By leveraging the compositional nature of WPF, you can use two elements for the column. Not only does the DataGridTemplateColumn have a CellTemplate element, but there's a CellEditingTemplate as well. Rather than ask the DatePicker control to trigger edit mode, I use the DatePicker only when I'm already editing. For

Code download available at code.msdn.microsoft.com/mag201104DataPoints.

we are Countersoft you are Control



GEMINI Project Platform

The most versatile project management platform for software development and testing teams around the world

The award-winning Gemini Project Platform is the .NET based project platform that uniquely allows teams to work the way they want to work with more functionality and easy customization for optimum team performance and predictable results.

BOOK A DEMO / FREE TRIAL / 3-for-FREE
Seeing is believing! We'll talk about YOU.
www.geminiplatform.com



ATLAS Product Support Optimized

The first comprehensive solution to address ISVs' profitability AND customer experience by targeting product support

Patent pending Atlas addresses the area of the software business that matters most: the product support ecosystem. Atlas is the logical alternative to revenue-sapping, customer-irritating forums. Atlas offers smarter community-based support and integrates FAQ/KB, documentation and how-to videos.

DOWNLOAD NOW
and lead from the front.
www.atlasanswer.com



COUNTERSOFT

ENABLING COLLECTIVE CAPABILITY www.countersoft.com

Contact // Europe/Asia: +44 (0)1753 824000 // US/Canada: 800.927.5568 // E: sales@countersoft.com



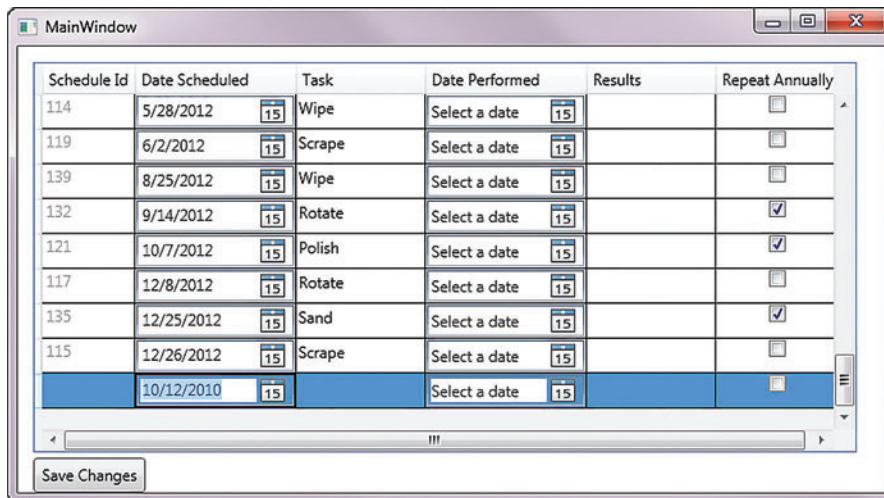


Figure 1 Entering a Date Value into a New Row Placeholder

displaying the date in the CellTemplate, I switched to a TextBlock. Here's the new XAML for dateScheduledColumn:

```
<DataGridTemplateColumn x:Name="dateScheduledColumn"
    Header="Date Scheduled" Width="125">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding Path= DateScheduled, StringFormat=\{0:d\}}" />
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
    <DataGridTemplateColumn.CellEditingTemplate>
        <DataTemplate>
            <DatePicker SelectedDate="{Binding Path=DateScheduled, Mode=TwoWay,
                ValidatesOnExceptions=true, NotifyOnValidationError=true}" />
        </DataTemplate>
    </DataGridTemplateColumn.CellEditingTemplate>
</DataGridTemplateColumn>
```

Notice that I no longer need to specify UpdateSourceTrigger. I've made the same changes to the DatePerformed column.

Now the date columns start out as simple text until you enter the cell and it switches to the DatePicker, as you can see in Figure 3.

In the rows above the new row, you don't have the DatePicker calendar icon.

But it's still not quite right. We're still getting the default .NET value as we begin editing the row. Now you can benefit from defining a default in the underlying class. I've modified the constructor of the ScheduleItem class to initialize new objects with today's date. If data is retrieved from the database, it will overwrite that default. In my project, I'm using the Entity Framework, therefore my classes are generated automatically. However, the generated classes are partial classes, which allow me to add the constructor in an additional partial class:

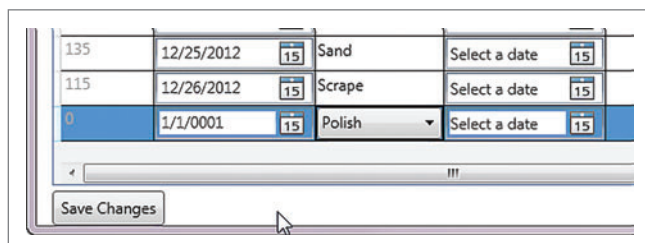


Figure 2 Date Value Is Lost After the Value of the Task Column in the New Row Is Modified

```
public partial class ScheduleItem
{
    public ScheduleItem()
    {
        DateScheduled = DateTime.Today;
    }
}
```

Now when I begin entering data into the new row placeholder by modifying the DateScheduled column, the DataGrid will create a new ScheduleItem for me and the default (today's date) will be displayed in the DatePicker control. As the user continues to edit the row, the value entered will remain in place this time.

Reducing User Clicks to Allow Editing

One downside to the two-part template is that you have to click on the cell twice to trigger the DatePicker. This is a frustration to anyone doing data entry, especially if they're used to using the keyboard to enter data without touching the mouse. Because the DatePicker is in the editing template, it won't get focus until you've triggered the edit mode—by default, that is. The design was geared for TextBoxes and with those it works just right. But it doesn't work as well with the DatePicker. You can use a combination of XAML and code to force the DatePicker to be ready for typing as soon as a user tabs into that cell.

One challenge that caused me frustration was user interaction with the date columns in my DataGrid.

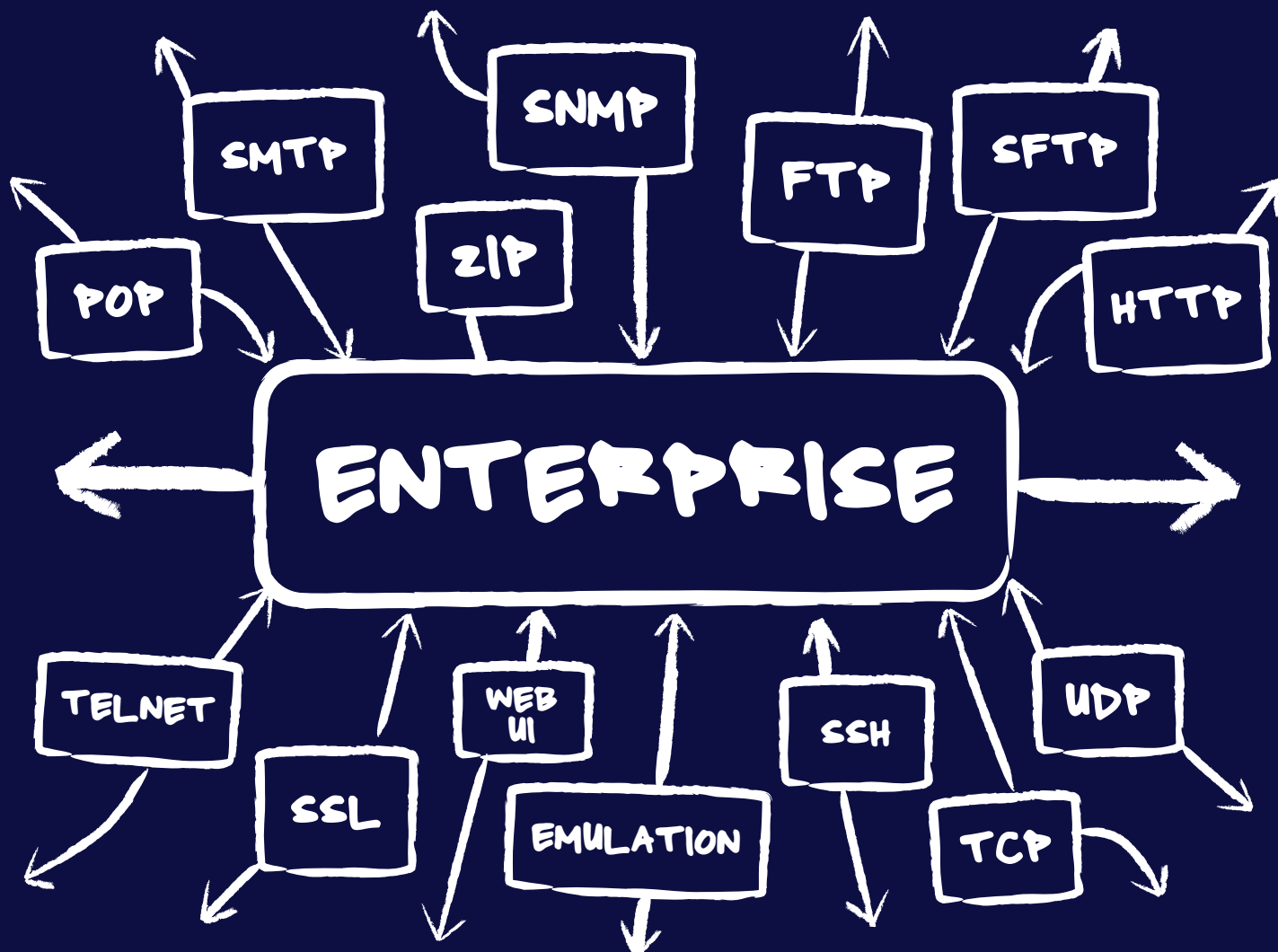
First you'll need to add a Grid container into the CellEditingTemplate so that it becomes a container of the DatePicker. Then, using the WPF FocusManager, you can force this Grid to be the focal point of the cell when the user enters the cell. Here's the new Grid element surrounding the DatePicker:

```
<Grid FocusManager.FocusedElement="{Binding ElementName= dateScheduledPicker}">
    <DatePicker x:Name=" dateScheduledPicker"
        SelectedDate="{Binding Path=DateScheduled, Mode=TwoWay,
            ValidatesOnExceptions=true, NotifyOnValidationError=true}" />
</Grid>
```

Notice I've provided a name for the DatePicker control and I'm pointing to that name using the FocusedElement Binding ElementName.

Moving your attention to the DataGrid that contains this DatePicker, notice that I've added three new properties (RowDetailsVisibilityMode, SelectionMode and SelectionUnit), as well as a new event handler (SelectedCellsChanged):

```
<DataGrid AutoGenerateColumns="False" EnableRowVirtualization="True"
    ItemsSource="{Binding}" Margin="12,12,22,31"
    Name="scheduleItemsDataGrid"
    RowDetailsVisibilityMode="VisibleWhenSelected"
    SelectionMode="Extended" SelectionUnit="Cell"
    SelectedCellsChanged="scheduleItemsDataGrid_SelectedCellsChanged">
```



Internet Connectivity for the Enterprise

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



PowerSNMP for ActiveX and .NET

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

PowerWEB for ASP.NET

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

PowerTCP for ActiveX and .NET

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

SSH
UDP
TCP
SSL

FTP
SFTP
HTTP
POP

SMTP
IMAP
S/MIME
Ping

DNS
Rlogin
Rsh
Rexec

Telnet
VT Emulation
ZIP Compression
more...

Ask us about Mono Platform support. Contact sales@dart.com.

Download a fully functional product trial today!

 **DART.com**

The changes to the DataGrid will enable notification when a user selects a new cell in the DataGrid. Finally, when this happens, you need to ensure that the DataGrid does indeed go into edit mode, which will then provide the user with the necessary cursor in the DatePicker. The `scheduleItemsDataGrid_SelectedCellsChanged` method will provide this last bit of logic:

```
private void scheduleItemsDataGrid_
SelectedCellsChanged
(object sender,
System.Windows.Controls.SelectedCellsChangedEventArgs e)
{
    if (e.AddedCells.Count == 0) return;
    var currentCell = e.AddedCells[0];
    string header = (string)currentCell.Column.Header;

    var currentCell = e.AddedCells[0];

    if (currentCell.Column ==
        scheduleItemsDataGrid.Columns[DateScheduledColumnIndex])
    {
        scheduleItemsDataGrid.BeginEdit();
    }
}
```

Note that in the class declarations, I've defined the constant, `DateScheduledColumnIndex` as 1, the position of the column in the grid.

It took a bit of poking around
to find the right combination
of XAML and code elements to
make the DatePicker work nicely
inside of a DataGrid.

With all of these changes in place, I now have happy end users. It took a bit of poking around to find the right combination of XAML and code elements to make the DatePicker work nicely inside of

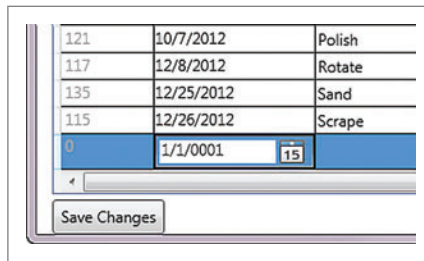


Figure 3 DateScheduled Column Using Both a TextBlock and a DatePicker

a DataGrid, and I hope to have helped you avoid making that same effort. The UI now works in a way that feels natural to the user.

Enabling a Restricted ComboBox to Display Legacy Values

Having grasped the value of layering the elements inside the `DataGridTemplateColumn`, I revisited another problem that I'd nearly given up on with a `DataGridComboBoxColumn`.

This particular application was being written to replace a legacy application with legacy data. The legacy application had allowed users to enter data without a lot of control. In the new application, the client requested that some of the data entry be restricted through the use of drop-down lists. The contents of the drop-down list were provided easily enough using a collection of strings. The challenge was that the legacy data still needed to be displayed even if it wasn't contained in the new restricted list.

My first attempt was to use the `DataGridComboBoxColumn`:

```
<DataGridComboBoxColumn x:Name="frequencyCombo"
    MinWidth="100" Header="Frequency"
    ItemsSource="{Binding Source={StaticResource frequencyViewSource}}"
    SelectedValueBinding=
        "{Binding Path=Frequency, UpdateSourceTrigger=PropertyChanged}"
/>
```

The source items are defined in codebehind:

```
private void PopulateTrueFrequencyList()
{
    _frequencyList =
        new List<String>{
            "Initial", "2 Weeks",
            "1 Month", "2 Months",
            "3 Months", "4 Months",
            "5 Months", "6 Months",
            "7 Months", "8 Months",
            "9 Months", "10 Months",
            "11 Months", "12 Months"
        };
}
```

This `_frequencyList` is bound to `frequencyViewSource.Source` in another method.

In the myriad possible configurations of the `DataGridComboBoxColumn`, I could find no way to display disparate values that may have already been stored in the `Frequency` field of the database table. I won't bother listing all of the solutions I attempted, including one that involved dynamically adding those extra values to the bottom of the `_frequencyList` and then removing them as needed. That was a solution I disliked but was afraid that I might have to live with.

I knew that the layered approach of WPF to composing a UI had to provide a mechanism for this, and having solved the DatePicker problem, I realized I could use a similar approach for the ComboBox. The first part of the trick is to avoid the slick `DataGridComboBoxColumn` and use the more classic approach of embedding a ComboBox inside of a `DataGridTemplateColumn`. Then, leveraging the compositional nature of WPF, you can use two elements for the column just as with the `DateScheduled` column. The first is a `TextBlock` to display values and the second is a `ComboBox` for editing purposes.

Figure 4 shows how I've used them together.

Figure 4 Combining a TextBlock to Display Values and a ComboBox for Editing

```
<DataGridTemplateColumn x:Name="taskColumnFaster"
    Header="Task" Width="100" >
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding Path=Task}" />
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>

    <DataGridTemplateColumn.CellEditingTemplate>
        <DataTemplate>
            <Grid FocusManager.FocusedElement=
                "{Binding ElementName= taskCombo}" >
                <ComboBox x:Name="taskCombo"
                    ItemsSource="{Binding Source={StaticResource taskViewSource}}"
                    SelectedItem="{Binding Path=Task}"
                    IsSynchronizedWithCurrentItem="False"/>
            </Grid>
        </DataTemplate>
    </DataGridTemplateColumn.CellEditingTemplate>
</DataGridTemplateColumn>
```

The TextBlock has no dependency on the restricted list, so it's able to display whatever value is stored in the database. However, when it's time to edit, the ComboBox will be used and entry is limited to the values in the frequencyViewSource.

Allowing Users to Edit the ComboBox When the Cell Gets Focused

Again, because the ComboBox won't be available until the user clicks twice in the cell, notice that I wrapped the ComboBox in a Grid to leverage the FocusManager.

I've modified the SelectedCellsChanged method in case the user starts his new row data entry by clicking the Task cell, not by moving to the first column. The only change is that the code also checks to see if the current cell is in the Task column:

```
private void scheduleItemsDataGrid_SelectedCellsChanged(object sender,
    System.Windows.Controls.SelectedCellsChangedEventArgs e)
{
    if (e.AddedCells.Count == 0) return;
    var currentCell = e.AddedCells[0];
    string header = (string)currentCell.Column.Header;

    if (currentCell.Column ==
        scheduleItemsDataGrid.Columns[DateScheduledColumnIndex]
        || currentCell.Column == scheduleItemsDataGrid.Columns[TaskColumnIndex])
    {
        scheduleItemsDataGrid.BeginEdit();
    }
}
```

The UI now works in a way that feels natural to the user.

Don't Neglect User Experience

While we developers are building solutions, it's common to focus on making sure data is valid, that it's getting where it needs to go and other concerns. We may not even notice that we had to click twice to edit a date. But your users will quickly let you know if the application you've written to help them get their jobs done more effectively is actually holding them back because they have to keep going back and forth from the mouse to the keyboard.

While the WPF data-binding features of Visual Studio 2010 are fantastic development time savers, fine-tuning the user experience for the complex data grid—especially when combining it with the equally complex DatePicker and ComboBoxes—will be greatly appreciated by your end users. Chances are, they won't even notice the extra thought you put in because it works the way they expect it—but that's part of the fun of our job. ■

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 at twitter.com/julielerman.

THANKS to the following technical expert for reviewing this article:
Varsha Mahadevan

msdnmagazine.com

EXTREME SCALABILITY

As a software architect, you know that delivering extreme scalability is the key to keeping up with the demands of your users. You can't let bottlenecks to data access or analysis get in your way. You need **ScaleOut StateServer's** distributed data grid to maximize your application's performance.

Its powerful architecture automatically scales in-memory data storage across multiple servers to combine blazing speed with industry-leading ease of use. Unique management tools and "map/reduce" parallel analysis give you everything you need to take your application's scalability to the next level.

Download your **FREE** trial copy of ScaleOut StateServer® today!



SCALEOUT SOFTWARE
Distributed Data Grids for the Enterprise

www.scaleoutsoftware.com/eval | 503.643.3422

Introducing the Windows Azure AppFabric Caching Service

Karandeep Anand and Wade Wegner

In a world where **speed and scale** are the key success metrics of any solution, they can't be afterthoughts or retrofitted to an application architecture—speed and scale need to be made the core design principles when you're still on the whiteboard architecting your application.

The Windows Azure AppFabric Caching service provides the necessary building blocks to simplify these challenges without having to learn about deploying and managing another tier in your application architecture. In a nutshell, the Caching service is the elastic memory that your application needs for increasing its performance and throughput by offloading the pressure from the data tier and the distributed state so that your application is able to easily scale out the compute tier.

The Caching service was released as a Community Technology Preview (CTP) at the Microsoft Professional Developers Conference in 2010, and was refreshed in February 2011.

This article discusses:

- Caching client and service
- Setting up AppFabric Caching
- Caching in your app
- Storing session data

Technologies discussed:

Windows Azure, Visual Studio 2010

The Caching service is an important piece of the Windows Azure platform, and builds upon the Platform as a Service (PaaS) offerings that already make up the platform. The Caching service is based on the same code-base as Windows Server AppFabric Caching, and consequently it has a symmetric developer experience to the on-premises cache.

The Caching service offers developers the following capabilities:

- Pre-built ASP.NET providers for session state and page output caching, enabling acceleration of Web applications without having to modify application code
- Caches any managed object—no object size limits, no serialization costs for local caching
- Easily integrates into existing applications
- Consistent development model across both Windows Azure AppFabric and Windows Server AppFabric
- Secured access and authorization provided by the Access Control service

While you can set up other caching technologies (such as memcached) on your own as instances in the cloud, you'd end up installing, configuring and managing the cache clusters and instances yourself. This defaults one of the main goals of the cloud, and PaaS in particular: to get away from managing these details. The Windows Server AppFabric Caching service removes this burden from you, while also accelerating the performance of ASP.NET Web applications running in Windows Azure with little or no code or configuration changes. We'll show you how this is done throughout the remainder of the article.

Under the Hood

Now that you understand how caching can be a strategic design choice for your application, let's dig a little deeper into what the Caching service really looks like. Let's take an example of a typical shopping Web site, say an e-commerce site that sells Xbox and PC games, and use that to understand the various pieces of the Caching service.

At a high level, there are three key pieces to the puzzle:

- Cache client
- Caching service
- Connection between the client and the service

Cache client is the proxy that lives within your application—the shopping Web site in this case. It's the piece of code that knows how to talk to the Caching service, in a language that both understand well. You need to include this client assembly in your application and deploy it with your Web application with the appropriate configuration to be able to discover and talk to the Caching service. (We'll cover this process later in the article.)

Because there are some common application patterns, such as ASP.NET session state, where caching can be used out of the box, there are two ways to use the client.

For explicit programming against the cache APIs, include the cache client assembly in your application from the SDK and you can start making GET/PUT calls to store and retrieve data from the cache. This is a good way to store your games catalog or other reference data for your gaming Web site.

For higher-level scenarios that in turn use the cache, you need to include the ASP.NET session state provider for the Caching service and interact with the session state APIs instead of interacting with the caching APIs. The session state provider does the heavy lifting of calling the appropriate caching APIs to maintain the session state in the cache tier. This is a good way for you to store information like user preferences, shopping cart, game-browsing history and so on in the session state without writing a single line of cache code.

You can pretty much keep any object in the cache: text, data, blobs, CLR objects and so on. There's no restriction on the size of the object, either. Hence, whether you're storing explicit objects in cache or storing session state, the object size is not a consideration to choose whether you can use the Caching service in your application.

One thing to note about the Caching service is that it's an explicit cache that you write to and have full control over. It's not a transparent cache layer on top of your database or storage. This has the benefit of providing full control over what data gets stored and managed in the cache, but also means you have to program against the cache as a separate data store using the cache APIs.

This pattern is typically referred to as the cache-aside, where you first load data into the cache and then check if it exists there for retrieving and, only when it's not available there, you explicitly read the data from the data tier. So, as a developer, you need to learn the cache programming model, the APIs, and common tips and tricks to make your usage of cache efficient.

One other thing to know about the cache client is the ability to cache a subset of the data that

resides in the distributed cache servers, directly on the client—the Web server running the gaming Web site in our example. This feature is popularly referred to as the *local cache*, and it's enabled with a simple configuration setting that allows you to specify the number of objects you wish to store and the timeout settings to invalidate the cache.

The second key part of the puzzle is the Caching service itself. Think of the Caching service as Microsoft running a large set of cache clusters for you, heavily optimized for performance, uptime, resiliency and scale out and just exposed as a simple network service with an endpoint for you to call. The Caching service is a highly available multitenant service with no management overhead for its users.

As a user, what you get is a secure Windows Communication Foundation (WCF) endpoint to talk to and the amount of usable memory you need for your application and APIs for the cache client to call in to store and retrieve data. The key here is usable memory. If you ask for 1GB cache memory, you get 1GB of usable memory to store your objects, unlike the amount of memory available on a Windows Azure instance you buy. The Caching service does the job of pooling in memory from the distributed cluster of machines it's running and managing to provide the amount of usable memory you need. As a result, it also automatically provides the flexibility to scale up or down based on your cache needs with a simple change in the configuration. In that sense, think of the Caching service as a virtual pool of partitioned and shared memory that you can consume flexibly.

The other under-the-hood design point to understand is that the Caching service automatically partitions your cache so that you don't lose data in the event of a machine going down, even if you haven't bought the more expensive high availability (HA) option for your cache. (The HA option is not currently available in Windows Azure AppFabric Caching, and only available on Windows Server AppFabric Caching today.) This partitioning scheme increases the performance and reduces the data loss probability automatically for you without ever having to learn about the back-end service.

The third and last part of the puzzle is the connection between your cache client and Caching service. The communication between the cache client and service is using WCF, and the WCF programming model is abstracted from the developer as the cache client translates GET/PUT calls to a WCF protocol that the service understands. The thing you need to know about this communication channel is that it's secure, which is extremely critical, especially in the cloud world. Your cache is secured using an Access Control service

token (which you get when you create the named cache). The Caching service uses this token to restrict access to your cached data by your client.

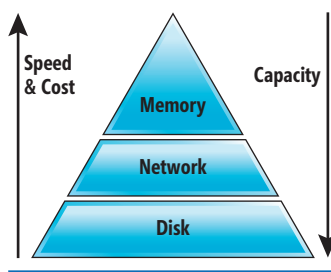


Figure 1 Memory, Network and Disk Usage in Data Manipulation Scenarios

Architectural Guidance

We've spent a lot of time with customers who have varying degrees of application complexity. Usually, before we start whiteboarding the design choices and architecture, we'll draw the simple diagram shown in **Figure 1**. For most cases this diagram captures the trade-offs between the three most basic elements involved in storing and manipulating data. (We've had some storage gurus argue

Figure 2 Data in Caching Scenarios

Data Type	Access Pattern
Reference	Shared read
Activity	Exclusive write
Resource	Shared, concurrently read and written into, accessed by a large number of transactions

with us that they can saturate network before they can max the disk throughput, hence we qualify the statement about “most” cases.)

The basic principle is that your memory on the local machine provides the fastest data access with lowest latency, but is limited to the amount of usable memory on the machine. As soon as you need more memory than your local machine can give you or you need to externalize the data from your compute tier (either for shared state or for more durability), the minimum price you pay is for the network hop. The slowest in this spectrum is storing data on a disk (solid-state drives help a little, but are still quite expensive).

Disk is the cheapest and largest store while memory is the most expensive and hence the most constrained in terms of capacity. The Caching service balances the various elements by providing the service as a network service to access a large chunk of distributed and shared memory across multiple compute tiers. At the same time, it provides an optimization with the local cache feature to enable a subset of the data to additionally reside on the local machine while removing the complexity of maintaining consistency between the local cache and the cache tier in the service.

With this background, let’s look at some top-level architectural considerations for using cache in your application.

What data should you put in the cache? The answer varies significantly with the overall design of your application. When we talk about data for caching scenarios, usually we break it into the data types and access patterns shown in **Figure 2** (see msdn.microsoft.com/library/ee790832 for a deeper explanation of these data access patterns).

Using this model to think about your data, you can plan for capacity and access patterns (for managing consistency, eviction, refreshes and so on) and restrict the data to the most frequently used or time-sensitive data used or generated by your application.

As an example, reference data should be readily partitioned into frequently accessed versus infrequently accessed to split between cache and storage. Resource data is a classic example where you

want to fit as much as possible in cache to get the maximum scale and performance benefits. In addition to the cache tier, even the use of local cache on the client goes hand in hand with the data type. Reference data is a great candidate for keeping in the local cache or co-located with the client; while in most cases resource data can become quite chatty for the local cache due to frequent updates and hence best fits on the cache tier.

As the second-most-expensive-resource and usually the bottleneck for most inefficient implementations, network traffic patterns for accessing cache data is something to which you should pay particular attention. If you have a large number of small objects, and you don’t optimize for how frequently and how many objects you fetch, you can easily get your app to be network-bound. Using tags to fetch like data or using local cache to keep a large number of frequently accessed small objects is a great trade-off.

While the option to enable HA for a named cache is not yet available in the Caching service, it’s another factor to consider in your application design. Some developers and architects choose to use cache only as a transient cache. However, others have taken the leap of faith to move exclusively to storing a subset of their data (usually activity data) only in the cache by enabling the HA feature. HA does have a cost overhead, but it provides a design model that treats the cache as the only store of data, thereby eliminating the need to manage multiple data stores.

However, the cache is not a database! We cannot stress this point too strongly. The topic of HA usually makes it sound like the cache can replace your data tier. This is far from the truth—a SQL database is optimized for a different set of patterns than the cache tier is designed for. In most cases, both are needed and can be paired to provide the best performance and access patterns while keeping the costs low.

How about using the cache for data aggregation? This is a powerful yet usually overlooked scenario. In the cloud, apps often deal with data from various sources and the data not only needs to be aggregated but also normalized. The cache offers an efficient, high-performance alternative for storing and managing this aggregated data with high throughput normalization (in-memory as opposed to reading from and writing to disk), and the normalized data structure of the cache using key-value pairs is a great way to think about how to store and serve this aggregated data.

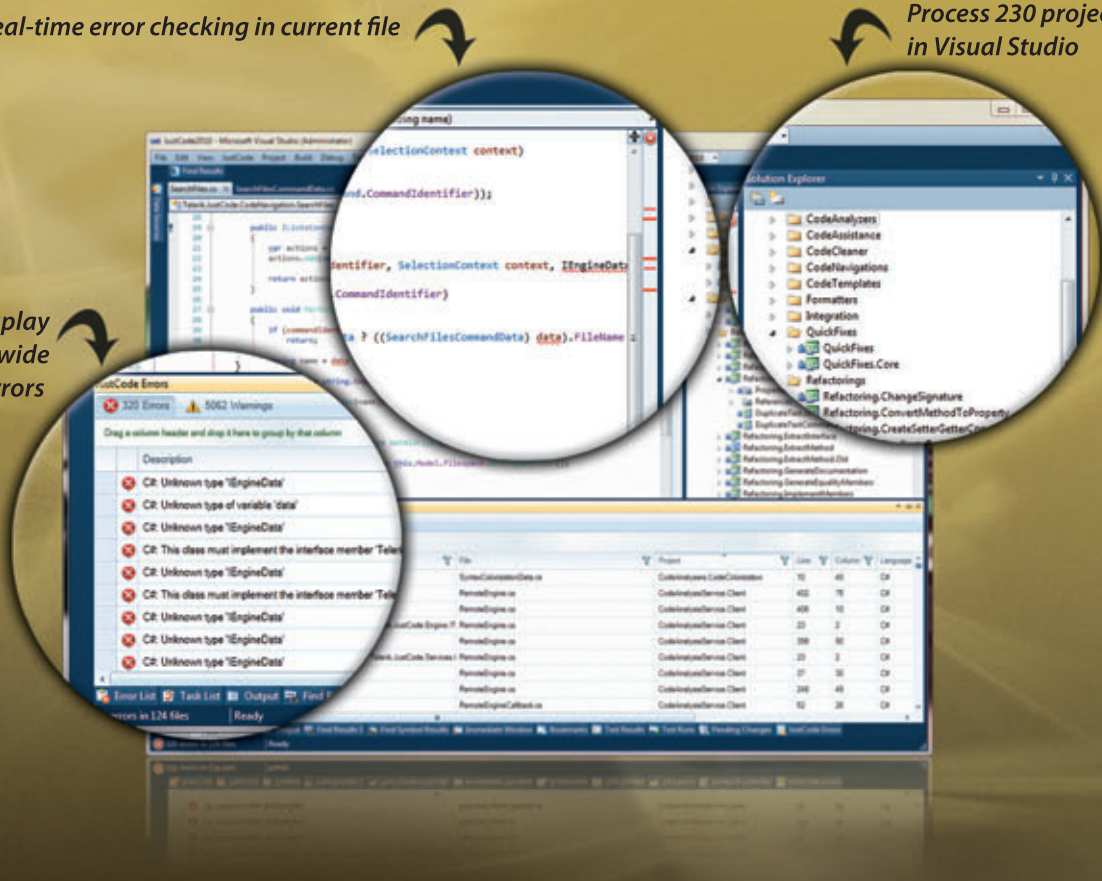
A common problem that application developers and architects have to deal with is the lack of guarantee that a client will always be routed to the same server that served the previous request. When these sessions can’t be sticky, you’ll need to decide what to store in session state and how to bounce requests between servers to work around the lack of sticky sessions. The cache offers a compelling alternative to storing any shared state across multiple compute nodes. (These nodes would be Web servers in this example, but the same issues apply to any shared compute tier scenario.) The shared state is consistently maintained automatically by the cache tier for access by all clients, and

Figure 3 Configuring a New Cache Service Namespace

Real-time error checking in current file

Process 230 projects
in Visual Studio

Instant display
of solution-wide
errors



Telerik JustCode

Write better .NET and JavaScript code. Faster.

- Agile development add-in for Visual Studio
- Fast, solution-wide, real time code analysis
- First-class JavaScript support
- Refactorings and Test Runner

Try now at: www.telerik.com/justcodemsdn

Exclusive offer for MSDN Magazine readers*

Purchase Telerik JustCode online and receive a **20% discount**.

At checkout, use coupon: **DEVPR0M-MSDNJC2**

*valid until May 31, 2011



2010
Microsoft Central & Eastern Europe
PARTNER OF THE YEAR
Winner

 **telerik**
deliver more than expected

at the same time there's no overhead or latency of having to write it to a disk (database or files).

The trade-off to consider here is that if you need ultra-low latency access to the shared state (session state in an online gaming Web site that tracks scores in real time, for example), then an external cache tier might not be your best option. For most other scenarios, using the cache tier is a quick yet powerful way to deal with this design pattern, which automatically eliminates the staleness from your data.

Be sure to spend enough time in capacity planning for your cache. Number of objects, size of each object, frequency of access of each object and pattern for accessing these objects are all critical in not only determining how much cache you need for your application, but also on which layers to optimize for (local cache, network, cache tier, using regions and tags, and so on). Remember that you can start with a simple configuration switch in your web.config file to start using caching without writing a single line of code, and you can spend years in designing a solution that uses caching in the most efficient way.

Setting Up AppFabric Caching

To get started using the Windows Azure AppFabric Caching service, head to portal.appfabriclabs.com. This is the CTP portal that you can use to get familiar with the Caching service. It doesn't cost anything, but there are no service level agreements for the service.

On the portal, select the Cache option and then create a new cache by clicking New Namespace. The dialog box to configure the Cache service namespace is shown in **Figure 3**.

The only two options you need to specify are a unique service namespace and the cache size (of which you can choose between 128MB and 256MB in the CTP). When you choose OK the service will provision the cache for you in the background. This typically takes 10 to 15 seconds. When complete, you have a fully functional, distributed cache available to your applications.

Now that it's created, you can take a look at the properties of your cache, shown in **Figure 4**. (Note that we've obfuscated some account-specific information here.)

You can see that we've created a cache using the namespace of CachingDemo. There are a few pieces of important information that you'll want to grab, as we'll use them later in our code: the Service URL and the Authentication Token. The Service URL is the TCP endpoint that your application will connect to when interacting with the Caching service. The Authentication Token is an encrypted token that you'll pass along to Access Control to authenticate your service.

Caching in Your App

Now, before you start coding, download the Windows Azure AppFabric SDK. You can find it at go.microsoft.com/fwlink/?LinkID=184288 or click the link on the portal.

Make sure you don't have the Windows Server AppFabric Cache already installed on your machine. While the API is symmetric,

Properties	
General	
Namespace	CachingDemo
Subscription	LabsSubscription
Region	United States (South/Ce
Status	Active
Created on	2/6/2011 3:44:34 AM U
Subscription ID	
Project ID	
Cache	
Service URL	CachingDemo.cache.app
Service Port	22233
Authentication Token	
Current	256MB Cache
Pending	No pending changes
Effective Date	2/6/2011 3:44:34 AM GI

Figure 4 Cache Service Properties

the current assemblies are not compatible. The Windows Server AppFabric Cache registers its Caching assemblies in the Global Assembly Cache (GAC), so your application will load the wrong assemblies. This will be resolved by the time the service goes into production, but for now it causes a bit of friction.

To begin, let's create a simple console application using C#. Once created, be sure to update the project so that it targets the full Microsoft .NET Framework instead of the Client Profile. You'll also need to add the Caching assemblies, which can typically be found under C:\Program Files\Windows Azure AppFabric SDK\V2.0\Assemblies\Cache. For now, add the following two assemblies:

- Microsoft.ApplicationServer.Caching.Client
- Microsoft.ApplicationServer.Caching.Core

One thing that changed between the October CTP and the February refresh is that you now must use the System.Security.SecureString for your authentication token. The purpose of SecureString is to keep you from putting your password or token into memory within your application, thereby keeping it more secure. However, to make this work in a simple console application, you'll have to create the following method:

```
static private SecureString Secure(string token) {  
    SecureString secureString = new SecureString();  
    foreach (char c in token) {  
        secureString.AppendChar(c);  
    }  
    secureString.MakeReadOnly();  
    return secureString;  
}
```

While this defeats the purpose of SecureString by forcing you to load the token into memory, it's only used for this simple scenario.

Now, the next thing to do is write the code that will set us up to interact with the Caching service. The API uses a factory pattern,

Figure 5 Loading the Default Cache

```
private static DataCache configureDataCache(  
    SecureString authorizationToken, string serviceUrl) {  
  
    // Declare an array for the cache host  
    List<DataCacheServerEndpoint> server =  
        new List<DataCacheServerEndpoint>();  
    server.Add(new DataCacheServerEndpoint(serviceUrl, 22233));  
  
    // Set up the DataCacheFactory configuration  
    DataCacheFactoryConfiguration conf =  
        new DataCacheFactoryConfiguration();  
    conf.SecurityProperties =  
        new DataCacheSecurity(authorizationToken);  
    conf.Servers = server;  
  
    // Create the DataCacheFactory based on config settings  
    DataCacheFactory dataCacheFactory =  
        new DataCacheFactory(conf);  
  
    // Get the default cache client  
    DataCache dataCache = dataCacheFactory.GetDefaultCache();  
  
    // Return the default cache  
    return dataCache;  
}
```

Build Interactive Reports with ZERO Code. Deliver Them to All .NET Platforms.



Telerik Reporting is the first .NET solution to offer developers the ability to easily deliver interactive and consistent reports to the most used Microsoft platforms - Windows Azure, SharePoint 2010, Silverlight, WPF, ASP.NET and Windows Forms. The tool offers a completely codeless experience and swift performance, fueled by a robust OLAP engine.

- Interactive reports in any .NET business application
- Powerful OLAP Data Processing Engine
- Extensive database support (including cubes)
- Intuitive WYSIWYG report designer with countless wizards, builders, previews and tools
- Numerous export formats and end-user features

Try now at: www.telerik.com/InteractiveReports



2010
Microsoft Central & Eastern Europe
PARTNER OF THE YEAR
Winner

telerik
deliver more than expected

so we'll have to define the factory, set some configuration settings and then load our default cache as shown in **Figure 5**.

You can see that we've defined a new `DataCacheServerEndpoint` based on a service URL (that we'll provide) that points to port 22233. We then create the `DataCacheFactoryConfiguration`, pass in our authentication token (which is a `SecureString`) and set it to the security properties—this will enable us to authenticate to the service. At this point it's simply a matter of constructing the `DataCacheFactory`, getting the `DataCache` based on the default cache and returning the default cache.

While it's not required to encapsulate this logic in its own method, it makes it much more convenient later on.

At this point, it's pretty simple to pull this together in our application's `Main` method (see **Figure 6**).

We provide the authentication token and service URL to our application, then pass them into the `configureDataCache` method, which sets the `DataCache` variable to the default cache. From here we can grab some input from the console, put it into the cache and then call `Get` on the key, which returns the value. This is a simple, but valid, test of the cache.

Including the tokens and service URL in the code is not ideal. Fortunately, the portal provides you with XML that you can insert within your `app.config` (or `web.config`) file, and the APIs will manage everything for you.

In the portal, select your cache, then click the `View Client Configuration` button. This opens a dialog that provides the configuration XML. Copy the XML snippet and paste it into your configuration file. The end result will look like **Figure 7**.

Now we can heavily refactor our code, get rid of the `createSecureString` and `configureDataCache` methods, and be left with this:

```
static void Main(string[] args) {
    DataCacheFactory dataCacheFactory =
        new DataCacheFactory();
    DataCache dataCache = dataCacheFactory.GetDefaultCache();
    Console.WriteLine("Enter a value: ");
    string value = Console.ReadLine();
    dataCache.Put("key", value);
    string response = (string)dataCache.Get("key");
    Console.WriteLine("Your value: " + response);
}
```

Figure 6 Console Application Main Method

```
static void Main(string[] args) {
    // Hardcode your token and service url
    SecureString authorizationToken =
        createSecureString("YOURTOKEN");
    string serviceUrl = "YOURCACHE.cache.appfabriclabs.com";

    // Create and return the data cache
    DataCache dataCache =
        configureDataCache(authorizationToken, serviceUrl);

    // Enter a value to store in the cache
    Console.WriteLine("Enter a value: ");
    string value = Console.ReadLine();

    // Put your value in the cache
    dataCache.Put("key", value);

    // Get your value out of the cache
    string response = (string)dataCache.Get("key");

    // Write the value
    Console.WriteLine("Your value: " + response);
}
```

You can see that all we have to do is create a new instance of the `DataCacheFactory`, and all the configuration settings in the `app.config` file are read in by default.

As you've seen, you can use the APIs directly or manage `DataCacheFactory` in your configuration. While we've only performed `PUT` and `GET` operations on simple data, we could easily store data retrieved from SQL Azure, Windows Azure or another provider of data. For a more complex look at using the cache for reference data stored in SQL Azure, refer to the `Caching Service Hands-On Lab` in the `Windows Azure Platform Training Course` (msdn.microsoft.com/gg457894).

Storing Session Data

Next, let's take a look at how to use the `Caching` service to store the session data for our ASP.NET Web application. This is a powerful technique, as it allows us to separate the session state from the in-process memory of each of our Web clients, thus making it easy to scale our applications beyond one instance in Windows Azure.

This is important for services that don't support sticky sessions, such as Windows Azure. There's no way to guarantee that a user will hit the same instance with each additional request—in fact, the Windows Azure load balancer explicitly uses a round-robin approach to load balancing, so it's likely that your user will hit a new instance. By using the `Caching` service for session state, it doesn't matter which instance your user hits because all the instances are backed by the same session state provider.

To begin, create a new Windows Azure Project and add an ASP.NET Web Role. In the Web Role, add all of the assemblies provided by the Windows Azure AppFabric SDK, including:

- `Microsoft.ApplicationService.Caching.Client`
- `Microsoft.ApplicationService.Caching.Core`
- `Microsoft.Web.DistributedCache`
- `Microsoft.WindowsFabric.Common`
- `Microsoft.WindowsFabric.Data.Common`

Next, update your `web.config` file so that the very first things to follow the `<configuration>` element are the `<configSections>` and `<dataCacheClient>` elements (you'll receive errors if they aren't the first elements).

Now, the key to using the `Caching` service for session state is the `Microsoft.Web.DistributedCache` assembly. It contains the custom session state provider that uses the `Caching` service. Return to the LABS portal where you grabbed the XML for your configuration files, and find the `<sessionState>` element—you can place this directly in the `<system.web>` element of your `web.config` file and it will immediately tell your application to start leveraging the `Caching` service for session state:

```
<system.web>
  <sessionState mode="Custom"
    customProvider="AppFabricCacheSessionStoreProvider">
    <providers>
      <add name="AppFabricCacheSessionStoreProvider"
        type="Microsoft.Web.DistributedCache.
          DistributedCacheSessionStateStoreProvider, Microsoft.Web.DistributedCache"
        cacheName="default"
        useBlobMode="false" />
    </providers>
  </sessionState>
  ...
</system.web>
```


Writing a Windows Phone 7 App?



Get a **better-looking** app to the marketplace **faster** with Telerik RadControls for Windows Phone

- 15 native Windows Phone controls
- Data visualization controls
- Light-weight and blazing fast

Download a **FREE** trial now at: www.telerik.com/phonemsdn

Exclusive offer for MSDN Magazine readers*

Buy online and at checkout use coupon: **DEVPR0M-MSDNWP2** for **20% discount**.

*The offer is valid till May 31st.



2010
Microsoft Central & Eastern Europe
PARTNER OF THE YEAR
Winner

 **telerik**
deliver more than expected

Figure 7 Client Configuration

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section
      name="dataCacheClient"
      type="Microsoft.ApplicationServer.Caching.DataCacheClientSection,
Microsoft.ApplicationServer.Caching.Core"
      allowLocation="true"
      allowDefinition="Everywhere"/>
  </configSections>
  <dataCacheClient deployment="Simple">
    <hosts>
      <host
        name="YOURCACHE.cache.appfabriclabs.com"
        cachePort="22233" />
    </hosts>
    <securityProperties mode="Message">
      <messageSecurity
        authorizationInfo="YOURTOKEN">
      </messageSecurity>
    </securityProperties>
  </dataCacheClient>
</configuration>
```

To validate that this is working, open the Global.asax.cs file and add the following code to the Session_Start method:

```
void Session_Start(object sender, EventArgs e) {
    int i = 0;
    while (i < 10) {
        Session.Add(Guid.NewGuid().ToString(), DateTime.Now.ToString());
        i++;
    }
}
```

This will add 10 random items into your session context. Next, open up your Default.aspx.cs page and update the Page_Load method:

```
protected void Page_Load(object sender, EventArgs e) {
    foreach (var key in Session.Contents) {
        Response.Write("key: " + key + ", guid: " +
            Session[key.ToString()].ToString() + "<br/>");
    }
}
```

This will write out all the values that were added into the session context. Finally, open the ServiceConfiguration.cscfg file and increase the instance count from 1 to 2:

```
<Instances count="2" />
```

Now, when you hit F5, you'll get two instances of your application running in the Compute Emulator. Notice that no matter how many times you refresh the page, you'll always have the same 10 values in your session state—this is because it's a shared session, and session start only runs once. Conversely, if you don't use the Caching service as your session state provides, but opt to keep the default in-process choice, you'll have different values on each of the instances.

What's Next?

Windows Azure AppFabric Caching service is slated to go into production as a commercial service in the first half of 2011. In the first commercial release, some of the features that are available in Windows Server AppFabric will not yet be available. Some of these exclusions are purposeful, as they may not apply in the cloud world. However, features like notifications are relevant in Windows Azure as well and are critical in completing the local-cache scenario, and hence are part of the short-term roadmap for Microsoft. Similarly, the option of turning on HA for a given named cache as a premium capability is also high on the priority list.

The growing popularity of Windows Server AppFabric Caching has resulted in a number of new feature requests that open up the applicability of caching to an even broader set of scenarios. Some of the capabilities that are being discussed include the ability to perform rich queries on the cache and enabling an easier way to retrieve bulk data from a named cache.

In addition, the success of the Caching session state provider scenarios with ASP.NET has resulted in requests for the ability to associate write-behind and read-through queries with the cache so that the cache can become the primary way to manipulate data, while letting the associated queries update the data tier in the back end.

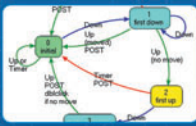
We'll be evaluating these and other features for possible inclusion in future releases of Windows Azure AppFabric Caching. In the meantime, we encourage you to experiment with the current Caching service implementation and let us know how it works for you. ■

KARANDEEP ANAND is a principal group program Manager with the AppFabric product group at Microsoft. His team is responsible for building the next generation application platform and services for the Windows Azure Platform and Windows Server. You can reach him at Karandeep.Anand@microsoft.com.


WADE WEGNER is a technical evangelist at Microsoft, responsible for influencing and driving the Microsoft technical strategy for the Windows Azure platform. You can reach him through his blog at wadewegner.com or on Twitter at twitter.com/WadeWegner.

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

34 msdn magazine

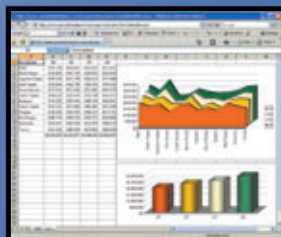
Cloud Cache

Microsoft Chose SpreadsheetGear...

"After carefully evaluating SpreadsheetGear, Excel Services, and other 3rd party options, we ultimately chose SpreadsheetGear for .NET because it is the best fit for MSN Money."

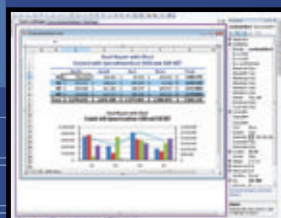
Chris Donohue, MSN Money Program Manager

SpreadsheetGear 2010



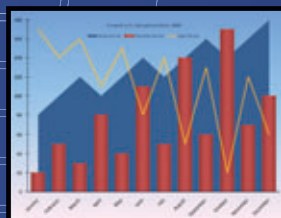
ASP.NET Excel Reporting

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.



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

CQRS on Windows Azure

Mark Seemann

Microsoft Windows Azure offers unique opportunities and challenges. There are opportunities in the form of elastic scalability, cost reduction and deployment flexibility, but also challenges because the Windows Azure environment is different from the standard Windows servers that host most Microsoft .NET Framework services and applications today.

One of the most compelling arguments for putting applications and services in the cloud is *elastic scalability*: You can turn up the

power on your service when you need it, and you can turn it down again when demand trails off. On Windows Azure, the least disruptive way to adjust power is to scale out instead of up—adding more servers instead of making existing servers more powerful. To fit into that model of scalability, an application must be dynamically scalable. This article describes an effective approach to building scalable services and demonstrates how to implement it on Windows Azure.

Command Query Responsibility Segregation (CQRS) is a new approach to building scalable applications. It may look different from the kind of .NET architecture you're accustomed to, but it builds on tried and true principles and solutions for achieving scalability. There's a vast body of knowledge available that describes how to build scalable systems, but it requires a shift of mindset.

Taken figuratively, CQRS is nothing more than a statement about separation of concerns, but in the context of software architecture, it often signifies a set of related patterns. In other words, the term CQRS can take on two meanings: as a pattern and as an architectural style. In this article, I'll briefly outline both views, as well as provide examples based on a Web application running on Windows Azure.

Understanding the CQRS Pattern

The underlying terminology for CQRS originates in object-oriented pattern language. A Command is an operation that changes the state of something, whereas a Query is an operation that retrieves information about state. More informally, Commands are *writes* and Queries are *reads*.

The CQRS pattern simply states that reads and writes must be explicitly modeled as segregated responsibilities. Writing data is

This article discusses:

- Understanding the CQRS pattern
- CQRS architectural style
- A reservation booking application
- Submitting commands
- Processing commands
- Making write operations idempotent
- Using optimistic concurrency with Windows Azure Storage
- Updating view data
- Querying view data

Technologies discussed:

Windows Azure; Command Query Responsibility Segregation; ASP.NET MVC 2

Code download available at:

code.msdn.microsoft.com/mag201104CQRS

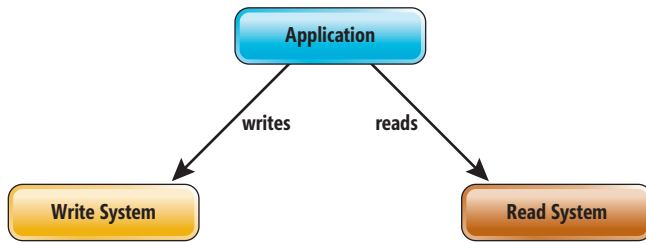


Figure 1 Segregating Reads from Writes

one responsibility, and reading data is another. Most applications need to do both, but as **Figure 1** shows, each responsibility must be treated separately.

The application writes to a different conceptual system than it reads from.

Obviously, the data that the application writes should eventually end up being available for reading. The CQRS pattern says nothing about how this could happen, but in the simplest possible implementation, the read and write systems could use the same underlying data store.

In this world view, reads and writes are strictly segregated; writes *never* return data. That seemingly innocuous statement opens up a rich set of opportunities for creating massively scalable applications.

CQRS Architecture Style

In addition to the CQRS pattern, the foundation of the CQRS architectural style is a simple but profound realization about displaying data. Consider **Figure 2**, which shows the UI for a booking app—imagine that it’s a restaurant reservation system.

The calendar shows dates in a given month, but some dates are disabled because they’re already fully booked.

How up-to-date is the data in such a UI? In the time it takes to render the data, transport it over the network, and for the user to interpret it and react to it, it may already have changed in the underlying data store. The longer the user looks at it before reacting, the staler it becomes. The user may be interrupted by a phone call or similar distraction before continuing, so think times (the time spent by a user perusing a Web page) may be measured in minutes.

A common way to address this issue is by using optimistic concurrency to handle the cases where conflicts occur. Application developers must write the code to handle such situations, but instead of treating them as exceptional cases, the CQRS architectural style embraces this basic condition. When display data is stale the moment it’s rendered, it doesn’t *have* to reflect the data in the central data store. Instead, an application can display data from a denormalized data source that may lag a bit behind the “real” data store.

The realization that display data is always stale, coupled with the CQRS principle that writes never return data, results in scalability opportunities. UIs don’t have to wait for data to be written, but can instead just send an asynchronous message and return a view to

the user. Background workers pick up the messages and process them at their own pace. **Figure 3** shows a more comprehensive view of the CQRS-style architecture.

Whenever the application needs to update data, it sends a Command as an asynchronous message—most likely via a durable queue. As soon as the Command is sent, the UI is free to return a view to the user. A background worker picks up the Command message in a separate process and writes any appropriate changes to the data store. As part of this operation, it also raises an event as another asynchronous message. Other message handlers can subscribe to such events and update a denormalized view of the data store accordingly.

Although the view data will lag behind the “real” data, this event propagation often happens so fast that users never notice it. But even if the system slows down due to excessive load, the view data will eventually be consistent.

UIs don’t have to wait for data to be written, but can instead just send an asynchronous message and return a view to the user.

This sort of architecture can be implemented on many different systems, but with its explicit concepts of Worker Roles and queues, Windows Azure is quite well-suited for it. However, Windows Azure also presents some unique challenges in relation to CQRS; the remainder of this article explores both opportunities and challenges through a sample application.

A Reservation Booking Application

A simple booking application serves as an excellent example of how to implement CQRS on Windows Azure. Imagine that the application takes reservation requests for a restaurant. The first page that meets the user is a date picker, as shown in **Figure 2**—notice again that some dates are already disabled to indicate that these dates are sold out.

When the user clicks an available date, a reservation form and subsequent receipt is displayed, as shown in **Figure 4**.

Notice that the receipt page makes an effort to inform the user that the reservation isn’t guaranteed at this point. The final decision will be communicated via e-mail.

In CQRS, the UI plays an important part in setting expectations because processing happens in the background. However, during normal load, a receipt page buys enough time that when the user moves on, the request has already been handled.

I’ll now demonstrate key points about implementing the sample booking app. As there are many moving parts in even this

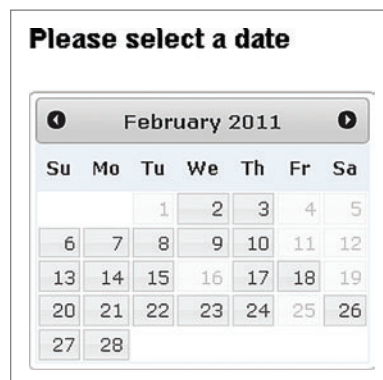


Figure 2 Display Data Is Stale at the Moment It’s Rendered

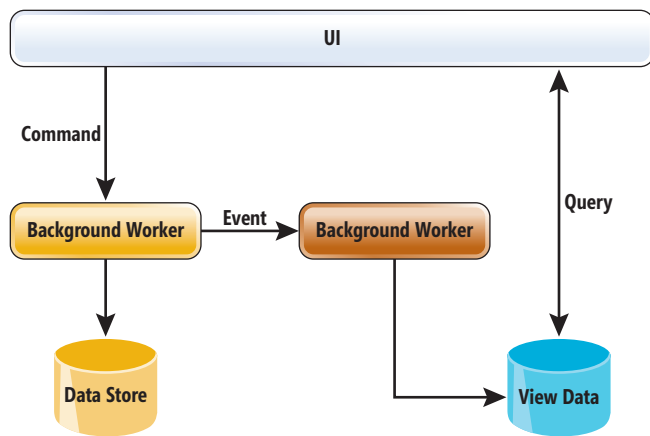


Figure 3 CQRS-Style Architecture

simple application, I'll focus on the most interesting code snippets here; the full code is available in the download accompanying this article.

Submitting Commands

The Web Role is implemented as an ASP.NET MVC 2 application. When the user submits the form shown in **Figure 4**, the appropriate Controller Action is invoked:

```
[HttpPost]
public ActionResult NewBooking(BookingViewModel model)
{
    this.channel.Send(model.MakeNewReservation());
    return this.View("BookingReceipt", model);
}
```

The channel field is an injected instance of this simple `IChannel` interface:

```
public interface IChannel
{
    void Send(object message);
}
```

The command that the `NewBooking` method sends through the channel is just the HTML form's data encapsulated in a Data Transfer Object. The `MakeNewReservation` method simply transforms the posted data into a `MakeReservationCommand` instance, as shown here:

```
public MakeReservationCommand MakeNewReservation()
{
    return new MakeReservationCommand(this.Date,
        this.Name, this.Email, this.Quantity);
}
```

Because the `Send` method returns void, the UI is free to return an HTML page to the user as soon as the Command is successfully sent. Implementing the `IChannel` interface on top of a queue ensures that the `Send` method returns as quickly as possible.

On Windows Azure, we can implement the `IChannel` interface on top of the built-in queues that are part of Windows Azure Storage. To put messages on such a durable queue, the implementation must serialize the messages. There are many different ways to do this, but to keep things

simple I've chosen to use the binary serializer built in to the .NET Framework. However, in a production application, you should seriously consider alternatives, as the binary serializer makes it difficult to handle versioning issues. For example, what happens when a new version of your code attempts to deserialize a blob serialized by an old version? Possible alternatives include XML, JSON or Protocol Buffers.

With this technology stack, the implementation of `IChannel.Send` is simple:

```
public void Send(object command)
{
    var formatter = new BinaryFormatter();
    using (var s = new MemoryStream())
    {
        formatter.Serialize(s, command);
        var msg = new CloudQueueMessage(s.ToArray());
        this.queue.AddMessage(msg);
    }
}
```

The `Send` method serializes the Command and creates a new `CloudQueueMessage` out of the resulting byte array. The queue field is an injected instance of the `CloudQueue` class from the Windows Azure SDK. Initialized with the correct address information and credentials, the `AddMessage` method adds the message to the appropriate queue. This usually happens surprisingly fast, so when the method returns, the caller is free to perform other work. At the same time, the message is now in the queue, waiting for a background processor to pick it up.

Processing Commands

While the Web Roles are happily displaying HTML and accepting data that they can send via the `IChannel` interface, Worker Roles receive and process messages from the queue at their own pace. These background workers are stateless, autonomous components, so if they can't keep up with the incoming messages you can add more instances. This is what provides the massive scalability of messaging-based architecture.

As was previously demonstrated, sending messages over Windows Azure queues is easy. Consuming them in a safe and consistent manner is a bit trickier. Each Command encapsulates an intention to change the state of the application, so the background worker must make sure that no message is lost and that underlying data is changed in a consistent way.

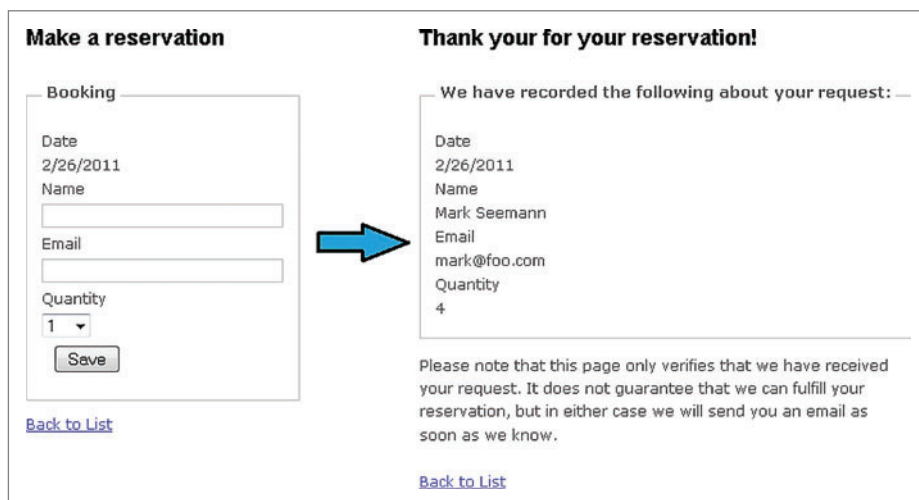


Figure 4 Reservation UI Flow



- fast
- comprehensive
- flexible

dotConnect



Advanced ADO.NET providers with ORM support

- ▣ Oracle, MySQL, PostgreSQL, and SQLite support
 - ▣ Entity Framework, LinqConnect, and NHibernate ORMs support
 - ▣ Fast data access for your applications
 - ▣ Database specific features support
 - ▣ Visual Studio integration
 - ▣ Rich design-time support
 - ▣ Enterprise Library Data Access Application Block
- ▣ Integration with Microsoft Business Intelligence Solutions

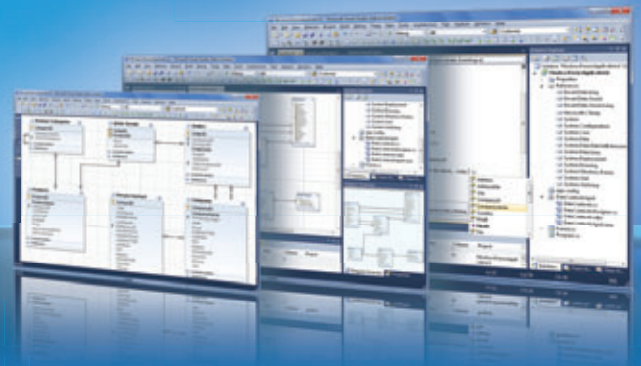
100%
Free
Edition

30-day
Fully
Functional
Trial

1 Year
Free
Updates



LinqConnect



Extended LINQ to SQL technology for your database

- ▣ Working with wide-spread database servers
- ▣ Compatibility with MS LINQ to SQL while extending its functionality
 - ▣ Own Visual Model Designer – Entity Developer
 - ▣ Comprehensive LINQ implementation
- ▣ ADO.NET data providers included
 - ▣ Many ways to optimize performance for your applications
- ▣ Monitoring of interaction with the database server

Figure 5 PollForMessage Method

```
public void PollForMessage(CloudQueue queue)
{
    var message = queue.GetMessage();
    if (message == null)
    {
        Thread.Sleep(500);
        return;
    }

    try
    {
        this.Handle(message);
        queue.DeleteMessage(message);
    }
    catch (Exception e)
    {
        if (e.IsUnsafeToSuppress())
        {
            throw;
        }
        Trace.TraceError(e.ToString());
    }
}
```

This might be fairly easy to ensure for a queuing technology that supports distributed transactions (such as Microsoft Message Queuing). Windows Azure queues aren't transactional, but they do come with their own set of guarantees. Messages aren't lost when read, but rather made invisible for a period of time. Clients should pull a message off the queue, perform the appropriate operations and delete the message as the last step in the process. This is what the sample booking application's general-purpose Worker Role does; it executes the PollForMessage method shown in **Figure 5** in an endless loop.

The GetMessage method may return null if no message is currently in the queue. In that case, the method simply waits 500 milliseconds and returns, in which case it will immediately be invoked again by the endless outer loop. When a message is received, the method handles the message by invoking the Handle method. This is where all the real work is supposed to happen, so if that method returns without throwing an exception, it's safe to delete the message.

On the other hand, if an exception happens while handling the message, it's important to suppress the exception; an unhandled

exception will crash the entire worker instance and it will stop pulling messages off the queue.

A production-ready implementation needs to be more sophisticated to handle so-called *poison messages*, but I decided to leave this out of the sample code to keep it simpler.

If an exception is thrown while a message is being processed, it won't be deleted. After a timeout, it will become available for processing once again. This is the guarantee that Windows Azure queues provide: A message can be processed *at least once*. As a corollary, it may be replayed several times. Thus, all background workers must be able to handle message replays. It's essential that all durable write operations are idempotent.

Making Write Operations Idempotent

Every method that handles a message must be able to deal with replays without compromising the state of the application. Handling a MakeReservationCommand is a good example. **Figure 6** provides an overview of the message flow.

The first thing the application must do is check if the restaurant has enough capacity for the requested date; all tables may already be reserved for the given date, or there may only be a few places left. To answer the question about available capacity, the application tracks the current capacity in durable storage. There are several options for doing this. Tracking all reservation data in a SQL Azure database is one possibility, but as there are limits to the size of SQL Azure databases, a more scalable option is to use either Windows Azure blob or table storage.

The booking sample application uses blob storage to store a serialized idempotent Value Object. This Capacity class keeps track of accepted reservations so that it can detect message replays. To answer the question about remaining capacity, the application can load a Capacity instance for the appropriate day and invoke the CanReserve method with the correct reservation ID:

```
public bool CanReserve(int quantity, Guid id)
{
    if (this.IsReplay(id))
    {
        return true;
    }
    return this.remaining >= quantity;
}

private bool IsReplay(Guid id)
{
    return this.acceptedReservations.Contains(id);
}
```

Each MakeReservationCommand has an associated ID. To ensure idempotent behavior, the Capacity class saves each accepted reservation ID so that it can detect replays. Only if the method call isn't a replay does it invoke the actual business logic, comparing the requested quantity against the remaining capacity.

The application serializes and stores a Capacity instance for each date, so to answer the question of whether the restaurant has remaining capacity, it downloads the blob and invokes the CanReserve method:

```
public bool HasCapacity(MakeReservationCommand reservation)
{
    return this.GetCapacityBlob(reservation)
        .DownloadItem()
        .CanReserve(reservation.Quantity, reservation.Id);
}
```

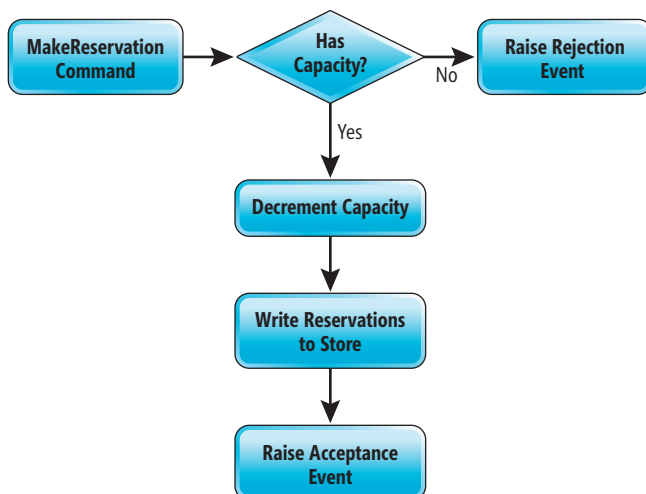


Figure 6 Workflow for Handling the Make Reservation Command

Our Name Says It All

activePDF®

Come and see why thousands of customers have trusted us over the last decade for all their server based PDF development needs.

- . Convert over 400 files types to PDF
- . High fidelity translation from PDF to Office
- . ISO 32000 Support including PDF/X & PDF/A
- . HTML5 to PDF
- . True PDF Print Server
- . Form Fill PDF
- . Append, Stamp, Secure and Digitally Sign



Download our FREE evaluation from
www.activepdf.com/MSDN

Call 1-866-GoTo-PDF | 949-582-9002 | Sales@activepdf.com

If the answer is “true,” the application invokes the set of operations associated with that outcome, as shown in **Figure 6**. The first step is to decrement the capacity, which involves invoking the `Capacity.Reserve` method shown in **Figure 7**.

This is another idempotent operation that first invokes the `CanReserve` and `IsReplay` methods as guards. If the method call represents a genuinely new request to reserve some capacity, a new `Capacity` instance is returned with the decremented capacity and the ID is added to the list of accepted IDs.

The `Capacity` class is just a Value Object, so it must be committed back to the Windows Azure blob storage before the operation is complete. **Figure 8** shows how the original blob is initially downloaded from Windows Azure blob storage.

This is the serialized `Capacity` instance that corresponds to the date of the requested reservation. If the capacity changed (that is, it wasn't a replay) the new `Capacity` is uploaded back to blob storage.

What happens if exceptions are thrown along the way? One way this could happen would be if the `Capacity` instance has changed since the `CanReserve` method was invoked. This is not unlikely in high-volume scenarios where many competing requests are being handled concurrently. In such cases, the `Reserve` method might throw an exception because there's not enough remaining capacity. That's OK; this simply means that this particular reservation request has lost a concurrent race. The exception will be caught by the exception handler in **Figure 5**, but because the message was never deleted, it will later reappear to be handled once more. When this happens, the `CanReserve` method will immediately return false and the request can be politely rejected.

However, another potential concurrency conflict lurks in **Figure 8**. What happens when two background workers update the capacity for the same date at the same time?

Using Optimistic Concurrency

The `Consume` method in **Figure 8** downloads the `Capacity` blob from blob storage and uploads a new value if it changed. Many background workers may be doing this concurrently, so the application must make sure that one value doesn't overwrite another.

Because Windows Azure Storage is REST-based, the recommended way to deal with such concurrency issues is to use ETags. The first time the application creates a `Capacity` instance for a given date, the ETag will be null, but when an existing blob is downloaded from storage, it will have an ETag value available via

`CloudBlob.Properties.ETag`. When the application uploads the `Capacity` instance, it must correctly set the correct `AccessCondition` on a `BlobRequestOptions` instance:

```
options.AccessCondition = etag == null ?
    AccessCondition.IfNoneMatch("*") :
    AccessCondition.IfMatch(etag);
```

When the application creates a new instance of `Capacity`, the ETag is null and the `AccessCondition` must be set to `IfNoneMatch("*")`. This ensures that an exception will be thrown if the blob already exists. On the other hand, if the current write operation represents an update, the `AccessCondition` must be set to `IfMatch`, which ensures that an exception is thrown if the ETag in blob storage doesn't match the supplied ETag.

Optimistic concurrency based on ETags is an important tool in your toolbox, but you must explicitly enable it by supplying the appropriate `BlobRequestOptions`.

Data rendered on a screen is
always disconnected, which
means that it's stale from the
moment it's rendered.

If no exception is thrown while decrementing the capacity, the application can move on to the next step in **Figure 6**: writing the reservation to table storage. This follows roughly the same principles as decrementing the capacity, so I'll skip it here. The code is available in the accompanying download, but the main point is that, once again, the write operation should be idempotent.

The last step in the workflow is to raise an event that signifies the reservation was accepted. This is done by sending another asynchronous message via the Windows Azure queue. Any other background workers that care about this Domain Event can pick it up and handle it. A relevant action would be to send a confirmation e-mail to the user, but the application also needs to close the loop to the UI by updating the view data store.

Updating View Data

Events that happen during processing of a `Command` are sent as asynchronous messages via the `IChannel` interface. As an example, the `Consume` method in **Figure 8** raises a new `SoldOutEvent` if the capacity is reduced to zero. Other message handlers can subscribe to such events to properly update view data, as shown here:

```
public void Consume(SoldOutEvent message)
{
    this.writer.Disable(message.Date);
}
```

The injected writer implements the `Disable` method by updating an array of disabled dates for the appropriate month in blob storage:

```
public void Disable(DateTime date)
{
    var viewBlob = this.GetViewBlob(date);
    DateTime[] disabledDates = viewBlob.DownloadItem();
    viewBlob.Upload(disabledDates
        .Union(new[] { date }).ToArray());
}
```

Figure 7 The `Capacity.Reserve` Method

```
public Capacity Reserve(int quantity, Guid id)
{
    if (!this.CanReserve(quantity, id))
    {
        throw new ArgumentOutOfRangeException();
    }

    if (this.IsReplay(id))
    {
        return this;
    }

    return new Capacity(this.Remaining - quantity,
        this.acceptedReservations
            .Concat(new[] { id }).ToArray());
}
```

Figure 8 Decrementing Capacity and Committing to Storage

```
public void Consume(MakeReservationCommand message)
{
    var blob = this.GetCapacityBlob(message);
    var originalCapacity = blob.DownloadItem();

    var newCapacity = originalCapacity.Reserve(
        message.Quantity, message.Id);

    if (!newCapacity.Equals(originalCapacity))
    {
        blob.Upload(newCapacity);
        if (newCapacity.Remaining <= 0)
        {
            var e = new SoldOutEvent(message.Date);
            this.channel.Send(e);
        }
    }
}
```

This implementation simply downloads an array of disabled `DateTime` instances from blob storage, appends the new date to the array and uploads it again. Because the `Union` method is used, the operation is idempotent, and once more the `Upload` method encapsulates ETag-based optimistic concurrency.

Querying View Data

The UI can now query directly from the view data. This is an efficient operation because the data is static—no calculation is required. For example, to update the date picker in **Figure 2** with disabled dates, the date picker sends an AJAX request to the controller to get the array.

The Controller can simply handle the request like this:

```
public JsonResult DisabledDays(int year, int month)
{
    var data = this.monthReader.Read(year, month);
    return this.Json(data, JsonRequestBehavior.AllowGet);
}
```

The injected reader implements the `Read` method by reading the blob that the `SoldOutEvent` handler writes:

```
public IEnumerable<string> Read(int year, int month)
{
    DateTime[] disabledDates =
        this.GetViewBlob(year, month).DownloadItem();
    return (from d in disabledDates
        select d.ToString("yyyy.MM.dd"));
}
```

Thus the loop is closed. The user browses the site based on current view data and fills out a form to submit data that's handled via asynchronous messaging. Finally, view data is updated based on the Domain Events raised during the workflow.

Denormalizing Data

Summing up, most applications read data a lot more than they write data, so optimizing the read side enables scalability—particularly when data can be read from static resources such as blobs. Data rendered on a screen is always disconnected, which means that it's stale from the moment it's rendered. CQRS embraces this

staleness by disconnecting the reading and writing of data. The data being read doesn't have to come directly from the same data source as the data being written. Instead, data can be asynchronously transported from the store to which it's written to view-specific data stores where the cost of projecting and manipulating the data is paid only once.

With its built-in queues and scalable, denormalized data stores, Windows Azure is a great fit for this kind of architecture. Even though distributed transactions aren't supported, queues guarantee that messages are never lost, but are served at least once. To handle potential replays, all asynchronous write operations must be idempotent. For denormalized data such as blob and table storage, ETags must be used to implement optimistic concurrency. Using these simple techniques, Eventual Consistency can be ensured.

This article only scratches the surface of CQRS. If you want to know more about CQRS, the body of knowledge is currently spread out over a number of resources on the Internet; however, Rinat Abdullin's CQRS Starting Page at abdullin.com/cqrs is a good place to start. ■

MARK SEEMANN is the Windows Azure technical lead for Commentor A/S, a Danish consulting company based in Copenhagen. He is also the author of the book "Dependency Injection in .NET" (Manning Publications, 2011) and the creator of the open source project AutoFixture. His blog is available at blog.ploeh.dk.

THANKS to the following technical experts for reviewing this article:
Rinat Abdullin and Karsten Strøbæk

Data Quality Tools for .NET



**Address
Verification**



**Email
Validation**



**Name
Parse**



**Phone
Verification**



**Smart
Mover**

**Clean your database with
tools that make it easy.**

Request a free trial at
MelissaData.com/mynet or
Call 1-800-MELISSA (635-4772)

MELISSA DATA®
Your Partner in Data Quality

Parsing Log Files with F#, MapReduce and Windows Azure

Noah Gift

As a longtime Python programmer, I was intrigued by an interview with Don Syme, architect of the F# language. In the interview, Don mentioned that, “some people see [F#] as a sort of strongly typed Python, up to syntactic differences.” This struck me as something worth investigating further.

As it turns out, F# is a bold and exciting new programming language that’s still a secret to many developers. F# offers the same benefits in productivity that Ruby and Python programmers have enjoyed in recent years. Like Ruby and Python, F# is a high-level language with minimal syntax and elegance of expression. What makes F# truly unique is that it combines these pragmatic features with a sophisticated type-inference system and many of the best ideas of the functional programming world. This puts F# in a class with few peers.

This article discusses:

- Interactive coding with F#
- MapReduce in F#
- Moving to Windows Azure
- Deploying the solution

Technologies discussed:

F#, Windows Azure

Code download available at:

code.msdn.microsoft.com/mag201104Parsing

But a new high-productivity programming language isn’t the only interesting new technology available to you today.

Broad availability of cloud platforms such as Windows Azure makes distributed storage and computing resources available to single developers or enterprise-sized companies. Along with cloud storage have come helpful tools like the horizontally scalable MapReduce algorithm, which lets you rapidly write code that can quickly analyze and sort potentially gigantic data sets.

This approach is useful
for quick and dirty debugging
of an F# program.

Tools like this let you write a few lines of code, deploy it to the cloud and manipulate gigabytes of data in an afternoon of work. Amazing stuff.

In this article, I hope to share some of my excitement about F#, Windows Azure and MapReduce. I’ll pull together all of the ideas to show how you can use F# and the MapReduce algorithm to parse log files on Windows Azure. First, I’ll cover some prototyping techniques to make MapReduce programming less complex; then, I’ll take the results ... to the cloud.

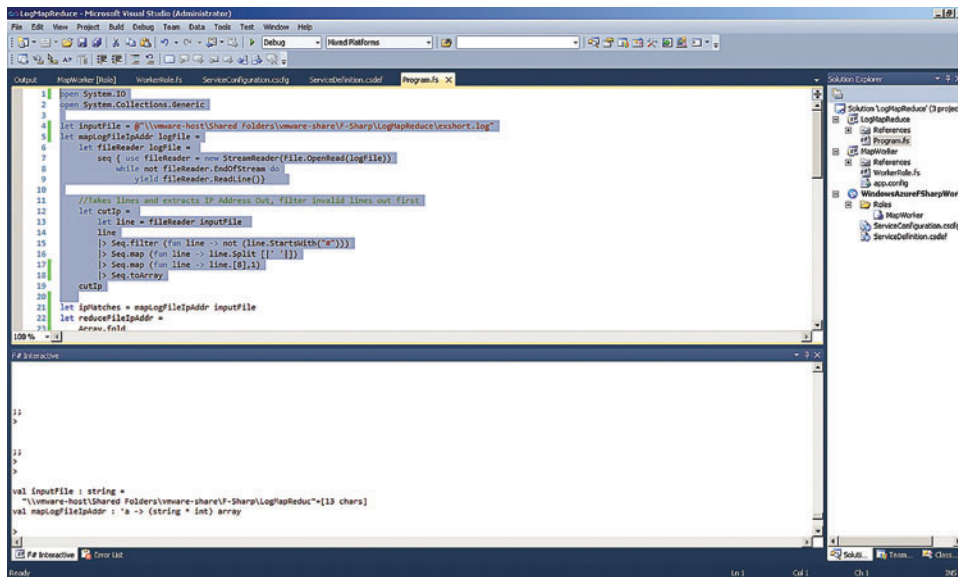


Figure 1 Using the F# Interactive Window

Hacking with F#

One of the new styles of working that's available to .NET programmers with F# is the Interactive workflow that many Perl, Python and Ruby programmers are accustomed to. This style of programming often uses an interactive coding environment like the Python shell itself, or a tool such as IPython, which provides readline completion. This allows a developer to import a class from a module, instantiate it, and then use tab completion to discover methods and data on the object.

A common method of interactive development that will be familiar to .NET developers is to simply write your code in Visual Studio and send snippets to the F# Interactive window for execution. Snippets are sent using the Alt+Enter key combination

of tab completion. You may find yourself using this technique as you get used to programming in a more interactive fashion.

You can also develop
interactively with F# by running
your code directly from
Windows PowerShell.

You can also develop interactively with F# by running your code directly from Windows PowerShell—in other words, by passing a script to fsi.exe (the F# Interactive Console executable) itself. One of the advantages of this approach is that it lets you prototype quick scripts and print the results to standard output. In addition, you can edit the code iteratively with a lightweight text editor, such as Notepad++. Figure 3 shows an example of a Map-Reduce script output that I'll be using throughout this article, as it's run from Windows PowerShell.

These different ways to write code all come in handy in helping you deal with complex algorithms, network programming and the cloud. You can write a quick-and-dirty prototype and run it from the command line to see if it gives you

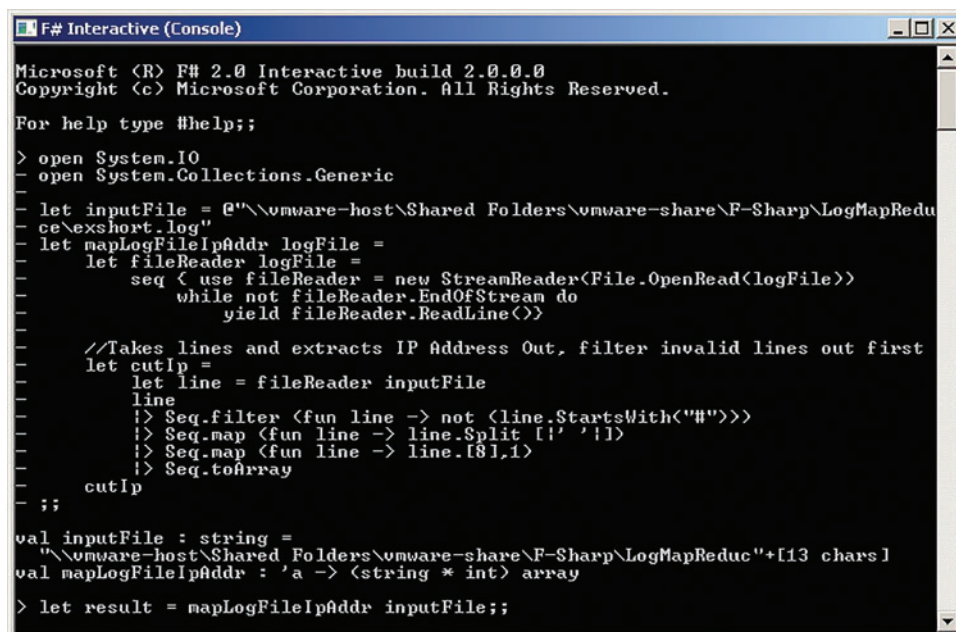


Figure 2 The F# Interactive Console

Figure 3 Running an F# Script in Windows PowerShell

```
PS C:\Users\Administrator\Desktop> & 'C:\Program Files (x86)\
FSharp-2.0.0.0\bin\fsi.exe' mapreduce.fsscript
192.168.1.1, 11
192.168.1.2, 9
192.168.1.3, 8
192.168.1.4, 7
192.168.1.5, 6
192.168.1.6, 5
192.168.1.7, 5
```

the results you expected. Then you can start building your larger project back in Visual Studio.

With this background information out of the way, let's dive into some actual code.

MapReduce-Style Log Parsing

In addition to the aforementioned benefits of interactive programming, F# code is also concise and powerful. The example in **Figure 4** is less than 50 lines of code, yet it contains all of the important parts of a MapReduce algorithm to calculate the top 10 IP addresses in a set of log files.

This standalone version, which will later become a network version, can be broken into three distinct phases: the map phase, the reduce phase and the display phase.

Phase 1 is the map phase. The function `mapLogFileIpAddr` takes a log file as a parameter. Defined inside this function is another function, `fileReader`, which uses a functional programming technique to lazily yield a line of text from the log file (although languages like C# and Python also have this). Next, the `cutIp` function parses each line of input, discards comment lines and then returns the IP address and an integer, 1.

This a powerful technique
for processing data.

To see why this is lazy, highlight the whole map block of code and run it in the F# Interactive window, along with the line:

```
let ipMatches = mapLogFileIpAddr inputFile
```

You'll see the following output:

```
val ipMatches : seq<string * int>
```

Note that nothing has actually been done yet, and the log file has not been read. The only thing that has happened is that an expression has been evaluated. By doing this, execution is delayed until it's actually needed, without pulling data into memory just for the sake of evaluating an expression. This is a powerful technique for processing data and it becomes especially noticeable when you have gigantic log files to parse that are in the gigabyte or terabyte range.

If you want to compare the difference and evaluate the code more eagerly, then simply add a line to the `cutIp` function, so that it looks like this:

```
let cutIp =
    let line = fileReader inputFile
    line
    |> Seq.filter (fun line -> not (line.StartsWith("#")))
    |> Seq.map (fun line -> line.Split [|' '|])
    |> Seq.map (fun line -> line.[8],1)
    |> Seq.toArray
cutIp
```

If you resend this code to the F# interpreter and you give it a large log file that contains several gigabytes of data, you might want to go get a cup of coffee, because your machine will busy reading in the whole file and generating key/value mappings for it in memory.

In the next section of the data pipeline, I take the output of the map result and fold the results of the sequence into an anonymous function that counts how many occurrences of IP addresses are in the sequence. It does this by continuously adding to the Map data structure via recursion. This style of programming can be hard to understand for developers new to functional programming, so you may want to embed print statements inside the anonymous function to see exactly what it does.

In an imperative programming style, you could accomplish the same thing by updating a mutable dictionary that holds each IP address as a key, looping over the sequence of IP addresses, and then updating the count of each value.

The final phase has nothing to do with the MapReduce algorithm, but is useful in hacking scripts during the prototyping phase. The results from the map phase are pipelined from a Map data structure to a Seq. The results are sorted and the top 10 results are printed out. Note that this data pipelining style enables the

Figure 4 MapReduce Algorithm to Parse a Log File

```
open System.IO
open System.Collections.Generic

// Map Phase
let inputFile = @"web.log"
let mapLogFileIpAddr logFile =
    let fileReader logFile =
        seq { use fileReader = new StreamReader(File.OpenRead(logFile))
              while not fileReader.EndOfStream do
                  yield fileReader.ReadLine() }

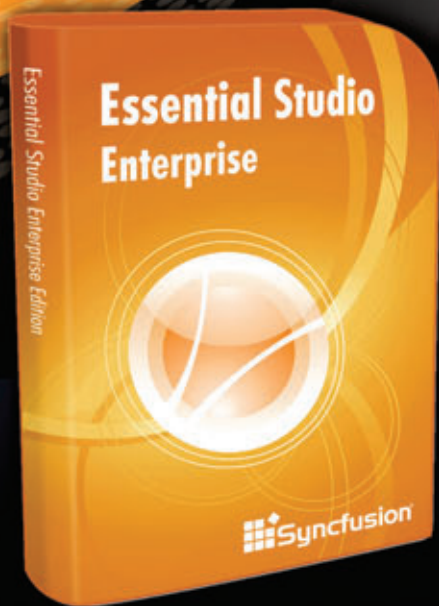
    // Takes lines and extracts IP Address Out,
    // filter invalid lines out first
    let cutIp =
        let line = fileReader inputFile
        line
        |> Seq.filter (fun line -> not (line.StartsWith("#")))
        |> Seq.map (fun line -> line.Split [|' '|])
        |> Seq.map (fun line -> line.[8],1)
        |> Seq.toArray
    cutIp

// Reduce Phase
let ipMatches = mapLogFileIpAddr inputFile
let reduceFileIpAddr =
    Array.fold
        (fun (acc : Map<string, int>) ((ipAddr, num) : string * int) ->
            if Map.containsKey ipAddr acc then
                let ipFreq = acc.[ipAddr]
                Map.add ipAddr (ipFreq + num) acc
            else
                Map.add ipAddr 1 acc)
        Map.empty
    ipMatches

// Display Top 10 IP Addresses
let topIpAddressOutput reduceOutput =
    let sortedResults =
        reduceFileIpAddr
        |> Map.toSeq
        |> Seq.sortBy (fun (ip, ipFreq) -> -ipFreq)
        |> Seq.take 10
    sortedResults
    |> Seq.iter(fun (ip, ipFreq) ->
        printfn "%s, %d" ip ipFreq);;

reduceFileIpAddr |> topIpAddressOutput
```

360° Application Transformation



Essential Studio Enterprise 2011 Vol. 1:

- Powerful RichText Editor control for Silverlight
- Support for ASP.NET MVC 3 and the new Razor view engine
- Support for Excel formula computation without using Excel



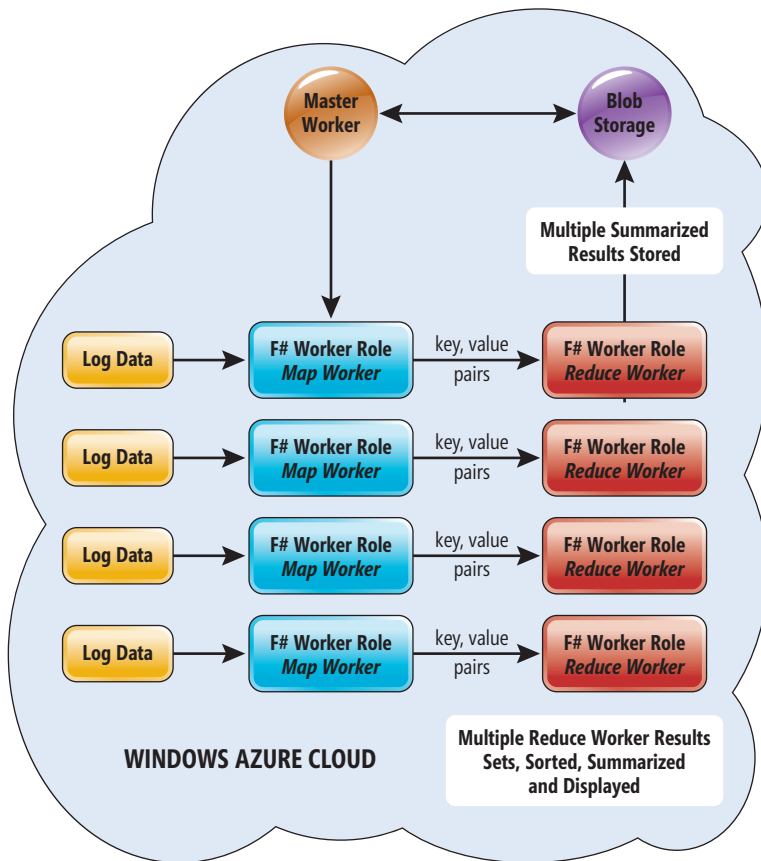


Figure 5 MapReduce Farm in Windows Azure

results of one operation to flow seamlessly into the next operation without a for loop in sight.

MapReduce Plus Windows Azure

With my proof-of-concept script completed—in less than 50 lines of code, remember—it's time move this to something resembling a

production environment. As an example, I'll move the example from desktop to Windows Azure.

As background, you might find it useful to look at the Windows Azure F# examples at code.msdn.microsoft.com/fsharpazure and to install the Windows Azure templates. Of particular interest is the webcrawler example in which an F# Worker Role uses both blob and queue storage endpoints. This would be a handy project to review as you further explore using F# with Windows Azure.

There are several ways to set up a MapReduce farm on Windows Azure.

I'm not going to go into too much detail about setting up a multinode MapReduce farm. Instead I'll cover it from a higher level. For specifics, see the *MSDN Magazine* article "Synchronizing Multiple Nodes in Windows Azure" by Josh Twist (msdn.microsoft.com/magazine/gg309174).

There are several ways to set up a MapReduce farm on Windows Azure. Figure 5 demonstrates one example using F# Worker Roles that would be equally divided between Map Workers and Reduce Workers.

Going back to script, this would be almost as simple as copying and pasting the map function to the Map Worker, and the reduce function to the Reduce Worker.

The MapReduce presentation by Jeff Dean and Sanjay Ghemawat is a great reference for further detail on the distributed algorithm and possible implementations (labs.google.com/papers/mapreduce-osdi04-slides/). In the Figure 5 example, though, it shows that several log

files are consumed in parallel by F# Worker Roles. They then return their output, consisting of IP address keys with a value of 1 via the Windows Azure AppFabric Service Bus to the Reduce Workers, or by writing it to disk.

Next, the Reduce Workers read this intermediate data and produce a summarized count of the key value pairs by writing it out to blob storage. Each Reduce Worker produces its own summarized report, which would need to be combined together before being sorted and displayed by the Master Worker.

Worker Role Creation and Publishing

With a prototype finished, and a high-level architecture planned, the next step is to create the necessary project in Visual Studio 2010 and publish it to Windows Azure.

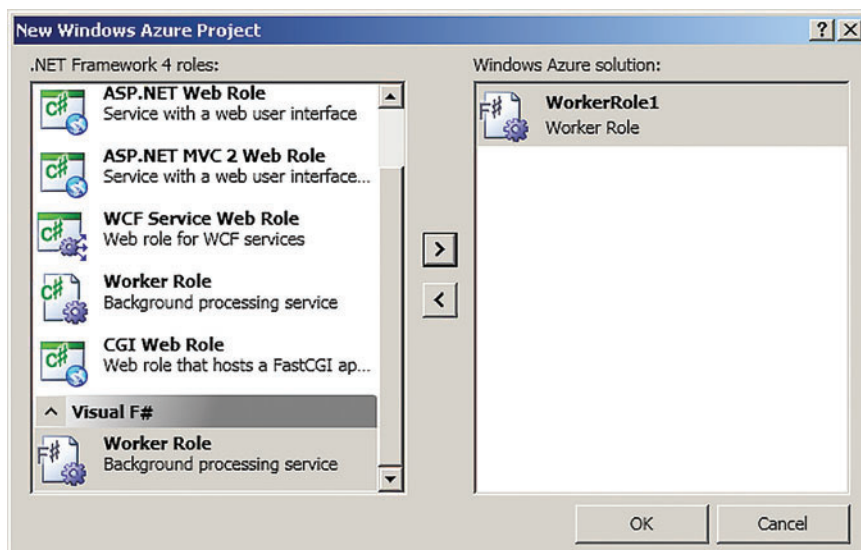


Figure 6 Creating an F# Worker Role



Now with charting!

SPREAD⁵

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

Spread 5 for Windows Forms 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



ActiveReports

Spread

DataDynamicsReports

ActiveAnalysis

WE ARE
SPREADSHEETS

GCPowerTools.com/Spreadsheets



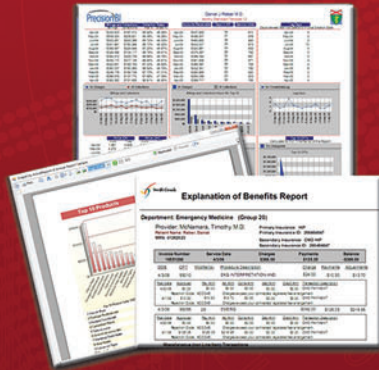


Now with Silverlight!

ACTIVE REPORTS⁶

*The de facto standard reporting tool
for Microsoft Visual Studio*

- 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
- Support for Azure



ActiveReports

Spread

DataDynamicsReports

ActiveAnalysis

WE ARE
REPORTING
GCPowerTools.com/Reporting

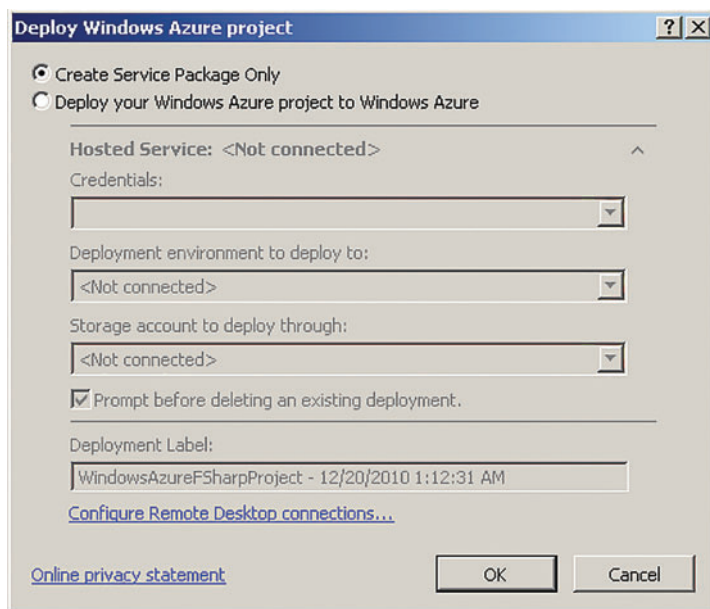


Figure 7 Publishing to Windows Azure

Creating an F# Worker Role isn't as straightforward as it could be, so let's walk through the steps involved. First, you'll need to download the Windows Azure F# templates mentioned earlier. Next, you'll need to create a Visual C# project for Windows Azure. I called mine AzureFSharpProject.

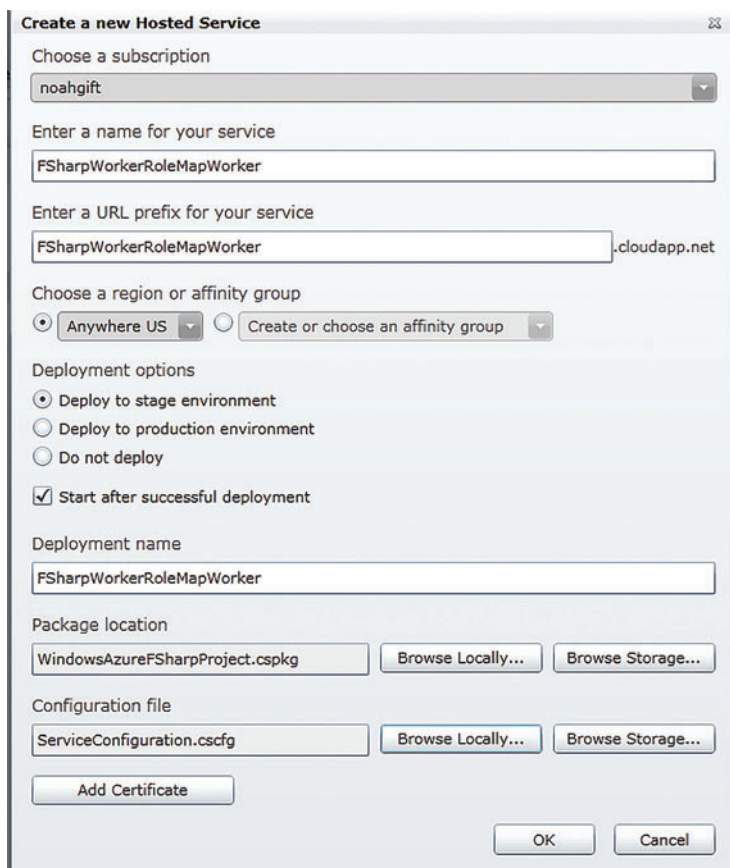


Figure 8 Configuring the New Worker Role

Next, you'll have the option to create an F# Worker Role as shown in Figure 6.

At that point, you can place your map function in your Map Worker role, or your reduce function in your Reduce Worker role. Then create further Worker Roles for additional Map Workers or Reduce Workers, depending on the scale of your data-crunching needs. The canonical reference to refer to is the Google MapReduce paper at labs.google.com/papers/mapreduce.html. It goes into further detail about the MapReduce architecture, caveats and use cases.

When you're ready to publish to Windows Azure, you can right-click on the project, select Publish, then Create Service Package Only, as shown in Figure 7.

Finally, you sign into the new Windows Azure management portal and use the interface to create your Worker Role (see Figure 8).

At this point, you can wire up your nodes in any way you see fit, and MapReduce your logs in the cloud. Of course, this same technique could be applied easily to data sources other than simple log files. The general outline of this F# MapReduce algorithm—along with the interactive tech-

niques I demonstrated in coding it—can be used for just about any parsing, mapping and reducing job.

Next Steps

F# is a powerful language that enables you to solve problems by both cranking out quick hacks and by building up more complex solutions from those hacks. In this article I used it to cut down the MapReduce algorithm into bite-sized morsels. This enabled me to demonstrate how only 50 lines of F# could then turn into a Windows Azure-based log analyzer.

Regarding the implementation of MapReduce on Windows Azure, you might want to take a look at two other interesting articles on the subject. First, take a look at "Building a Scalable, Multi-Tenant Application for Windows Azure" on MSDN for discussion of Worker Roles and MapReduce (msdn.microsoft.com/library/ff966483). Also, Juan G. Diaz has a blog entry, "Comparison of the use of Amazon's EC2 and Windows Azure, cloud computing and implementation of MapReduce," that's worthwhile reading (bit.ly/hBQFst).

If you haven't looked at F# yet, I hope this article encouraged you to give it a try. And if you're interested in hearing all of the Don Syme interview that turned me on to F#, head over to the Simple-Talk blog and give it a listen (bit.ly/el74i0). ■

NOAH GIFT is associate director of engineering at AT&T Interactive. He has a B.S. in Nutritional Science from Cal Poly San Luis Obispo, an M.S. in Computer Information Systems from California State University, Los Angeles, and is an MBA Candidate at UC Davis, specializing in Business Analytics, Finance and Entrepreneurship.

THANKS to the following technical experts for reviewing the article: Michael Bakkemo, Don Syme and Paras Wadehra

Visual Studio TFS Team Project and Collection Guidance

Willy-Peter Schaub and Mike Schimmel

In the *MSDN Magazine* article, “Visual Studio TFS Branching and Merging Guidance” (msdn.microsoft.com/magazine/gg598921), the Visual Studio ALM Rangers introduced a number of new branching scenarios and associated guidance to assist you in complex, real-world branching and merging environments, in order to improve the consistency and quality of solutions and the overall Application Lifecycle Management (ALM) process.

To recap, the Rangers are a group of experts who promote collaboration between the Visual Studio product groups, Microsoft Services and the Microsoft Most Valuable Professional (MVP) community by addressing missing functionality and removing adoption blockers.

In this article, the Rangers offer guidance for organizing and provisioning Team Foundation Server (TFS) Team Projects and Team Project Collections.

After reading this article, you should have a better understanding of:

- Team Project Collections and their benefits

This article discusses:

- Planning
- Visual Studio Team Projects
- Visual Studio Team Project Collections
- Team Project archiving strategies

Technologies discussed:

Visual Studio Team Foundation Server 2010

- Considerations for choosing to combine one or more Team Projects into a single Team Project Collection, or to keep them in separate Team Project Collections
- Considerations for organizing Team Projects and Team Project Collections to increase isolation or improve scalability
- How to archive one or more inactive Team Projects

This article will help you understand how Team Project and Team Project Collection organization is influenced by issues such as:

- Security and provisioning for TFS, Team Project Collections and Team Projects
- Selecting a Process Template
- Process Template customizations including customized work flows, Work Item Type customizations, custom reports, custom queries and customized process guidance
- Team organization
- Team Project organization including Areas and Iterations
- Project management considerations including project milestones and schedules

Planning

Visual Studio TFS planning usually starts with the recommended infrastructure and the structuring of the Team Projects and Team Project Collections to ensure an effective and scalable ALM system for all stakeholders.

The Visual Studio 2010 Quick Reference Guidance (vs2010quickref.codeplex.com) and Visual Studio 2010 and TFS 2010 Virtual Machine

(VM) Factory (rangersvsvmfactory.codeplex.com) Rangers guidance projects provide concepts, guidance and quick reference posters to support capacity planning and help answer the question of whether infrastructure should be physical or virtual, or both.

Although you typically plan and define a Team Project Collection before a Team Project in a TFS 2010 environment, we'll cover the Team Project first.

Visual Studio Team Projects

In TFS 2005 and TFS 2008, a single TFS server could host one or more Team Projects. Each Team Project is essentially a container for artifacts, including source code (organized into folders, branched folders and branches) and containing one or more Visual Studio solutions, Team Build configuration files, Team Load Test Agents, an optional SharePoint repository containing the pertinent documents for the project and security provisioning used by a team to track and perform a set of work as governed by a process template. The Team Project should not be confused with a Visual Studio .NET Project, which contains all of the build and configuration settings required to generate a Microsoft .NET Framework assembly; a Visual Studio .NET Solution, which contains one or more Visual Studio .NET projects and defines project dependencies, build process and order; or a project initiative, which is a scheduled initiative for building something from a set of requirements.

For more information on creating and managing Team Projects, you can refer to the "Creating and Managing Team Projects" MSDN Library page (bit.ly/eCP0yX).

Let's discuss considerations for organizing Project Initiatives into Team Projects. Team Projects often contain the largest unit of work associated with developing a single product or product line. For example, at Microsoft, Visual Studio and Office are two product lines each contained within its own separate Team Project (see **Figure 1**), because the development of these two product lines proceeds along completely different timelines with separate milestones and release schedules.

It's important to take note of provisioning considerations and security isolation. One of the most time-consuming aspects of creating a new Team Project is the effort required to provision the project for use by one or more teams, primarily by defining the security users, groups and permissions that control access to a Team Project and its artifacts. This provisioning effort needs to be balanced against the benefits of properly defining security at the right level of granularity so as to allow members of the team to be able to do what they *need* to do. At the same time, proper security provisioning protects a Team Project and its assets from inadvertent or intentional damage done by people who aren't supposed to be performing certain tasks.

For product lines with different milestones and release schedules, such as Visual Studio and Office, it

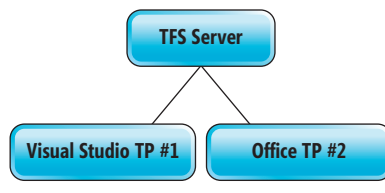


Figure 1 Visual Studio and Office Team Projects

makes sense to organize each product line into a separate Team Project for security isolation. The development teams associated with each product line are likely to be completely separate and unlikely to require and be granted contribution or build permissions to both team environments.

For project initiatives with different methodologies—for example, where one team chooses to use the Microsoft Solutions Framework (MSF) for Agile Software Development version 5.0 and the second team chooses to use the MSF for CMMI Process Improvement version 5.0 Process Template—separate Team Projects are required because a given Team Project has one, and only one, Process Template associated with it.

It's important to take note of provisioning considerations and security isolation.

For project initiatives that require unique definitions for Areas and Iterations, we suggest separate Team Projects, as a given Team Project defines a single Area and Iteration hierarchy. Alternatively, a single Team Project can use Areas to organize multiple feature teams (see **Figure 2**) that share the same Iterations (see **Figure 3**). Also see "Project of Projects with Team Foundation Server 2010," by Martin Hinshelwood, at bit.ly/hSnHGw, for a discussion on a scenario using Areas instead of having many small Team Projects.

Version Control Check-out settings (such as exclusive Check-out, Check-in policies and Check-in notes) are defined for a Team Project, and these settings aren't shared across Team Project boundaries. If

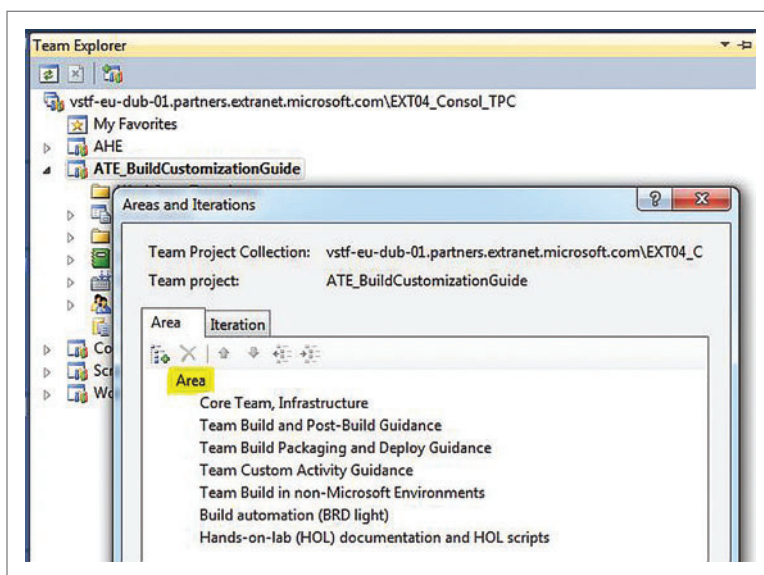


Figure 2 Team Project Using Areas to Organize Separate Feature Teams

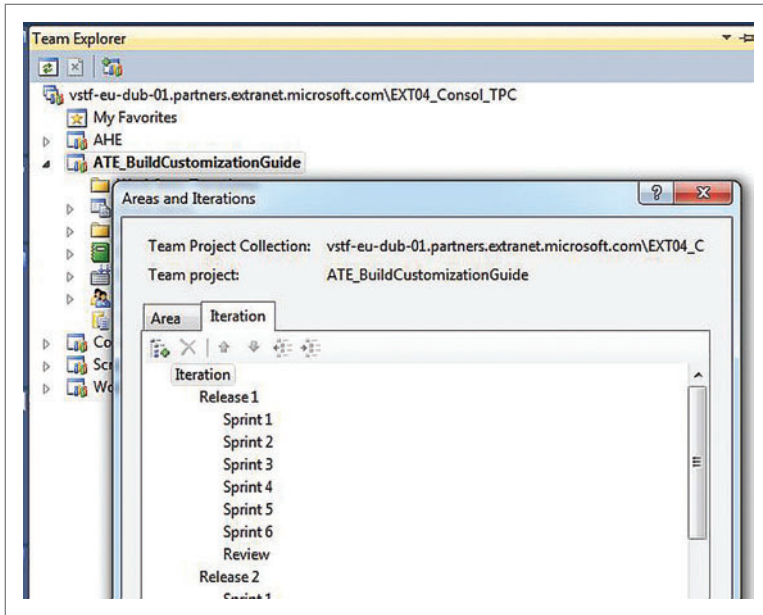


Figure 3 Team Project with Multiple Feature Teams Sharing the Iteration Hierarchy

separate project initiatives require different Source Control settings, they must be associated with separate Team Projects.

Process Template Customizations include customized or custom Work Flows, Work Item Types (WITs), Reports and Queries. You can customize a process template used to create new Team Projects or customize the specific Process Template used by a Team Project, whereby the latter is not shared across Team Projects.

Sharing of Team Project Artifacts is typically achieved by branching from one Team Project to another. In our previous article, we discussed various approaches for sharing common code, most of which involve branching.

As you can see, there are a number of considerations for deciding whether separate projects or project initiatives can share the same Team Project or must be associated with separate Team Projects. You should understand the various considerations and make Team Project organization decisions that are best for your organization.

Visual Studio Team Project Collections

Although Team Projects are somewhat independent of one another, in TFS 2005 and TFS 2008, certain maintenance activities such as consolidating multiple TFS servers into a single server were difficult. Further, separate business units within an organization could only achieve organizational isolation by implementing two or more TFS servers within the organization, increasing the cost of infrastructure, maintenance and overall complexity.

Visual Studio 2010 introduced Team Project Collections. Each TFS Server can have one or more Team Project Collections. In turn, a Team Project Collection contains one or more Team Projects. A Team Project Collection is the basic unit of recovery for TFS. From a backup and restore perspective, a Team Project Collection is similar to a SharePoint Site Collection.

The Visual Studio 2010 Quick Reference Guidance project (vs2010quickref.codeplex.com) provides a quick reference poster to assist

you with the planning of the new Team Project Collection feature, as well as Team Projects. Team Project Collections provide a more scalable deployment for TFS servers. One Team Project Collection in TFS 2010 is the rough equivalent of one TFS Server in TFS 2005 or TFS 2008. For more information on creating and managing Team Project Collections, refer to msdn.microsoft.com/library/dd236915.

Considerations for isolating Team Projects into separate Team Project Collections include scalability, backup, recovery, security isolation and sharing of information among Team Projects.

Scalability of TFS servers is supported by the ability to load balance Team Project Collections across physical SQL Servers and SQL Server instances, taking advantage of scalability and load-balancing infrastructure that's typically associated with database environments. If you have the ability to load balance across SQL Servers, you may benefit from splitting Team Projects across multiple Team Project Collections.

As noted earlier, the basic unit of recovery for TFS is the Team Project Collection. Team Projects can't be

individually backed up or restored. If you require granular backup and recovery capabilities (for example, if you don't want to recover more than a single Team Project), it may benefit your organization to isolate Team Projects into separate Team Project Collections for the purpose of backup and recovery.

Security provisioning for Team Project Collections, if administered properly and with the correct degree of control and granularity, may be time consuming. As you add new Team Project Collections, you must consider the initial and ongoing effort to provision each of these collections. If the project teams for the Team Projects being managed on a TFS server have different security requirements, it may benefit you to isolate Team Projects into separate Team Project Collections for security purposes.

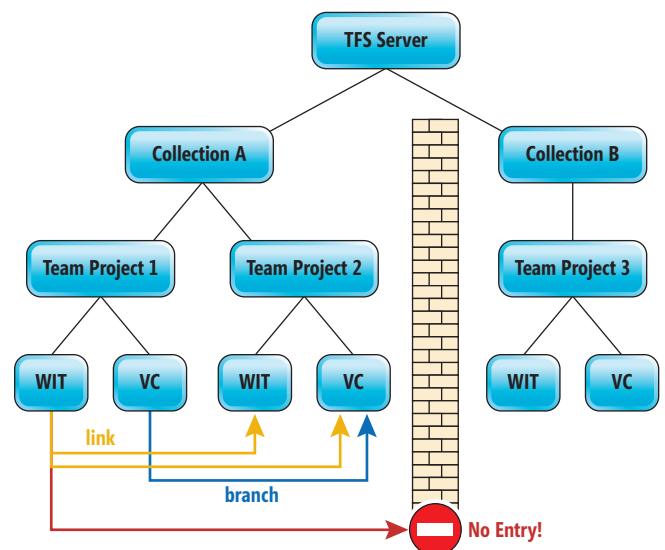
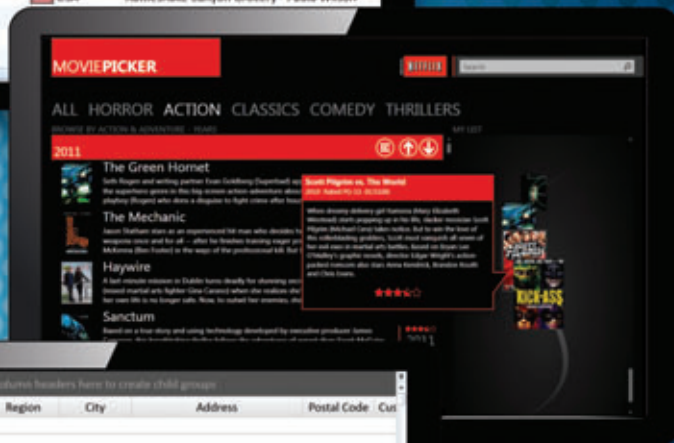


Figure 4 Team Project Collection Sharing and Isolation Boundaries

THESE GUYS STAND ON THEIR OWN.

That's because we focus on the most important controls, not dozens of generic, bundled ones.

ID	Employee	Country	Customer
USA (122 items)			
Albuquerque (18 items)			
11,077	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
11,000	Fuller, Andrew	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,988	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,889	Dodsworth, Anne	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,852	Callahan, Laura	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,820	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,761	Buchanan, Steven	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,598	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,569	Buchanan, Steven	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,564	Peacock, Margaret	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,479	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,401	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,346	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson



Order ID	Country	Region	City	Address	Postal Code	Customer
USA						
CO						
FL						
10310	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
10317	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
10805	USA	FL	Miami	89 Jefferson Way Suite 2	97201	77
10867	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
10883	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
10992	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11018	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
11169	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11172	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11179	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11406	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11524	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11729	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
11854	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11880	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48



XCEED
DataGrid
for WPF

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



XCEED
DataGrid
for Silverlight

The only Silverlight datagrid on the market with fast remote data retrieval and a completely fluid UI!



XCEED
Ultimate ListBox
for Silverlight

The same data virtualization and smooth-scrolling as our Silverlight datagrid, packed in the streamlined format of a listbox.

Try them live at
xceed.com

XCEED
MULTI-TALENTED COMPONENTS

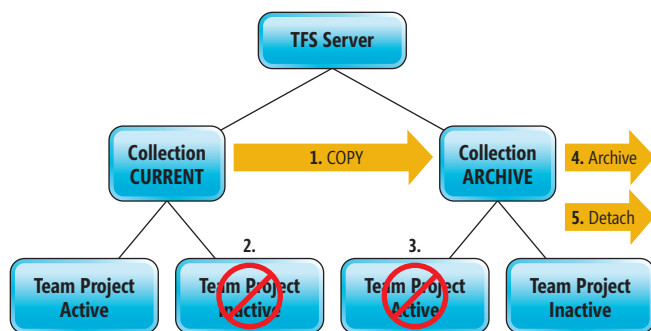


Figure 5 Possible Archive Strategy

On the other hand, if the project teams for the Team Projects being managed on a TFS server don't require security isolation, it may benefit your organization to have their Team Projects within the same Team Project Collection (see Figure 4).

While sharing of artifacts (such as source code files) can be done between Team Projects in the same Team Project Collection, they can't be shared across collection boundaries (see Figure 4). If two Team Projects must share artifacts, they must be contained within the same Team Project Collection.

Discussions about the topics of using Team Project Collections for sharing and isolation of source code, and branching and merging, are not within the scope of this article. We recommend you refer to tfsbranchingguideiii.codeplex.com for associated discussions and information that will be included in forthcoming guidance.

Team Project Archiving Strategies

Periodic maintenance is essential, as a TFS environment can never be installed on unlimited physical resources. Administrators need to plan for periodic maintenance to archive completed project data and relieve pressure on the server before it becomes a performance issue for development teams.

The Rangers Upgrade Guidance (vs2010upgradeguide.codeplex.com) defines a number of possible strategies as part of the upgrade guidance, similar to the procedure that the Microsoft Consulting Services has developed (see Figure 5):

1. Start by making a copy of the Team Project Collection.
2. Delete active Team Projects from the newly cloned archival Team Project Collection (using the `TFSDelProject` command-line utility).
3. Delete the archived projects from the original (active) Team Project Collection.
4. The new Team Project Collection can then be stored off to external media, such as tape or flash media, and then removed from the hardware. If an audit is made of the system, the archive media can easily be restored for that purpose.
5. Detach the new Team Project Collection in order to easily get it back later.

In concept, this seems like a trivial strategy, but it really requires a policy for determining which Team Projects can be archived and which can't. Take special note that source code branches are removed from the system when a `TFSDelProject` action is performed, and this isn't an undoable event.

Following are suggestions for an archiving policy:

- Establish a project close-out policy for development teams to cleanse project data and store it. Source code is not the only material that needs to be saved. Project requirements are interesting, but they're probably not in a format that can be reused or from which enhancements can be made. Functional specifications need to be blended with previous ones to reflect the state of the product as it stands in production at the completion of the project. The requirements for that project can then be archived with no worry of losing data. Work Items can't be merged from one Team Project to another Team Project like source code can, so decisions about how to store completed work items after a project is completed will be necessary.
- Send a standard request to all teams notifying them of a pending archival action prior to the event, listing all Team Projects targeted for the archiving.
- Establish a milestone folder in each Team Project that serves as a container for a completed project initiative's final project data, including requirements listings, final project reports and any documentation that's stored under normal methodology policy for project closeout.

Essentially, all project data—not only source code—needs to be preserved in order for an archive to be successful.

Splitting Team Project Collections is needed to react to a possible separation of a business unit into two or more managed entities. In this scenario, a similar approach to that used for archiving Team Projects is employed, although none of the Team Projects will be archived. The second Team Project Collection will have to be treated like the original in that it will now require periodic maintenance separate from the original Team Project Collection.

Moving Team Projects between Team Project Collections isn't easy, and once a collection is split, it can't simply be remerged, as only new Team Projects can be added to a collection. In order to merge Team Project Collections, custom code is required using the TFS API to copy project data from one collection to another.

Although not recommended, TFS Integration Tools can be used to merge Team Projects into Team Project Collections—see the TFS Integration Tools documentation (bit.ly/9tHwDg).

In a future article, we'll investigate how the Rangers are overcoming and managing the challenges of distributed Agile team management using the Visual Studio ALM tooling. ■

WILLY-PETER SCHAUB is a senior program manager with the Visual Studio ALM Rangers at the Microsoft Canada Development Center. Since the mid-'80s, he's been striving for simplicity and maintainability in software engineering. His blog is at blogs.msdn.com/b/willy-peter_schaub and you can follow him on Twitter at twitter.com/wpschaub.

MIKE SCHIMMEL is an ALM solution architect with the Microsoft Consulting Services U.S. Architects and a core member of the Visual Studio ALM Rangers. He has nearly 25 years of experience studying and practicing ALM, from research and consulting to competitive product sales and service delivery.

THANKS to the following technical experts for reviewing this article:

Bill Essary, Bill Heys, Martin Hinshelwood, Bijan Javidi and Mario Rodriguez



Powered by
Microsoft Silverlight®



Silverlight Imaging SDK

- Load, display and save 100+ formats including PDF, JPEG2000 and TIFF
- Pure managed Silverlight binaries for Silverlight 3, 4 and Windows Phone
- Multi-touch support for Silverlight
- Annotation and markup
- Load, view, process and save DICOM
- Apply visual effects with pixel shaders

The LEADTOOLS Silverlight SDK gives Web and Windows Phone developers the ability to load, display, process, convert and save many different image formats that are not intrinsic to the Silverlight framework. Because of the 100% managed Silverlight binaries, images can be processed and annotated directly within the Silverlight application without the need for external server calls.

The SDK comes with several viewing components including Image Viewer, Image List and Pan Window. The viewer is equipped with automated interactive tools including scroll, pan, scale, zoom and magnifying glass.

Adding Annotations and Markup to your Silverlight application is a cinch with LEADTOOLS fully automated annotations. Load, save, create and edit popular annotation types like line, ruler, polygon, curve, note, highlight, redaction and more.

Create highly interactive medical imaging applications with LEADTOOLS support for DICOM. Load, save and modify tags from DICOM data sets and display 12/16 bpp grayscale images with Window Leveling. LEADTOOLS is currently the only Windows Phone imaging toolkit on the market with support for DICOM data sets.

LEADTOOLS includes sample projects to demonstrate all of the SDK's capabilities including the display of over 100 image formats and incorporation of more than 200 image processing functions.

Free 60 Day Evaluation! www.leadtools.com/sd
(800) 637-1835



Use Bee Colony Algorithms to Solve Impossible Problems

James McCaffrey

Simulated Bee Colony (SBC) algorithms model the behavior of honey bees and can be used to find solutions to difficult or impossible combinatorial problems. In this article I'll explain what exactly SBC algorithms are, describe the types of problems that can be solved using SBC algorithms, and present a complete end-to-end example that uses an SBC algorithm to solve the Traveling Salesman Problem.

A good way for you to get a feel for SBC algorithms and see where I'm headed in this article is to examine the demo program shown running in **Figure 1**. The demo program uses an SBC algorithm to analyze a set of 20 cities (labeled A through T) and find the shortest path that visits every city exactly once. The city data is artificially constructed so that the best path starts at city A and continues through city T, in order, and the best path has a length of 19.0 units.

Behind the scenes the SBC algorithm instantiates a hive of 100 simulated bees, each of which has a random potential solution. Initially, the best of the random solutions has a path length of 95.0 units. The SBC algorithm enters a processing loop, indicated by the text-based progress bar, which simulates the behavior of common honey bees foraging for food. At the end of the SBC processing loop,

the best solution found has 16 correct city positions out of 20, and has a path length of 26.5—close to, but not quite, the optimal solution.

SBC algorithms are often called meta-heuristics because they provide a general framework and set of guidelines for creating a problem solution rather than providing a highly detailed solution prescription. This article presents an example of using an SBC to solve a specific problem. Once you understand how an SBC can be used to solve one problem, you can adapt the SBC algorithm presented here to solve your own problems. As this article will demonstrate, SBC algorithms are best suited for solving complex combinatorial problems that have no practical deterministic solutions.

This article assumes you have intermediate-level programming skills. The example in this article is coded using C#, but all the code has been written so that it can be easily refactored to other programming languages. I think you'll find this article quite interesting and the ability to use SBC algorithms a useful addition to your personal skill set.

About the Bees

Common honey bees such as *Apis mellifera* assume different roles within their colony over time. A typical hive may have 5,000 to 20,000 individual bees. Mature bees (20 to 40 days old) usually become foragers. Foraging bees typically occupy one of three roles: active foragers, scout foragers and inactive foragers.

Active foraging bees travel to a food source, examine neighbor food sources, gather food and return to the hive.

Scout bees investigate the area surrounding the hive, often a region of up to 50 square miles, looking for attractive new food sources. Roughly 10 percent of foraging bees in a hive are employed as scouts.

At any given time some of the foraging bees are inactive. These inactive foragers wait near the hive entrance. When active foragers and scouts return to the hive, depending on the quality of the food source they've just visited, they may perform a waggle dance to the waiting

This article discusses:

- The Traveling Salesman Problem
- A Simulated Bee Colony algorithm
- Implementing Bees and Hives
- Three essential SBC methods

Technologies discussed:

Algorithms, Honey Bees

Code download available at:

code.msdn.microsoft.com/mag201104BeeColony

```

file:///C:/SimulatedBeeColony/bin/Debug/SimulatedBeeColony.EXE
Begin Simulated Bee Colony algorithm demo
Loading cities data for SBC Traveling Salesman Problem analysis
Cities: A B C D E F G H I J K L M N O P Q R S T
Number of cities = 20
Number of possible paths = 2,432,902,008,176,640,000
Best possible solution (shortest path) length = 19.0000

Initial random hive
Best path found: B->A->D->C->G->H->K->T->S->M->J->L->F->I->Q->P->N->E->R->O
Path quality: 95.5000

Entering SBC Traveling Salesman Problem algorithm main processing loop
Progress: !=====!
          ~~~~~

Final hive
Best path found: A->B->C->D->E->F->G->H->I->J->L->M->N->O->P->Q->R->S->T
Path quality: 26.5000

End Simulated Bee Colony demo

```

Figure 1 Simulated Bee Colony Demo

inactive bees. There's strong evidence that this waggle dance conveys information to the inactive bees about the location and quality of the food source. Inactive foragers receive this food source information from the waggle dance and may become active foragers.

In general, an active foraging bee continues gathering food from a particular food source until that food source is exhausted, at which time the bee becomes an inactive forager.

The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is one of the most widely studied problems in computer science research. There are many variations of the TSP but, informally, the problem is to find the shortest path that visits every city in a given set of cities exactly once.

If you look at **Figure 1** you can see the SBC demo program uses a set of 20 cities that are arbitrarily labeled A through T. A valid path consists of an ordered set of the 20 city labels where each city occurs exactly once. Therefore there are a total of $20! = 2,432,902,008,176,640,000$ possible paths.

Behind the scenes there's a distance value associated with each pair of cities. For simplicity, if city $c_1 < c_2$ the distance between c_1 and c_2 is just 1.0 times the ordinal distance between the city labels. If $c_1 > c_2$ the distance is 1.5 times the ordinal distance between c_1 and c_2 . So the distance from A to B is 1.0 arbitrary units, the distance from B to A is 1.5 units, the distance from A to C is 2.0 units and so on. Therefore, the best path that visits every city exactly once is A -> B -> C -> ... -> T and the best (shortest) path length is $(20-1) * 1.0 = 19.0$.

In most TSP scenarios, the distances between cities wouldn't be artificially computed. Instead, the distances would likely be stored in some sort of lookup data structure. Some variations of the TSP assume that every city is connected to every other city and some problems assume the cities are not fully connected. Additionally, some TSP variations assume that the distance from any city c_1 to city c_2 is the same as the distance from city c_2 to c_1 , and some problem variations don't make this bidirectional assumption.

Except in trivial situations, a brute force approach to finding the shortest path is not feasible. For example, with 20 cities, even

if you could evaluate 1 million paths per second, examining all $20!$ possible paths would require more than 77,000 years. This kind of extremely difficult combinatorial optimization problem is the type of problem that SBC algorithms are well suited to handle.

The dummy TSP problem is implemented in a class named `CitiesData`, shown in **Figure 2**. The complete source code for the SBC demo program is available at code.msdn.microsoft.com/mag0411BeeColony.

The definition of some class or data structure that represents your particular problem will be different from the one shown here. However, as a general rule of thumb, problems that

are well-suited for solution using an SBC algorithm usually can be represented as a non-numeric array, a non-numeric matrix or a non-numeric jagged array of arrays.

The `CitiesData` constructor accepts a value for the number of cities and assigns the first city a label of A, the second city a label of B and so on.

The `Distance` method is defined in an unidirectional way as I previously described and assumes that any city can be reached by any other city.

The `ShortestPathLength` method returns the optimal path length given the `Distance` definition. In most SBC cases you won't have

Figure 2 The `CitiesData` Class Definition

```

class CitiesData {
    public char[] cities;

    public CitiesData(int numberCities) {
        this.cities = new char[numberCities];
        this.cities[0] = 'A';
        for (int i = 1; i < this.cities.Length; ++i)
            this.cities[i] = (char)(this.cities[i - 1] + 1);
    }

    public double Distance(char firstCity, char secondCity) {
        if (firstCity < secondCity)
            return 1.0 * ((int)secondCity - (int)firstCity);
        else
            return 1.5 * ((int)firstCity - (int)secondCity);
    }

    public double ShortestPathLength() {
        return 1.0 * (this.cities.Length - 1);
    }

    public long NumberOfPossiblePaths() {
        long n = this.cities.Length;
        long answer = 1;
        for (int i = 1; i <= n; ++i)
            checked { answer *= i; }
        return answer;
    }

    public override string ToString() {
        string s = "";
        s += "Cities: ";
        for (int i = 0; i < this.cities.Length; ++i)
            s += this.cities[i] + " ";
        return s;
    }
}

```


Figure 3 Overall Program Structure

```
using System;
namespace SimulatedBeeColony {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("\nBegin Simulated Bee Colony algorithm demo\n");
            . . .
            Console.WriteLine("End Simulated Bee Colony demo");
        }
    }

    class Hive {
        public class Bee {
            . . .
        }

        // Hive data fields here

        public override string ToString() { . . . }

        public Hive(int totalNumberBees, int numberInactive,
            int numberActive, int numberScout, int maxNumberVisits,
            int maxNumberCycles, CitiesData citiesData) {
            . . .
        }

        public char[] GenerateRandomMemoryMatrix() { . . . }

        public char[] GenerateNeighborMemoryMatrix(char[] memoryMatrix) { . . . }

        public double MeasureOfQuality(char[] memoryMatrix) { . . . }

        public void Solve() { . . . }

        private void ProcessActiveBee(int i) { . . . }

        private void ProcessScoutBee(int i) { . . . }

        private void ProcessInactiveBee(int i) { . . . }

        private void DoWaggleDance(int i) { . . . }
    }

    class CitiesData {
        . . .
    }
} // ns
```

the information necessary to implement a method that returns the optimal solution.

The `NumberOfPossiblePaths` method just computes $n!$ where n is the number of cities. In some TSP scenarios the number of possible paths is $n-1!$ (if the start city doesn't matter) and in some scenarios the number of possible paths is $n/2!$ (if the direction of the path doesn't matter).

The `ToString` method uses string concatenation rather than the more efficient `StringBuilder` class to assemble a string with descriptive data. String concatenation is used in order to simplify refactoring to other programming languages.

In this article, to keep the code relatively short and clean, I take shortcuts you wouldn't use in production code, such as removing most error-checking. For example, method `NumberOfPossiblePaths` doesn't deal with numeric overflow if the result is greater than `long.MaxValue`.

SBC Program Structure

The SBC algorithm shown in **Figure 1** is implemented as a C# console application. The overall structure of the program is listed in **Figure 3**. Notice that the SBC program structure is relatively simple and uses only the basic `System` namespace. There are three classes: the `Program`

class, which houses a single `Main` method; the `Hive` class, which houses all the SBC algorithm logic; and the `CitiesData` class presented in the previous section of this article. The `Hive` class is generically named `Hive` rather than given a more specific name like `TravelingSalesmanHive`, even though SBC algorithm implementations are heavily dependent upon the particular problem they're designed to solve.

The `Main` method is quite simple. After displaying a begin message the `CitiesData` object is instantiated:

```
Console.WriteLine(
    "Loading cities data for SBC Traveling Salesman Problem analysis");
CitiesData citiesData = new CitiesData(20);
Console.WriteLine(citiesData.ToString());
Console.WriteLine("Number of cities = " + citiesData.cities.Length);
Console.WriteLine("Number of possible paths = " +
    citiesData.NumberOfPossiblePaths().ToString("#,###"));
Console.WriteLine("Best possible solution (shortest path) length = " +
    citiesData.ShortestPathLength().ToString("F4"));
```

In many SBC scenarios your problem data will reside in external storage such as a text file or a SQL databases, and you'll load problem data via some load constructor or load method along the lines of `myProblemData.Load(dataFile)`.

Next, the `Hive` constructor is prepared and called:

```
int totalNumberBees = 100;
int numberInactive = 20;
int numberActive = 50;
int numberScout = 30;
int maxNumberVisits = 100;
int maxNumberCycles = 3460;

Hive hive = new TravelingSalesmanHive(totalNumberBees,
    numberInactive, numberActive, numberScout, maxNumberVisits,
    maxNumberCycles, citiesData);
```

As you'll see in more detail in the following sections of this article, an SBC algorithm uses three types of bees: active, inactive and scout. Although the counts of each of these types of bees can be hard-coded, in most scenarios it's better to pass these counts in as parameters to the `Hive` constructor so the algorithm can be more easily tuned for performance.

The value of the `totalNumberBees` variable could be derived from the other three variables, but the extra variable improves code readability here. The total number of bees will depend on your particular problem. More bees are better but slow the program's execution. In terms of ratios, there's some research that suggests the best percentages of active, inactive and scout bees are often roughly 75 percent, 10 percent and 15 percent, respectively.

The 100 value used for the `maxNumberVisits` variable will be explained shortly, but it's related to the number of neighbor solutions there are relative to a given solution.

The `maxNumberCycles` variable is used to control how many times the `Solve` routine will iterate; this is necessary because in SBC algorithm scenarios you usually don't know when you've reached an optimal solution—if you know the optimal solution you really don't have a problem to solve. In this case, the value of `maxNumberCycles` was limited to 3,460 only so that the SBC algorithm did not produce a perfect result. The point here is that although SBC algorithms may produce an optimal result, you usually have no way of knowing this and so you must be willing to accept a "good" result.

When the constructor executes it creates a collection of bees, each of which has a random solution. The `Hive` object tracks the overall best (shortest) path found by any of the bees in the hive and the best solution's associated path length.

Reporting. Defined.

The XtraReports Suite

WinForms



ASP.NET AJAX



Silverlight



WPF



Experience the power of the **XtraReports Suite** today.
Download your FREE trial – www.DevExpress.com/FreeEval.



All trademarks and registered trademarks are the property of their respective owners.

Figure 4 Bee Class Definition

```
public class Bee {
    public int status;
    public char[] memoryMatrix;
    public double measureOfQuality;
    public int numberOfVisits;

    public Bee(int status, char[] memoryMatrix,
        double measureOfQuality, int numberOfVisits) {
        this.status = status;
        this.memoryMatrix = new char[memoryMatrix.Length];
        Array.Copy(memoryMatrix, this.memoryMatrix, memoryMatrix.Length);
        this.measureOfQuality = measureOfQuality;
        this.numberOfVisits = numberOfVisits;
    }

    public override string ToString() {
        string s = "";
        s += "Status = " + this.status + "\n";
        s += "Memory = " + "\n";
        for (int i = 0; i < this.memoryMatrix.Length-1; ++i)
            s += this.memoryMatrix[i] + "->";
        s += this.memoryMatrix[this.memoryMatrix.Length-1] + "\n";
        s += "Quality = " + this.measureOfQuality.ToString("F4");
        s += "Number visits = " + this.numberOfVisits;
        return s;
    }
}
```

After calling the Hive constructor, the Main method finishes up by using the ToString method to display the initial, randomly generated Hive values, calling the Solve method with a parameter indicating that a text-based progress bar should be printed, and then displaying the best path and associated path length found:

```
...
Console.WriteLine("\nInitial random hive");
Console.WriteLine(hive);

bool doProgressBar = true;
hive.Solve(doProgressBar);

Console.WriteLine("\nFinal hive");
Console.WriteLine(hive);
Console.WriteLine("End Simulated Bee Colony demo");
}
```

The Bee Class

As shown in **Figure 3**, the Hive class contains a nested Bee class definition. The details of the Bee definition are listed in **Figure 4**.

The Bee class has three data fields common to all SBC implementations and one problem-specific data field. The problem-specific field is named memoryMatrix. Every SBC implementation must have some way to represent a solution. In the case of the TSP in this article, a solution can be represented by an array of type char. Let me emphasize that the object that represents a solution is highly problem-dependent and every problem must be analyzed separately to produce a meaningful solution representation.

The field named status holds an int value that indicates the Bee object's type: 0 for inactive, 1 for active and 2 for inactive. If you're coding in a language that supports enumeration types, you may want to refactor the status field as an enumeration.

The field measureOfQuality holds a double value that represents the goodness of the Bee object's memoryMatrix. In the case of the TSP, a natural measure of solution quality is the path length represented by the memoryMatrix object. Notice that in this situation, shorter path lengths are better than longer path lengths and so smaller values of the measureOfQuality field represent better solutions than larger

values. Every SBC implementation must have some way of computing a measure of solution quality. In many situations this measure is best represented by a value of type double.

The third common data field in the Bee class is named numberOfVisits. This field holds an int value that represents the number of times the Bee object has visited a particular food source/problem solution, without finding a better neighbor solution. This field is used to simulate a bee visiting a food source until that food source is used up. For an active bee, when the value of the numberOfVisits field exceeds a threshold value, the simulated bee will have virtually exhausted the food supply and transition to inactive status (and an inactive bee will transition to active status).

The Bee constructor accepts values for the four data fields, status, memoryMatrix, measureOfQuality, and numberOfVisits. Note that the Bee constructor must accept a value for measureOfQuality because a Bee can't directly compute this from its memoryMatrix field; determining the measure of quality depends on information stored in the problem-specific CitiesData object.

The Bee class definition contains a ToString method, which exposes the values of the four data fields. The Bee.ToString method is not absolutely necessary but can be quite useful during SBC development to help uncover bugs by using WriteLine statements.

The Hive Data Fields

The Hive class code is presented in **Figure 5**. There are 14 hive data fields and understanding the purpose of each is the key to understanding how to implement a specific SBC algorithm.

For simplicity and easier debugging with WriteLine statements, all 14 data fields are defined with public scope. You may want to restrict the fields to private scope and create properties for those fields you need to access outside the class definition.

The first field is named random and is type Random. SBC algorithms are probabilistic and the random object is used to generate pseudorandom numbers for several purposes. The random object will be instantiated in the hive constructor.

The second data field is an object of type CitiesData. The SBC implementation needs to know details of the problem being solved. In most cases, such as this one, the problem data is represented as an object. Recall that CitiesData has a list of city labels and a method that returns the distance between any two cities.

Figure 5 The 14 Hive Data Fields

```
static Random random = null;

public CitiesData citiesData;

public int totalNumberBees;
public int numberInactive;
public int numberActive;
public int numberScout;

public int maxNumberCycles;
public int maxNumberVisits;

public double probPersuasion = 0.90;
public double probMistake = 0.01;

public Bee[] bees;
public char[] bestMemoryMatrix;
public double bestMeasureOfQuality;
public int[] indexesOfInactiveBees;
```

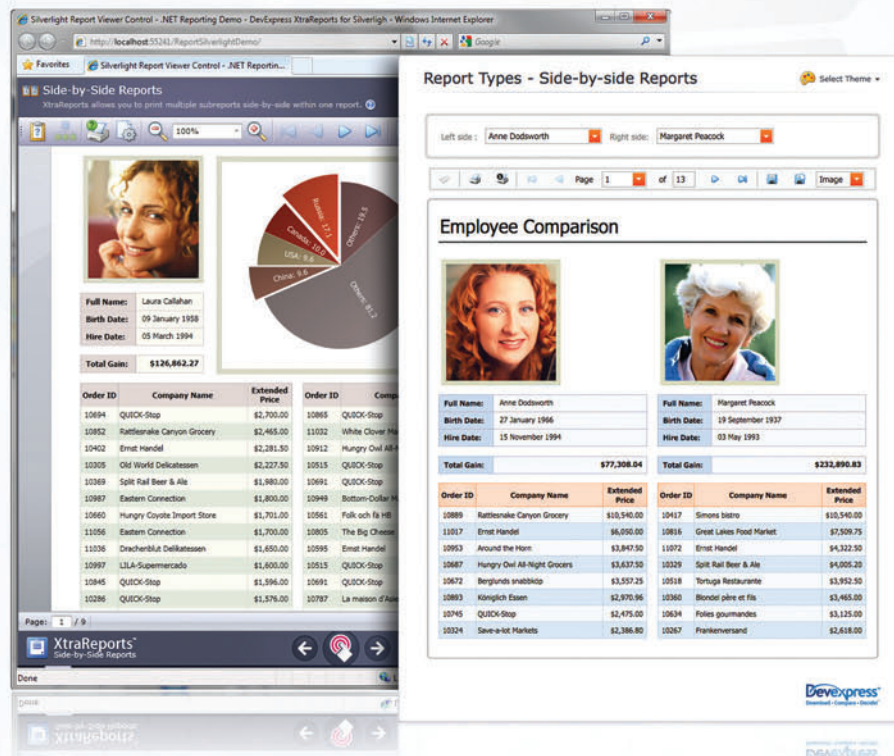

ca·pa·ble (rə'pōrtɪŋ)

– **adjective**

1. having power and ability; efficient; competent
2. having the tools required for a specific task

– **synonyms**

accomplished, efficient, experienced



A Reporting Library should be able to accommodate the structure of your data. Our banding system allows for the skillful construction of elegant reports that conform to you and your data—without imposing restrictions on your ability to address customer needs in the shortest possible time.

Experience the power of the **XtraReports Suite** today.
Download your **FREE** trial – www.DevExpress.com/FreeEval.

Devexpress™
Download • Compare • Decide!



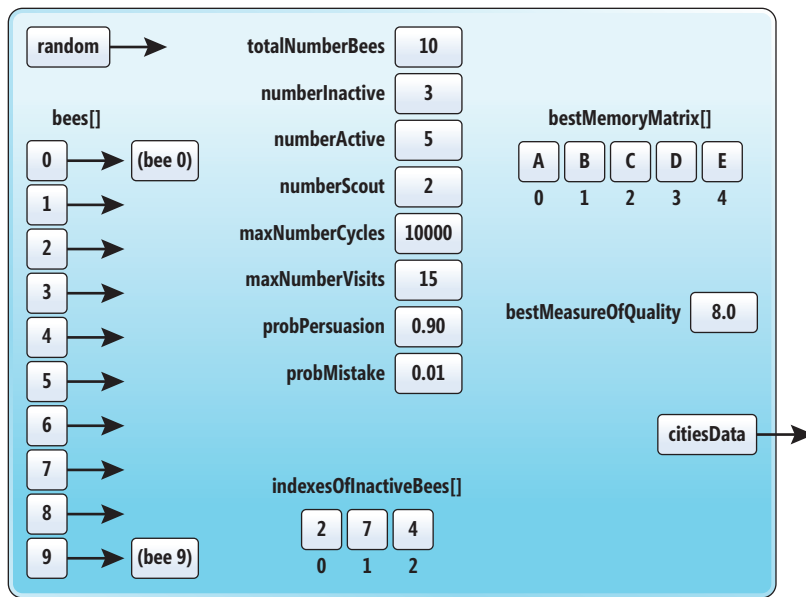


Figure 6 The Hive Representation

The third through sixth data fields are int variables that hold the total number of bees, the number of inactive bees, the number of active bees and the number of scout bees. As mentioned earlier, because each bee represents a potential solution, the more bees in the hive, the better. However, larger numbers of bees degrade program performance.

The seventh data field, `maxNumberCycles`, is a threshold value used to constrain how long the `Solve` method runs. One cycle represents processing of each bee in the hive.

The eighth data field, `maxNumberVisits`, is a threshold value used to prevent a bee from staying too long at a particular solution. In every iteration of the main processing loop in the `Solve` method, if a bee does not find a neighbor food source with better quality, the bee's `numberOfVisits` counter is incremented. If the `numberOfVisits` counter in a `Bee` object exceeds the `maxNumberVisits` threshold value, the bee transitions to an inactive state.

The ninth data field, `probPersuasion`, is a probabilistic threshold value used to determine whether an inactive bee who observes the waggle dance of a bee that has returned to the hive with a better solution will be persuaded to update its memory with the better solution.

The value of `probPersuasion` is hard-coded to 0.90, which means that an inactive bee will be persuaded to accept a better solution about 90 percent of the time. The 0.90 value for `probPersuasion` is based on research findings, but you may want to experiment with other values. Larger values will produce an SBC algorithm which converges to a solution more quickly, at the risk of more likely converging to a non-optimal solution.

The tenth data field, `probMistake`, is a probabilistic threshold value used to determine whether an active bee will make a mistake—that is, incorrectly reject a neighbor solution that's better than the bee's current solution, or incorrectly accept a neighbor solution that's worse than the bee's current solution. The value of `probMistake` is hardcoded to 0.01, which means that an active bee will make a mistake in evaluating a neighbor solution about 1 percent of the time.

The 11th data field, `bees`, is an array of `Bee` objects. Recall that each bee has a status (active, inactive, scout), a solution (memoryMatrix), a measure of the solution's quality (`measureOfQuality`), and a counter of the number of times a particular virtual food source has been visited without finding a better neighbor food source (`numberOfVisits`). Because a `Bee` is defined as a class, each entry in the `bees` array is a reference to a `Bee` object.

The 12th data field, `bestMemoryMatrix`, is an array of `char` and represents the best solution in the `bees` array. Recall that because simulated bee colony algorithms are specific implementations of a meta-heuristic, the representation of a problem solution will vary from problem to problem. An alternative design approach to hardcoding a solution type definition is to parameterize this data field as a generic type. When I use an SBC algorithm I'm usually trying to solve a specific problem, so I prefer to recode each new SBC implementation from scratch.

The 13th data field, `bestMeasureOfQuality`, is the measure of quality that corresponds to the `bestMemoryMatrix` solution.

The last hive data field, `indexesOfInactiveBees`, is an array of `int`. This array holds the indexes of the bees in the hive that are currently inactive. Recall that active bees can transition to an inactive state and inactive bees can transition to an active state. An SBC algorithm implementation must frequently determine which bees are currently inactive when an active bee performs a virtual waggle dance, and storing the indexes of inactive bees improves performance compared to the alternative of iterating through the entire `bees` array and checking the status data field of each bee.

A visual representation of a possible `Hive` object is presented in **Figure 6**. The hive shown has 10 bees: 5 active, two scout and three inactive. The currently inactive bees are at indexes 2, 7 and 4 in the `bees` array. The `CitiesData` object has five cities. The current best solution is:

A->B->E->C-D

This solution has a path length, in distance units, of:

$1.0 + (3 * 1.0) + (2 * 1.5) + 1.0 = 8.0$

Note that the `citiesData` field is a reference to a `CitiesData` object defined outside of the hive object.

The Hive Constructor

The code for the hive constructor is presented in **Figure 7**. The hive constructor accepts seven input parameters. Six of the parameters are scalar and one is an object (`citiesData`). The `totalNumberBees` parameter is redundant in the sense that it can be determined from `numberInactive`, `numberActive` and `numberScout`, but I feel the improvement in readability is worth the extra code.

The class-scope `random` object is instantiated with a seed value of 0. Supplying a seed value allows you to reproduce results. Next, six input parameter values for the scalar data fields are copied to hive data fields. The hive object has a total of 14 data fields; the threshold values `probPersuasion` and `probMistake` are hardcoded.

The `Hive` constructor takes the input `citiesData` parameter and assigns the `citiesData` field to the parameter as a reference. An

flex·i·ble (rə'pôrtɪŋ)

– **adjective**

1. easily modified; adaptable
2. demonstrating the ability to adapt to new and different requirements

– **synonyms**

adaptable, modifiable, adjustable



A Reporting Library must be flexible so as to fully accommodate your platform requirements. The XtraReports Suite allows you to target WinForms, ASP.NET, WPF or Silverlight—without the need to re-create individual reports from scratch.

Experience the power of the **XtraReports Suite** today.
Download your **FREE** trial – www.DevExpress.com/FreeEval.

alternative to this by-reference approach is to make a new copy of the problem data, like so:

```
int n = citiesData.cities.Length;
this.citiesData = new CitiesData(n);
```

This approach uses more memory, but avoids potential side-effect errors. The alternative approach can be used if you're refactoring the code presented here to a programming language that doesn't support pointers or references.

At this point in the Hive constructor all entries in the bees array will be null. Next, the constructor initializes the globally best solution (that is, the best solution among all bees in the hive) and corresponding solution quality:

```
this.bestMemoryMatrix = GenerateRandomMemoryMatrix();
this.bestMeasureOfQuality =
    MeasureOfQuality(this.bestMemoryMatrix);
```

The `GenerateRandomMemoryMatrix` helper method generates a random solution. The `MeasureOfQuality` helper method accepts the randomly generated solution and computes its quality. I'll discuss the code for these two helper methods later in the article.

After initializing the global best solution and its corresponding quality, the hive constructor allocates the `indexesOfInactiveBees` bees array using the value in the `numberInactive` field. At this point all values in this indexes array will be 0.

Figure 7 Hive Constructor

```
public Hive(int totalNumberBees, int numberInactive,
    int numberActive, int numberScout, int maxNumberVisits,
    int maxNumberCycles, CitiesData citiesData) {

    random = new Random(0);

    this.totalNumberBees = totalNumberBees;
    this.numberInactive = numberInactive;
    this.numberActive = numberActive;
    this.numberScout = numberScout;
    this.maxNumberVisits = maxNumberVisits;
    this.maxNumberCycles = maxNumberCycles;

    this.citiesData = citiesData;

    this.bees = new Bee[totalNumberBees];
    this.bestMemoryMatrix = GenerateRandomMemoryMatrix();
    this.bestMeasureOfQuality =
        MeasureOfQuality(this.bestMemoryMatrix);

    this.indexesOfInactiveBees = new int[numberInactive];

    for (int i = 0; i < totalNumberBees; ++i) {
        int currStatus;
        if (i < numberInactive) {
            currStatus = 0; // inactive
            indexesOfInactiveBees[i] = i;
        }
        else if (i < numberInactive + numberScout)
            currStatus = 2; // scout
        else
            currStatus = 1; // active

        char[] randomMemoryMatrix = GenerateRandomMemoryMatrix();
        double mq = MeasureOfQuality(randomMemoryMatrix);
        int numberOfVisits = 0;

        bees[i] = new Bee(currStatus,
            randomMemoryMatrix, mq, numberOfVisits);

        if (bees[i].measureOfQuality < bestMeasureOfQuality) {
            Array.Copy(bees[i].memoryMatrix, this.bestMemoryMatrix,
                bees[i].memoryMatrix.Length);
            this.bestMeasureOfQuality = bees[i].measureOfQuality;
        }
    }
}
```

The next part of the constructor code iterates through each Bee object in the bees array and instantiates it using the Bee constructor. The logic in this loop assumes that the first `numberInactive` cells in the bees array are inactive bees, the next `numberScout` cells are scout bees and the remaining cells are active bees.

For example, if there are five active bees, two inactive bees and three scout bees, the constructor initializes a bees array of size 10, and instantiates cells 0 and 1 as inactive bees, cells 2-4 as scout bees and cells 5-9 as active bees. Additionally, the `indexesOfInactiveBees` array will have size 2 and initially hold values 0 and 1.

After the status of the current bee is determined based on the loop index variable, a randomly generated solution is created and its corresponding quality is computed, the number of visits counter is explicitly set to 0, and the Bee constructor is called:

```
char[] randomMemoryMatrix = GenerateRandomMemoryMatrix();
double mq = MeasureOfQuality(randomMemoryMatrix);
int numberOfVisits = 0;
bees[i] = new Bee(currStatus, randomMemoryMatrix,
    mq, numberOfVisits);
```

After each bee is instantiated, the quality of the bee's randomly generated solution is checked to see if it's better than the global best solution. If so, the current bee's solution and corresponding quality are copied to the `bestMemoryMatrix` and `bestMeasureOfQuality` fields. Note in the check for a global best solution quality, a smaller value is better than a larger value because quality value are path lengths and the TSP wishes to minimize the path length.

Instead of explicitly copying a bee's memory into the `bestMemoryMatrix` array, an alternative approach is to assign by reference. This approach improves performance at the expense of an increase in complexity.

Three Essential SBC Methods

Every SBC algorithm implementation must have three problem-specific methods: a method to generate a random solution, a method to generate a neighbor solution relative to a given solution, and a method to compute the quality of a given solution. In this TSP example each method is implemented in a custom and completely problem-dependent way.

A design alternative is to define interfaces and implement these interfaces. Programming via interfaces has several advantages and disadvantages compared to the non-interface approach used, but is mostly a matter of personal preference in my opinion. The method to generate a random solution, `GenerateRandomMemoryMatrix`, is shown here:

```
public char[] GenerateRandomMemoryMatrix() {
    char[] result = new char[this.citiesData.cities.Length];
    Array.Copy(this.citiesData.cities, result,
        this.citiesData.cities.Length);
    for (int i = 0; i < result.Length; i++) {
        int r = random.Next(i, result.Length);
        char temp = result[r];
        result[r] = result[i];
        result[i] = temp;
    }
    return result;
}
```

Because a solution to the TSP problem is an array of char where each char represents a city label, the `GenerateRandomMemoryMatrix` method returns a char array. The local result array size is allocated based on the hive's `CitiesData` object, and the city IDs stored in the hive's reference to the `CitiesData` object are copied

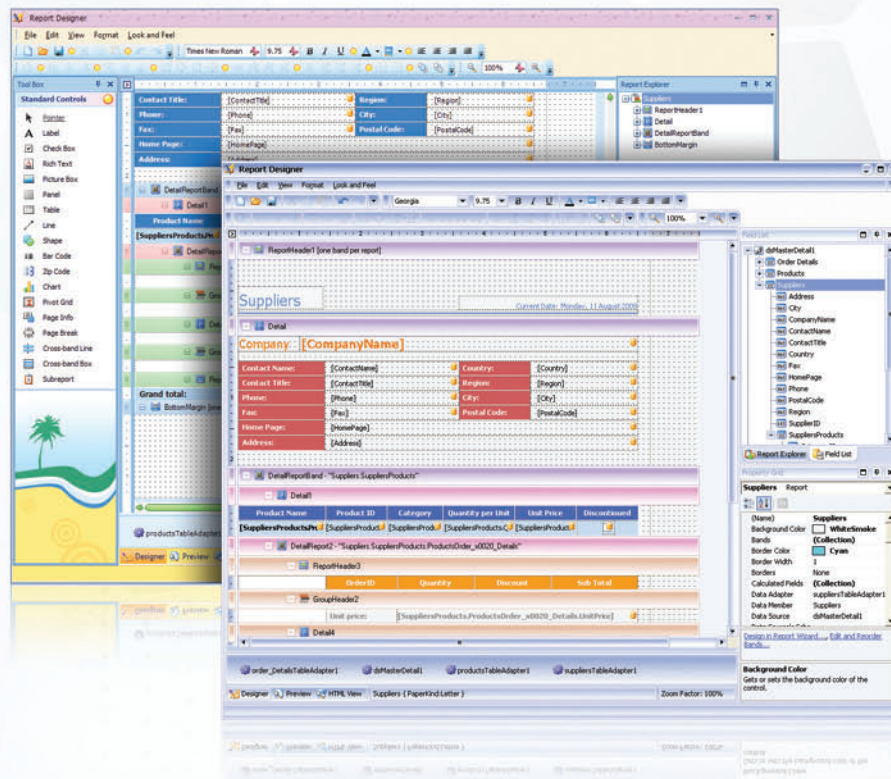
de·pend·a·ble (rə'pōrtɪŋ)

– **adjective**

1. trustworthy; reliable
2. able to rely upon an individual or organization in order to achieve your goals

– **synonyms**

solid, faithful, responsible



Though the award-winning XtraReports Suite delivers on its promise of multi-platform reporting, technical questions and issues will undoubtedly arise. Our support team is second to none when it comes to timeliness and quality—we are your extended team and want to do everything possible to ensure your success in the marketplace.

Experience the power of the **XtraReports Suite** today.
Download your **FREE** trial – www.DevExpress.com/FreeEval.

Figure 8 Generating a Neighbor Solution

```
public char[] GenerateNeighborMemoryMatrix(char[] memoryMatrix) {
    char[] result = new char[memoryMatrix.Length];
    Array.Copy(memoryMatrix, result, memoryMatrix.Length);

    int ranIndex = random.Next(0, result.Length);
    int adjIndex;
    if (ranIndex == result.Length - 1)
        adjIndex = 0;
    else
        adjIndex = ranIndex + 1;

    char tmp = result[ranIndex];
    result[ranIndex] = result[adjIndex];
    result[adjIndex] = tmp; return result;
}
```

into the result array. The order of the values in result array is then randomized using the class scope random object and the Fisher-Yates shuffle algorithm (sometimes called the Knuth shuffle).

At first, it might seem that method GenerateRandomMemoryMatrix conceptually belongs to a Bee object. However, because generating a random solution depends in part on problem-specific data—CitiesData in this case—placing the random solution generation method in the overall hive definition is a better design.

The method to generate a neighbor solution, GenerateNeighborMemoryMatrix, is presented in **Figure 8**.

A key concept in SBC algorithms is the idea that each virtual food source that represents a solution has some sort of neighbor. Without the neighbor concept, the entire idea of an algorithm based on bee behavior collapses. In the case of the TSP, where a solution can be represented as an array of city IDs representing a path from city to city, a natural neighbor solution relative to a current solution is a permutation of the current solution where two adjacent cities have been exchanged.

For example, if a current TSP solution is A,B,C,D,E, then a reasonable neighbor solution is A,C,B,D,E. It's not so obvious if a permutation where any two arbitrary cities are exchanged (as opposed to two adjacent cities) represents a reasonable neighbor solution. For the previous example, is A,D,C,B,E a reasonable neighbor solution? Deciding on the definition of a neighbor solution for an SBC algorithm is problem-dependent and typically involves subjective criteria.

The neighbor-solution concept also serves to illustrate in part why non-numeric combinatorial problems are especially well suited for solution by SBC algorithms. If a problem is inherently numeric, the idea of a neighbor is often difficult to define satisfactorily. If a problem is inherently combinatorial, the idea of a neighbor can often be nicely defined by some form of mathematical permutation or combination.

The GenerateNeighborMemoryMatrix method accepts a bee's current memoryMatrix representation of a solution as an input parameter and copies it into a result array. The method selects a single random index into the current result array using the class-scope random object. If the random index points to the last cell then the first and last city IDs are exchanged; otherwise, if the random index points to any non-last cell, the IDs pointed to by the random index and the next index are exchanged.

The neighbor solution concept is related to the maxNumberVisits value. There's some research that suggests a good value for maxNumberVisits is about five times the number of neighbor solutions possible for any given solution. For example, for three cities

(A,B,C), if a neighbor solution is defined as exchanging any pair of adjacent cities, then there are three possible neighbors (exchange A and B, exchange B and C, exchange A and C). So for 20 cities, a reasonable maxNumberVisits value is about $20 * 5 = 100$.

The method to evaluate the quality of a bee's solution, MeasureOfQuality, is:

```
public double MeasureOfQuality(char[] memoryMatrix) {
    double answer = 0.0;
    for (int i = 0; i < memoryMatrix.Length - 1; ++i) {
        char c1 = memoryMatrix[i];
        char c2 = memoryMatrix[i + 1];
        double d = this.citiesData.Distance(c1, c2);
        answer += d;
    }
    return answer;
}
```

To solve a problem using an SBC algorithm, an essential characteristic of the problem is that any solution must be able to be evaluated to yield a measure of the solution's quality. In conceptual terms, a real-world combinatorial optimization problem almost always has some inherent and common-sense measure of quality. However, in practical terms, computing the measure of quality of a solution can be difficult and time-consuming.

In this example, the MeasureOfQuality method simply iterates through every pair of consecutive city IDs in the solution represented by parameter memoryMatrix, determines the distance between each pair using the Distance method of the CitiesData object, and accumulates the total distance. Recall that the city data was artificially constructed so that the distance between any two cities could be quickly and easily computed simply by using the ordinal distance between two city IDs. But in a real problem, the distance between two cities would likely have to be looked up in some sort of data structure. In SBC implementations, the MeasureOfQuality method is often the routine that dominates the running time of the program. Therefore, it's usually worthwhile to make sure that this method is optimized for performance—as well as feasible, given the memory resources of the host system.

Figure 9 The Solve Method

```
public void Solve(bool doProgressBar) {
    bool pb = doProgressBar;
    int numberOfSymbolsToPrint = 10;
    int increment = this.maxNumberCycles / numberOfSymbolsToPrint;
    if (pb) Console.WriteLine("\nEntering SBC Traveling Salesman Problem algorithm main processing loop\n");
    if (pb) Console.WriteLine("Progress: |=====|");
    if (pb) Console.Write(" ");
    int cycle = 0;

    while (cycle < this.maxNumberCycles) {
        for (int i = 0; i < totalNumberBees; ++i) {
            if (this.bees[i].status == 1)
                ProcessActiveBee(i);
            else if (this.bees[i].status == 2)
                ProcessScoutBee(i);
            else if (this.bees[i].status == 0)
                ProcessInactiveBee(i);
        }
        ++cycle;

        if (pb && cycle % increment == 0)
            Console.Write("^");
    }

    if (pb) Console.WriteLine("");
}
```


trans·par·ent (rə'pɔrtɪŋ)

– adjective

1. free from the unknown; open; detectable
2. delivering without secrecy and vagueness

– synonyms

clear, visible



Unlike competing products which do not ship 100% of the source code when purchased, the XtraReports Suite is fully transparent and not a black box with a myriad of unknowns. When you integrate DevExpress Reporting, you can rest assured that every single line of code is available for review.

Experience the power of the **XtraReports Suite** today.
Download your **FREE** trial – www.DevExpress.com/FreeEval.

DevExpress™
Download • Compare • Decide!



The Solve Method

The Solve method houses all the logic that simulates the behavior of foraging bees to solve a problem. The Solve method is moderately complex and uses three helper methods, ProcessActiveBee, ProcessScoutBee and ProcessInactiveBee. The ProcessActiveBee and ProcessScoutBee methods in turn use a DoWaggleDance helper method. The Solve method is presented in **Figure 9**.

Most of the actual work is farmed out to helper methods ProcessActiveBee, ProcessScoutBee and ProcessInactiveBee. A Boolean input parameter is passed to Solve to indicate whether to print a rudimentary text-based progress bar. This is useful when developing an SBC algorithm to monitor the speed of the implementation and help uncover performance bottlenecks. This approach makes the assumption that the Solve method is part of a console application.

The value of the Boolean parameter value is transferred into a local Boolean variable named pb just to have a short variable name to work with. The numberOfSymbolsToPrint is set to 10 so that each increment in the status bar will represent 10 percent of the total progress, which is determined by the maxNumberCycles value (the increment variable is used to determine how many cycles represent 10 percent progress).

After the loop control variable, cycle, is initialized to 0, a while loop is used to process each bee in the hive. A for loop could just as easily be used. On each cycle, the bees array is iterated using a for loop and each Bee object is processed by the appropriate helper method. After each bee has been processed, if the Boolean doProgressBar parameter is true, the code uses the modulus operator, %, to check if it's time to print an update to the progress bar using a ^ character.

The ProcessActiveBee Method

The ProcessActiveBee method is the heart of an SBC algorithm and is the most complex method in terms of code size and branching. The ProcessActiveBee method is presented in **Figure 10**.

The ProcessActiveBee method accepts a single input parameter, i, which is the index of the bee in the bees array. The active bee first obtains a neighbor solution relative to its current solution stored in memoryMatrix, and then determines the quality of that neighbor:

```
char[] neighbor =
    GenerateNeighborMemoryMatrix(bees[i].memoryMatrix);
double neighborQuality = MeasureOfQuality(neighbor);
```

Next, the algorithm sets up three local variables that will be used later:

```
double prob = random.NextDouble();
bool memoryWasUpdated = false;
bool numberOfVisitsOverLimit = false;
```

The prob variable has a value between 0.0 and 1.0 and will be compared against the probMistake field value to determine if the bee makes a mistake in evaluating the neighbor solution—that is, rejects a better neighbor solution or accepts a worse neighbor solution.

The Boolean memoryWasUpdated value will be used to determine if the active bee should perform a waggle dance to the inactive bees (if true) or not (if false). The Boolean numberOfVisitsOverLimit will be compared against the maxNumberVisits field to determine if the bee has exhausted a particular food source without finding a better neighbor solution, and if so should convert from active status to inactive status.

If the current bee finds a better neighbor solution, the algorithm determines if the bee makes a mistake and rejects the better neighbor

Figure 10 The ProcessActiveBee Method

```
private void ProcessActiveBee(int i) {
    char[] neighbor = GenerateNeighborMemoryMatrix(bees[i].memoryMatrix);
    double neighborQuality = MeasureOfQuality(neighbor);
    double prob = random.NextDouble();
    bool memoryWasUpdated = false;
    bool numberOfVisitsOverLimit = false;

    if (neighborQuality < bees[i].measureOfQuality) { // better
        if (prob < probMistake) { // mistake
            ++bees[i].numberOfVisits;
            if (bees[i].numberOfVisits > maxNumberVisits)
                numberOfVisitsOverLimit = true;
        }
        else { // No mistake
            Array.Copy(neighbor, bees[i].memoryMatrix, neighbor.Length);
            bees[i].measureOfQuality = neighborQuality;
            bees[i].numberOfVisits = 0;
            memoryWasUpdated = true;
        }
    }
    else { // Did not find better neighbor
        if (prob < probMistake) { // Mistake
            Array.Copy(neighbor, bees[i].memoryMatrix, neighbor.Length);
            bees[i].measureOfQuality = neighborQuality;
            bees[i].numberOfVisits = 0;
            memoryWasUpdated = true;
        }
        else { // No mistake
            ++bees[i].numberOfVisits;
            if (bees[i].numberOfVisits > maxNumberVisits)
                numberOfVisitsOverLimit = true;
        }
    }

    if (numberOfVisitsOverLimit == true) {
        bees[i].status = 0;
        bees[i].numberOfVisits = 0;
        int x = random.Next(numberInactive);
        bees[indexesOfInactiveBees[x]].status = 1;
        indexesOfInactiveBees[x] = i;
    }
    else if (memoryWasUpdated == true) {
        if (bees[i].measureOfQuality < this.bestMeasureOfQuality) {
            Array.Copy(bees[i].memoryMatrix, this.bestMemoryMatrix,
                bees[i].memoryMatrix.Length);
            this.bestMeasureOfQuality = bees[i].measureOfQuality
        }
        DoWaggleDance(i);
    }
    else {
        return;
    }
}
```

or if the bee accepts the better neighbor. Similarly, if the current bee did not find a better neighbor solution, the algorithm determines whether the bee makes a mistake and accepts the worse neighbor solution or does not make a mistake and rejects the neighbor.

Notice that there are two different types of mistakes possible, but that both types of mistakes have the same probability, probMistake = 0.01. There's some research that suggests using two different probabilities for the two different types of mistakes does not improve the effectiveness of SBC algorithms, but you may want to experiment with two different threshold values.

After the current active bee accepts or rejects the neighbor solution, the algorithm checks if the number of visits counter has exceeded the maxNumberVisits threshold. If so, the current bee's status is converted to inactive, a randomly selected inactive bee is converted to active status and the indexesOfInactiveBees array is updated. Next, the algorithm checks to see if the bee's memory was updated. If so, the new solution is checked to see if it's a new global

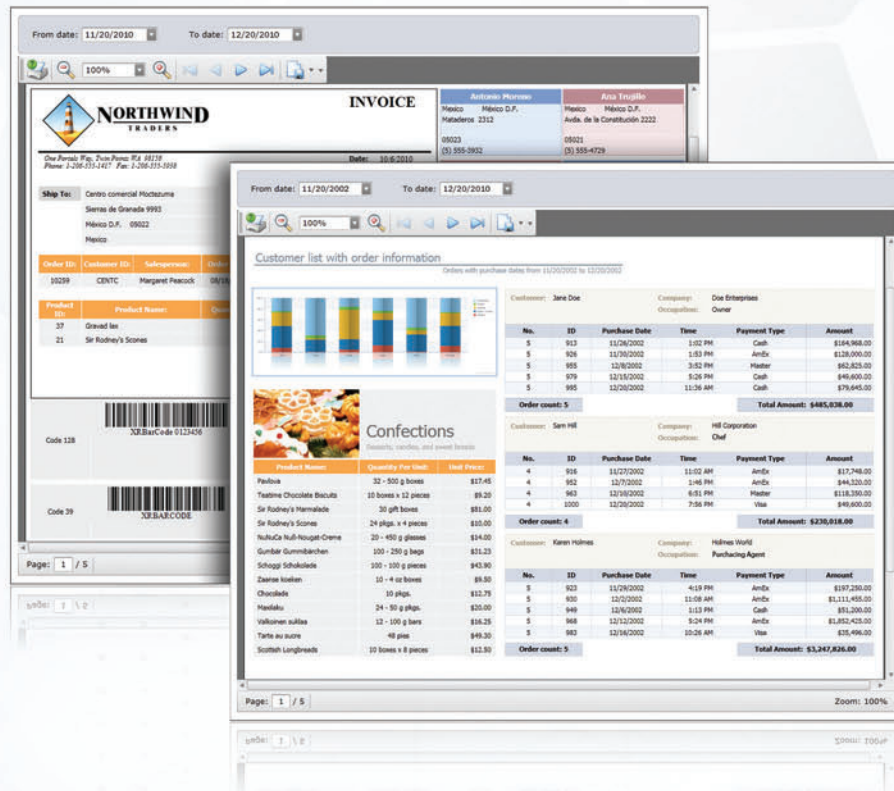
af·ford·able (rə'pôrtɪŋ)

– adjective

1. within a developer's financial means; reasonably priced
2. all the benefits, without the high price

– synonyms

fair, reasonable



A feature-complete and multi-platform Reporting Library should not force you to spend thousands of dollars—it should be affordable and ship with the capabilities you need to get the job done... and offer the services you need to achieve mastery. **Prices start at \$349.99**

Experience the power of the **XtraReports Suite** today.
Download your **FREE** trial – www.DevExpress.com/FreeEval.

Devexpress™
Download • Compare • Decide!



Figure 11 The ProcessScoutBee Method

```
private void ProcessScoutBee(int i) {
    char[] randomFoodSource = GenerateRandomMemoryMatrix();
    double randomFoodSourceQuality =
        MeasureOfQuality(randomFoodSource);
    if (randomFoodSourceQuality < bees[i].measureOfQuality) {
        Array.Copy(randomFoodSource, bees[i].memoryMatrix,
            randomFoodSource.Length);
        bees[i].measureOfQuality = randomFoodSourceQuality;
        if (bees[i].measureOfQuality < bestMeasureOfQuality) {
            Array.Copy(bees[i].memoryMatrix, this.bestMemoryMatrix,
                bees[i].memoryMatrix.Length);
            this.bestMeasureOfQuality = bees[i].measureOfQuality;
        }
        DoWaggleDance(i);
    }
}
```

best solution, and then a private helper method, `DoWaggleDance`, is called to simulate the bee returning to the hive and conveying information about the new food source to the inactive bees.

The DoWaggleDance Method

The `DoWaggleDance` helper method simulates an active or scout bee returning to the hive and then performing a waggle dance to inactive bees in order to convey information about the location and quality of a food source. Here's the `DoWaggleDance` method:

```
private void DoWaggleDance(int i) {
    for (int ii = 0; ii < numberInactive; ++ii) {
        int b = indexesOfInactiveBees[ii];
        if (bees[i].measureOfQuality < bees[b].measureOfQuality) {
            double p = random.NextDouble();
            if (this.probPersuasion > p) {
                Array.Copy(bees[i].memoryMatrix, bees[b].memoryMatrix,
                    bees[i].memoryMatrix.Length);
                bees[b].measureOfQuality = bees[i].measureOfQuality;
            }
        }
    }
}
```

The input parameter `i` is the index of the current bee performing the virtual waggle dance. The measure of quality of the current bee's solution is compared against the measure of quality of each inactive bee. If the current bee's solution is better and the current inactive bee is persuaded (with probability `probPersuasion = 0.90`), the current bee's memory is copied to the inactive bee's memory.

Note that there are many opportunities to insert error checking into the code presented in this article. For example, inside the for loop in `DoWaggleDance`, you may want to check the current inactive bee's status with:

```
if (bees[b].status != 0) throw new Exception( . . );
```

Or you may want to verify the inactive bee's number of visits counter with:

```
if (bees[b].numberOfVisits != 0) throw new Exception( . . );
```

ProcessScoutBee and ProcessInactiveBee

The `ProcessScoutBee` helper method used by the `Solve` method simulates the action of a scout bee randomly searching for appealing food sources. The `ProcessScoutBee` method is presented in **Figure 11**.

The input parameter `i` represents the index of a scout bee in the `bees` array. A scout bee generates a random solution, checks if the random solution is better than the current solution in memory, and, if so, copies the random solution into memory. Recall that smaller quality values are better. If the scout bee has found a

better solution, the algorithm checks to see if the new solution is a global best solution.

Note that unlike active bees, in this SBC implementation scout bees never make mistakes evaluating the quality of a food source. There's no current research on the effect of scout bee mistakes.

The `ProcessInactiveBee` method is:

```
private void ProcessInactiveBee(int i) {
    return;
}
```

In this SBC implementation inactive bees are exactly that—inactive—so the `ProcessInactiveBee` method is merely a placeholder in case you wish to implement some problem-dependent logic for an inactive bee. One possible modification would be to randomly mutate an inactive bee's memory with some very small probability.

Wrapping Up

The overall process of implementing an SBC algorithm starts with problem identification. Complex, non-numerical, combinatorial optimization problems with no practical deterministic solutions are often good candidates for an SBC solution. The target problem must have a way to represent a solution (often as an array or matrix) and each solution must have some sort of a neighbor solution and a measure of solution quality.

For example, consider the problem of dividing a graph into two parts so that the number of connections within each part is maximized and the number of connections between the two parts is minimized. This graph partition problem is combinatorial and there's no quick algorithm that finds the optimal solution (although there are deterministic algorithms that are good at finding near-optimal solutions). There are many other NP-complete and NP-hard problems that might be tackled using an SBC.

SBC algorithms are based on the behavior of natural systems. There are other such algorithms, including genetic algorithms (GA) based on natural selection and evolution, ant colony optimization (ACO) based on the behavior of ants, and simulated annealing (SA) based on the physical properties of cooling metals.

Algorithms based on natural systems are often easy to implement relative to deterministic approaches. However, algorithms based on natural systems typically require the specification of values for several parameters that tend to be sensitive with regard to their effect on solution convergence speed and solution accuracy. In the case of an SBC, sensitive parameters that must be fine-tuned for each problem include the number of each type of bee, the maximum number of visits before a food source is exhausted, inactive bee persuasion probability threshold and active bee mistake probability.

Although SBC algorithms aren't applicable to every problem, in some scenarios they can be extremely powerful tools. ■

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:
Dan Liebling and Anne Loomis Thompson, both of Microsoft Research

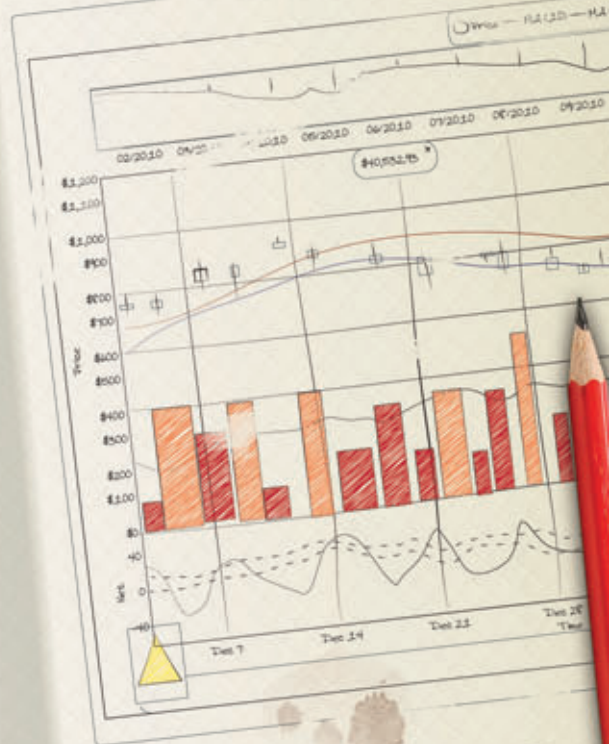
Your Guide to .NET Charts

Ahead of Schedule and Under Budget

With the right tools you can turn your long development list of requirements to a finished solution, from conception to completion. Developers, just like you, continue to turn to ComponentOne Charts® for their enterprise solutions. Get rock solid, advanced charts for all .NET platforms; download ComponentOne Studio® Enterprise today.

Requirements:

- ✓ Financial & scientific charts
- ✓ Stacked charts
- ✓ Interactivity - zoom, animation, drag-and-drop
- ✓ Tooltips and markers
- ✓ Trend lines
- ✓ ~~Fast~~ Really fast
- ✓ Live updates
- ✓ 2D & 3D
- ✓ Multiple platforms - Silverlight, WPF, ASP.NET and WinForms



ComponentOne®
**Studio®
Enterprise**

Download your
FREE Trial @
c1.ms/c1charts

C1 ComponentOne®

© 2011 ComponentOne LLC. All rights reserved. All other product and brand names are trademarks and/or registered trademarks of their respective holders.

Introduction to WebMatrix

Clark Sell

There's no shortage of available tooling in the market for a Web developer today. In fact, chances are good that you already spend much of your day in Visual Studio. If you've heard of Microsoft WebMatrix, you might be wondering what it is and what its existence means to you—maybe even why you should care.

In this article I'll explore those questions and more. I'll start by exploring a few of the recent changes to the ASP.NET stack. Then

I'll explain how to get WebMatrix set up and create a "Hello Web" app. Then I'll dive into some of the bells and whistles such as layout, helpers, data access, reporting and, of course, deploying. Before all that, I'll start by defining WebMatrix.

WebMatrix is a new all-in-one Web site editor for ASP.NET Web Pages. It's aimed at a different Web developer and a different part of ASP.NET than is usual for a Microsoft product. WebMatrix isn't a competitor to Visual Studio; it's more of a complement, with some overlap between the two.

You'll find installing WebMatrix to be quick and painless. That's because the entire package and its dependencies are less than 50MB. The package includes a text editor, Web server, database engine and the underlying framework—basically everything you need to create a Web site and deploy it. What might surprise you is that it's not just limited to ASP.NET. WebMatrix also supports PHP and MySQL. This article will specifically focus on the .NET features of WebMatrix.

The New ASP.NET Stack

Over the past few years, the ASP.NET stack has gone through a bit of a transformation. In April 2009, ASP.NET MVC was released as a new option for Web application development in ASP.NET. The MVC, or Model-View-Controller, pattern was nothing new, but its implementation on top of ASP.NET was. Furthermore, Web Forms and MVC can both coexist in the same site in perfect harmony.

To properly enable its introduction into ASP.NET, there was some gentle framework refactoring going on so that those LEGO pieces could easily snap together how you chose. Fast forward

This article discusses:

- The new ASP.NET stack
- Getting started
- Hello Web
- Coding in Razor
- Layout
- Helpers
- Data access
- Serving the site
- Search Engine Optimization
- Deployment
- Scalability

Technologies discussed:

ASP.NET, WebMatrix, Razor, NuGet

Code download available at:

code.msdn.microsoft.com/mag201104WebMatrix

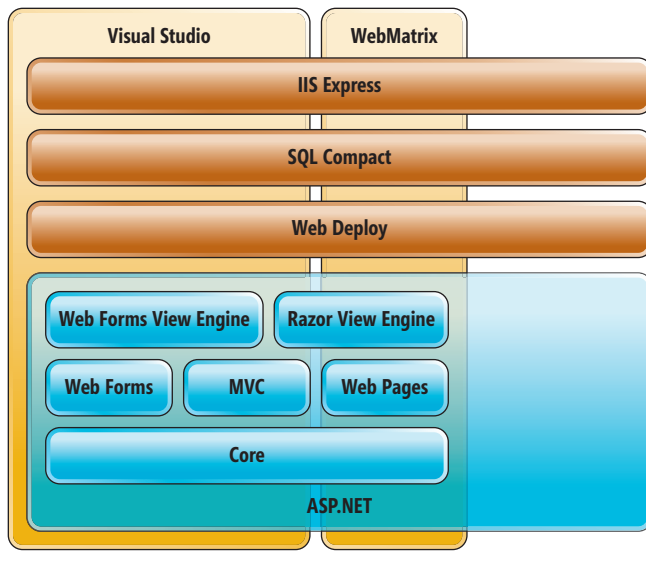


Figure 1 The Web Stack

to now, and Microsoft has just released ASP.NET MVC 3 and WebMatrix. These represent major releases that include a number of new additions and enhancements. There are three notable new additions to the framework: the Razor Parser and View Engine, or Razor for short; ASP.NET Web Pages; and IIS Express. **Figure 1** depicts the relationships between all of the associated ASP.NET framework pieces.

In **Figure 1**, you might have noticed that Razor, IIS Express, SQL Compact and Web Deploy are valid options for other areas in ASP.NET Web development. Furthermore, with the release of ASP.NET MVC 3, Razor is now the default view engine. I'll cover Razor in more detail later.

Conversely, ASP.NET Web Pages is being introduced as a peer to Web Forms and MVC. The Web Page model is a page-centric execution model, similar to PHP. Markup and code are both contained in the page itself, with helpers being leveraged to keep the code succinct. You write these pages in Razor in either Visual Basic or C#.

Getting Started

Getting started with WebMatrix is easy. The installation is delivered courtesy of the Microsoft Web Platform Installer version 3.0, or WebPI for short. It's a client-side program that makes things such as installing SQL Express or ASP.NET MVC a breeze. For more information about WebPI, visit microsoft.com/web/downloads/platform.aspx. To quickly install it, visit web.ms/webmatrix and click the Download Now button. That will not only install WebPI but will install WebMatrix as well. Once the installation completes, you'll have the following installed:

- Microsoft WebMatrix (editor)
- ASP.NET Web Pages with Razor Syntax (engine and framework)
- SQL Compact (database)
- IIS Express (development Web server)

I mentioned earlier that WebMatrix also supports PHP development. While the package supports development out of the box, you'll need to install the PHP runtime if you're going to create a PHP site from scratch.

Hello World Web

Just about every technical book out there has the classic "Hello World" example, so let's create a "Hello Web" site with WebMatrix. When you open WebMatrix for the first time, you'll be presented with a dialog with four choices. Each of these choices represents a different starting point:

- My Sites
- Site From Folder
- Site From Template
- Site From Web Gallery

My Sites simply includes the past sites you've been working on. *Site From Folder* will open any folder as the root of a site. Anything that's listed in that folder will show up as part of the assets of the site. You can also right-click on a folder and select Open as a Web Site with Microsoft WebMatrix, which behaves the same as the *Site From Folder*.

WebMatrix ships with a number of prebuilt Web templates to jumpstart your site creation. Selecting *Site From Template* will set up your folder structure with all of the source code needed to have a site based on that template. At the time of this writing, there were five templates: Empty Site, Starter Site, Bakery, Photo Gallery and Calendar. More will be available in the future.

WebMatrix ships with a number of prebuilt Web templates to jumpstart your site creation.

Site From Gallery might be the most interesting. For a number of years now, you could use WebPI and download applications such as Umbraco or WordPress, customize them at will and deploy to your hosting provider. Unfortunately there really wasn't an integrated end-to-end story. Now, from within WebMatrix, you can actually start development from an existing application such as Umbraco. WebMatrix and WebPI will set up everything needed to start your site from one of the applications listed in the gallery. Once WebMatrix sets things up, you can customize as needed and use the deployment capabilities of WebMatrix to deploy to your hosting provider.

To create our Hello Web demo, let's start by opening WebMatrix and selecting *Start from Template*. Select the *Empty Site* template and give the site a name—HelloWeb

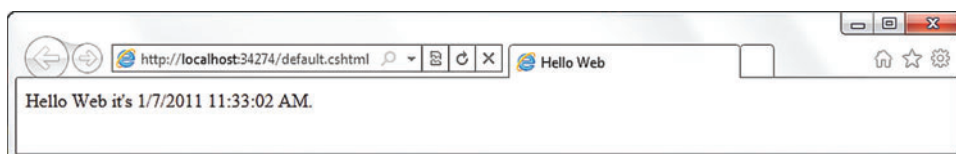


Figure 2 Hello Web in the Browser

Figure 3 CodingRazor.cshtml

```
@[
var cars = new string [] { "Camaro", "Avalanche", "Honda";
var emailAddress = "cars@something.com";
}

<ul>
@foreach ( var car in cars ) {
<li>
<a href="http://@car.something.com">@car</a>

@if ( car == "Camaro" ) {
@: Is my favorite!
}
</li>
}
</ul>

To comment on this list of cars e-mail us at @emailAddress.
```

in this case. WebMatrix will create an empty folder with the name you chose. Now Select the *Files* tab and then *Add a file to your site* from that main window. The first thing you should notice is the new file extension. ASP.NET WebPages has two new file formats: CSHTML and VBHTML. The first two letters of the extension indicate the language that the Razor parser should use to parse code: C# and Visual Basic, respectively. I'll select a CSHTML file and name it *HelloWeb.cshtml*. That will automatically open that file prepopulated with some default HTML.

Adding the text *Hello Web* to the existing body element will finish Hello Web. WebMatrix will automatically detect all of the Web browsers you have installed. You can run your site in any or all of them with a simple button click. Clicking Run in the ribbon will start the default browser and open and render the page you had focus on. Of course, in the case of Hello Web, you'll see the words "Hello Web."

Coding in Razor

As I stated earlier, Razor is a code and markup templating engine. What that really means is that you write some syntax that's later interpreted and sent to the requestor. Here's the HTML for our simple Web page, with a little code calling `@DateTime.Now`:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Hello World</title>
</head>
<body>
Hello Web it's @DateTime.Now
</body>
</html>
```

That code makes a call to the `Now` method of the `.NET System.DateTime` class. The result is a page that would render what you see in **Figure 2**.

We can get a bit more complicated, too. Let's create some properties, run a loop, intermix some HTML and just mix it all together for fun. **Figure 3** is just that, with some valid HTML omitted for brevity. And **Figure 4** shows the rendered results.

Let's take a closer look at some of that code listed in **Figure 3**. Start with the hyperlink "http://

@(@car).something.com." You see "@(@car)." There I need to explicitly tell Razor that @car is the variable, because it's embedded within the URL. The next code nugget is the content block defined by "@:". Within the loop, there's an If statement containing the text content "Is my favorite!". If you remove "@:", Razor wouldn't compile because that text content isn't wrapped in HTML. Finally, at the end, we call @emailAddress, but end the sentence with a period. Razor was smart enough to know @emailAddress was a variable and not a class. The coolness doesn't end there. By viewing the source on that page, you'll see that pure HTML was rendered to the browser, as shown here:

```
<ul>
<li><a href="http://Camaro.something.com">Camaro</a> Is my favorite! </li>
<li><a href="http://Avalanche.something.com">Avalanche</a> </li>
<li><a href="http://Honda.something.com">Honda</a> </li>
</ul>
```

To comment on this list of cars e-mail us at cars@something.com

Layout

Razor introduces a new way to structure your site content in a reusable way. If you're familiar with the Master Pages idiom in ASP.NET Web Forms, Razor uses layout to accomplish the same thing, but with a different approach. As a tangential observation, it's important to note the two layout approaches don't work together, meaning Master Pages can't be used as a layout engine for Razor, and vice versa.

Razor is a code and markup templating engine.

To start with layout, I'm going to create a page that begins with an underscore; for example, `_MasterLayout.cshtml`. Underscored pages can't be rendered directly in the browser and must be referenced by other, public pages. In a way, an underscored page is like a class in an assembly marked *internal*. In that page, I'll structure how and where different pieces of the page will get inserted by downstream references. There are three main classes you use to help structure things: `RenderSection`, `RenderPage` and `RenderBody`. **Figure 5** shows a simple master layout page using all three.

This master page will need to be referenced by any child page that wants to inherit its defined layout. As seen here, the first line does exactly that:

```
@[ Layout = "~/layout/shared/_MasterLayout.cshtml"; ]
```

```
@section ClientScripts {
Custom Script Here </br>
}
```

Some body content

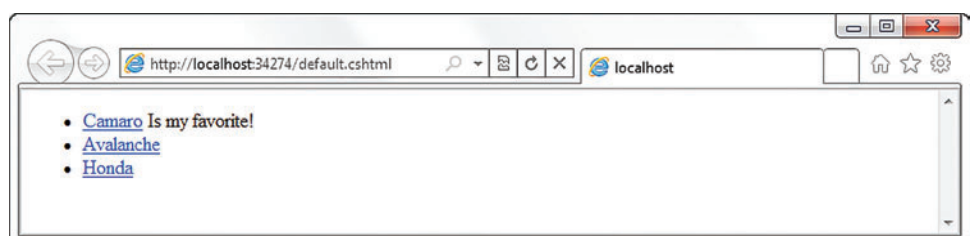


Figure 4 The Cars Code Rendered in a Browser

MORE THAN JUST
TRADITIONAL REPORTING

ASPOSE...
Your File Format Experts

CONVERT, IMPORT, EXPORT PRINT & MODIFY

Files Like:

DOCX PDF PPT ODF

XLSX SWF MPP MSG

Barcode Report InfoPath

File management solutions for:

- .NET
- Java
- SSRS Rendering Extensions
- SharePoint
- JasperReports Exporters

Try Aspose FREE for 30 days!

Word Documents **PDF**
HTML Excel Spreadsheets
RDLC AdHoc Queries & Reports
FLV PowerPoint Documents .NET
Barcodes **Aspose**
Free Trial

JAVA PPTs
MSG Flash W
RTF SharePoint F
OOXML 3D RENDERED

DOC Total Suite CHARTS
www.aspose.com

JASPERREPORTS

S PDF OpenDocument

S Viewer Free Tech Support

R Modify

S EXPORTING

POT PPTX EPUB

PPS **Excel**

Reporting

CREATING XPS 30 Day

PDF Converting Evaluation

Excel Spreadsheets

AdHoc Queries & Reports PowerPoint

RDLC SAVING FLV

Bar-.NET Documents

Codes Free Trial Aspose

IMPORTING JAVA

PPT SWF SharePoint

Msg Flash RTF

3D Rendered Charts Total Suite **DOC**

www.aspose.com JasperReports

POT PDF Viewer FREE TECH OpenDocument Converting

Exporting PPS SUPPORT **MODIFY** AdHoc Queries H

PRINTING E Excel SSRS PDF.NET & Reports T

P Reporting Excel Spreadsheets L

30 Day Evaluation B Creating XPS Word Documents

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



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

Figure 5 The RenderSection, RenderPage and RenderBody Classes

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title></title>
    @RenderSection("ClientScripts", required: false)
  </head>
  <body>
    @RenderPage("~/layout/shared/_header.cshtml")
    @RenderBody()
    @RenderPage("~/layout/shared/_footer.cshtml")
  </body>
</html>
```

In my layout page, I defined an optional section (@RenderSection) in the head section of our page. Because it was optional, a child page has the option to implement it if it so chooses. To implement a section, use the “@section” followed by the name you specified in the “@RenderSection” call. The remainder of the page (in this example) is assumed to be the content. This will be consumed by the “@RenderBody.” See the online source code accompanying this article for a more detailed example.

Running default.cshtml (in the folder named “layout” in the source code download) will produce the output shown in **Figure 6**. All sections were rendered correctly and emitted as pure HTML to the browser.

Help Me Help You

You hear it all the time in the software industry: reuse. WebMatrix introduces a concept similar to the plug-in model used in WordPress or jQuery. In WebMatrix, these plug-ins are known as helpers. The concept is similar: they’re published packages of reusable functionality, or little nuggets of .NET code that are purposely built and packaged for easy distribution and consumption. The helper package itself is merely a NuGet package (see nuget.codeplex.com). If you’re a .NET developer building and using helpers, it will feel no different than consuming any standard .NET class, because that’s all it really is.

Helpers are stored and retrieved from a cloud-based NuGet feed. So how do you find these elusive helpers? After you’ve opened up your site in WebMatrix, click Run. That opens your site in the browser, for example: <http://localhost:53655/default.cshtml>. Then browse to http://localhost:53655/_admin. This is the only underscored page that will render, and that’s just on your local machine. At this point, you’ll be prompted to create a password, after which you’ll be taken to the Package Manager (see **Figure 7**).

You can view helpers in one of three categories: *Installed*, *Online* and *Updates*. You can also view by the *source from* which they originated. That source is a NuGet feed. The

Figure 6 Output from Running default.cshtml

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title></title>
    Custom Script Here <br>
  </head>
  <body>
    <p>this was rendered from ~/layout/shared/_header.cshtml</p>
    Some body content
    <p>this was rendered from ~/layout/shared/_footer.cshtml</p>
  </body>
</html>
```

default feed is the official directory of WebMatrix helpers, but you could add other sources as well.

Now that we have the list of packages, let’s install one. For demonstration purposes, let’s install and explore a helper called the Url Shortener Helper (see UrlShortenerHelper.CodePlex.com). This is a helper I built to aid in shortening URLs, leveraging a provider such as bit.ly. The concept is simple: Take a URL and shorten it, hiding the interaction with the provider chosen to accomplish it. Now let’s switch to *Online* and select *Install* for the UrlShortener. This will download, install and configure the helper into your site.

Helpers are currently written in two ways—either directly in Razor or in a compiled assembly. Which should you choose? It depends, of course. In the case of my Url Shortener, the code is really just an interaction with an external provider API; it doesn’t emit any HTML. So in my scenario, the helper was better suited to be packaged in an assembly. The Facebook helper, on the other hand, emits a large amount of markup and, as such, is written as a CSHTML helper. For helpers that need to wrap API calls and emit markup, there’s also a hybrid approach in which you could do both—have some functionality in Razor and in an assembly.

You’ll find an example of this in the UserVoiceHelper at UserVoiceHelper.codeplex.com.

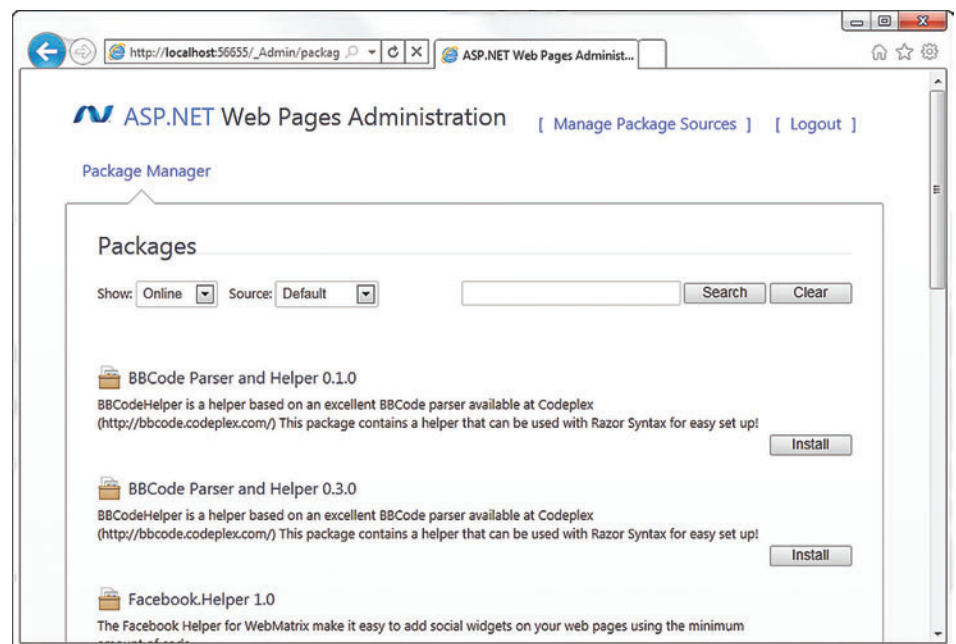


Figure 7 ASP.NET Web Pages Administration Package Manager

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.



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



Learn more at www.devtouchpro.com

More Development Tools Available at:

www.amyuni.com

All trademarks are property of their respective owners. © 1999-2010 AMYUNI Technologies. All rights reserved.

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

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

AMYUNI 
Technologies

Figure 8 Cars.cshtml Querying for Data

```
@{
    var db = Database.Open("Cars");
    var selectQueryString = "SELECT * FROM Models ORDER BY Year";
}
<!DOCTYPE html>
<html lang="en">
...
<tbody>
    @foreach (var row in db.Query(selectQueryString))
    {
        <tr>
            <td>@row.Year</td>
            <td>@row.Maker</td>
            <td>@row.Model</td>
        </tr>
    }
</tbody>
...
</html>
```

Looking back at our site, you now see a few new assets. Under the root of your site, there's now a `/bin` folder with two new assemblies. This happened as a result of NuGet installing our package. There will also be a `ReadMe.UrlShortener.txt` file in the root with instructions on how to use the helper. Other helpers, such as the Facebook helper, will actually add their own folders and code to the `App_Code` folder.

Data

It's pretty hard to have a Web site without having a data store behind it. WebMatrix ships with a new version of SQL Compact Server Compact Edition 4, or SQL Compact. It's a free embedded database engine that supports no installation and xcopy deployments.

Because SQL Compact is a version of SQL Server itself, you can always upgrade to more advanced versions of SQL Server with little work. WebMatrix even ships with a tool that will help you upgrade your SQL Compact database to SQL Server.

WebMatrix has built-in editor support for your database needs. From table creation to data creation, you can do it all from the comfy confines of WebMatrix. Once you've created your database and all of the necessary assets, it will all be contained in a database file with an SDF extension. **Figure 8** is a simple data-access call.

I start by opening the cars database and later query that connection in a foreach loop. In the loop, I display a result per row. After you render the page, send the page to source view and you'll again find pure HTML. Parts of this example were omitted for brevity but are available in the online source code in a file named `Data-Access.cshtml`.

Serving It Up

Your Web site is basically useless if you don't have something to serve it up. On the Windows platform, that has always meant IIS. For more information on IIS, visit iis.net. IIS has historically been a feature of the Windows OS and something that your hosting provider or IT professionals would configure and use. Web developers on Windows typically used IIS or the built-in Visual Studio Web server, called Cassini. With the release of WebMatrix and Visual Studio 2010 SP1, we now have IIS Express as an additional option. IIS Express is the best of both worlds—the power of IIS and the simplicity of Cassini.

IIS Express installs with WebMatrix by default, but you can download it separately with WebPI. Unlike the full version of IIS, IIS Express is installed just like any other user-mode program. You'll find it at `C:\Program Files (x86)\IIS Express`. It's comprised of two programs, `iisexpress.exe` and `iisexpressstray.exe`. The former is really IIS Express, and `iisexpressstray` is the visual tray that runs in the Windows system notification area.

Starting up IIS Express is typically done in the command line (`cmd.exe`). There are a number of options you can pass in such as path, port, name, framework version and more. For all of the commands, type: `> iisexpress.exe /?` If I wanted to start up a Web server for my new WebMatrix site, outside of WebMatrix itself, I would just need to type the following:

```
> iisexpress.exe /path:"c:\MySite" /port:81
```

Then you could open your favorite Internet browser and browse to `http://localhost:81`, which would serve up your site located at `C:\MySite`.

WebMatrix hides all of this from you, but that power is there if you want it. Upon clicking Run, WebMatrix will start up IIS Express in the system

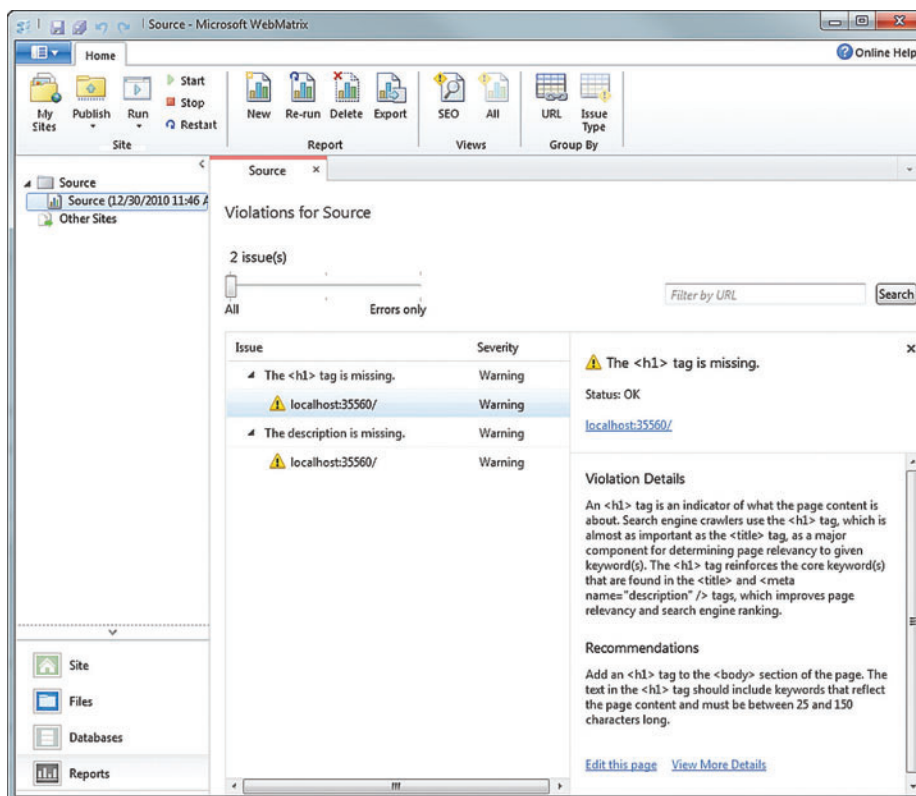


Figure 9 SEO Reporting



THE FUTURE OF AGILE IS...

A balanced blend of agile and traditional development

DevSuite - One platform for balanced development



Used by the world's largest development teams for:

- Managing agile and non-agile methods on a unified platform
- Integrated defect and quality management
- Wiki based requirement and knowledge management

Try DevSuite live. Watch a recorded overview. Request an online demo.

www.techexcel.com/TryDevSuite

1.800.439.7782

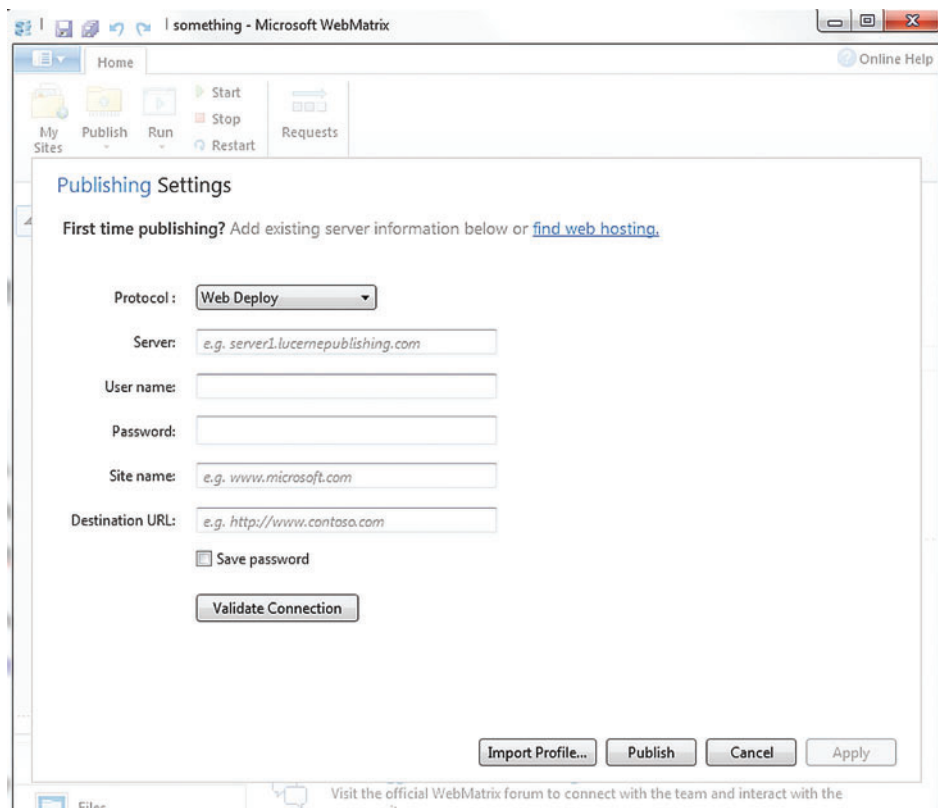


Figure 10 Publishing Settings

tray and configure it all on your behalf. When you shut down WebMatrix, IIS Express is shut down as well.

Search Relevance

Search relevance is more important now than ever. You can't just wait for Bing or Google to make you relevant. To aid you in your relevance quest, WebMatrix has a reporting feature called Search Engine Optimization (SEO).

Clicking on the reporting tab down in the lower-left-hand corner of WebMatrix will present you with the ability to create a new SEO Report. In doing so, WebMatrix crawls your site and looks for those potential areas where your site could be more relevant to search engines. Once WebMatrix has finished crawling your site, it will give you a detailed list of your violations as well as recommendations on how to resolve them. See **Figure 9** for a sample SEO Report that was run with the demo code for this article.

Deploying

You've spent countless hours perfecting your site. Now it's time to deploy it. WebMatrix does that in just two steps: configure and deploy. In the WebMatrix ribbon, you'll find a Publish button. Configuring your deployment settings is of course dependent on your hosting provider and what it has enabled. **Figure 10** shows the publish settings dialog. Out of the box, WebMatrix supports both *FTP* and *WebDeploy*. Enter your appropriate settings and click Publish.

Don't have a hosting provider? Click Find Hosting Provider. That will take you to asp.net/webmatrix/hosts, where you can choose a hosting provider that's already configured and certified to work with

WebMatrix. If you already had a site, WebMatrix has that covered, too. In that same Publish menu, you'll find Download Published Site. It's simply the reverse of deploying. It works off the same settings you configured to deploy. Once you've configured your publish settings, you can then download your current site locally to edit. When your edits are done, just redeploy.

Growing Up

When choosing a framework, wouldn't it be nice to know if it could scale with you as your needs and complexity increase? Over the years, ASP.NET has been building the LEGO pieces to do exactly that. Today we have an entire bucket full and we can snap them together as our needs demand it. In **Figure 1** you saw how everything is built on top of ASP.NET and, furthermore, .NET. This means that taking your WebMatrix application and migrating it to Visual Studio is in fact

possible. It's a click of a button.

Looking closer at the WebMatrix ribbon will reveal a button to open in Visual Studio located at the top-right-hand corner. This will open your site as an ASP.NET Web Site project in Visual Studio. But it doesn't stop at the core Web site. Chances are that you have a database, too. You can port any SQL Compact database to higher levels of SQL Server. WebMatrix will also aid you in this migration process by using the Migration button in the database menus.

Summing up, WebMatrix is designed to be a tool for everyone, and it complements Visual Studio well. How you leverage WebMatrix will depend on the task at hand, but nonetheless it's a tool in your toolbox. It's seamless to pick and choose what ASP.NET pieces you might leverage, and it makes migrating to Visual Studio and SQL Server seamless, too. The advent of helpers and the new layout subsystem have made reuse become a reality. If creating a site from scratch wasn't appealing, with WebMatrix you can start from a proven application listed in WebPI such as Umbraco or WordPress and deploy right to your hosting provider.

Unfortunately, there are many more features I couldn't cover in this article, including caching, routing and debugging. You can find all of the links used in this article at bit.ly/IntroToWebMatrix. ■

CLARK SELL works as a senior developer evangelist for Microsoft outside of Chicago. He blogs at csell.net, podcasts at DeveloperSmackdown.com and you can follow him on twitter.com/csell5.

THANKS to the following technical experts for reviewing this article:
Erik Porter, Mark Nichols and Brandon Satrom

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

Visit us at booth #1847

Microsoft®
tech·ed
North America | 2011

to win an XBOX

TX
TEXT CONTROL®
word processing components

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

WWW.TEXTCONTROL.COM

Microsoft®
Visual Studio
PARTNER

TX Text Control Sales:

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

Windows Phone Navigation, Part 2: Advanced Recipes

[This concludes the first installment of MSDN Magazine's newest monthly column, Mobile Matters. With the growing importance of mobile technologies such as Windows Phone 7, we'll regularly feature experts giving practical, hands-on advice to developers of all levels who want to increase their skills in this crucial technology space. We encourage you to help us shape our new column as we move forward. Please e-mail us at mmeditor@microsoft.com to suggest topics, share ideas or ask questions. We welcome all feedback. —Ed.]

Last month we introduced you to the Windows Phone navigation model and related APIs (see msdn.microsoft.com/magazine/gg650662). Now we'll share tips and tricks to accomplish more advanced navigation tasks. We'll address the following aspects of Windows Phone:

- Using the navigation APIs, you can use the `Navigate` and `GoBack` methods. These two methods either push or pop a URL to the navigation history (back stack). Because there are no APIs to manage the back stack, you have to manage transient screens yourself. Transient screens are dialogs or prompts that you don't want users to revisit when they hit Back—for example, a message prompt or a login dialog.
- Windows Phone allows only one navigation to be active at a time. If you need the “take me home” feature, which is usually a shortcut for a user to navigate back by multiple pages, you'll have to serialize the sequence of `GoBack` method calls one at a time; this can affect your UX.
- Neither of the two navigation participants—`PhoneApplicationPage` and `PhoneApplicationFrame`—have out-of-the-box support for page transitions.

Let's dissect these aspects so we can arrive at actionable recipes to address them.

Transient Content

Transient screens are screens that shouldn't be added to the journaling back stack (the journaling history)—for example, a login dialog. Users navigate from a page to a login dialog, but when they move forward and hit Back, you don't want users to go back to the dialog, you want them to go back to the previous screen.

There are multiple ways to create and display a transient screen. The obvious one is to use a Silverlight Popup to show the UI. There are two small issues with the Popup approach:

1. A Popup isn't orientation-aware. This isn't a huge problem; you could write an orientation-aware Popup or you could just insert the Popup into the visual tree and it would then do layout according to the page orientation.
2. Content inside a full-screen Popup isn't hardware-accelerated. For the login dialog UI (and for many transient pages) this will be fine, but imagine where the Popup has a `ListBox` with enough items to scroll through, or it has an animated `ProgressBar`—you'll want to hardware accelerate the Popup to maximize your UI thread's responsiveness.

Windows Phone allows
only one navigation to be active
at a time.

Because of these small issues, we recommend you inline the UI by inserting it into the visual tree with a higher z-index than the content of the page you're in. Here's the step-by-step recipe:

1. Package the content you want to inline into the visual tree in a User control so you can insert it into the tree and show it in your page (without it being mixed into the UI of your page).
2. In the `PhoneApplicationPage` that will show the UI, handle the `OnBackKeyPress` callback or the `BackKeyPress` event to dismiss the transient UI (without navigating back) when the Back button is pressed.
 - a. Custom controls that have multiple states (expanded and collapsed) should expose properties so that the host page can query into their state and know if they're in a state that can consume the Back button press (if applicable). For example, `ListPicker` on the control toolkit has a `ListPickerMode` property, `ContextMenu` has an `IsOpen` property and so on.
3. Hide the `ApplicationBar` in the page when displaying the transient UI (if applicable). The `ApplicationBar` is drawn by the OS and when it's `Opaque` (`Opacity = 1.0`) it reserves space below the content area for your page. To simulate your transient UI as being a new screen, you likely want to hide the application bar.

Code download available at code.msdn.microsoft.com/mag201103Mobile.



ActiveReports 6 | from \$685.02

GrapeCity PowerTools

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 for Windows Forms/WPF | 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 and WPF rich 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 PowerTools

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

Figure 1 A Summary of Animations

Animation	Usage	Directions	Transition Notes
Turnstile	Takes user from one space to another. Defaults across device. It's heavy by design to emphasize that a transition happened.	ForwardIn, ForwardOut, BackwardIn and BackwardOut	This will be a common animation you'll want to use. It's fairly straightforward.
Continuum	Transitions user from one space to another, but gives the perception of continuity. Carries context from one space to the other. Almost feels like you didn't leave.	In, Out	Continuum is common, but harder to implement. It requires carrying the continuum context (the perception that a UIElement is carried across two pages).
Swivel	Used for transient UI; for example, for dialogs. It's different from other animations in that it doesn't transition, but keeps user in same space or at least aims to give that perception.	ForwardIn, ForwardOut, FullScreenIn, FullScreenOut, BackwardIn, BackwardOut	Not used much for transitions; mostly used for dialogs.
Slide	Used for transient UI. Brings content over the existing content.	SlideUpFadeIn, SlideUpFadeOut, SlideDownFadeIn, SlideDownFadeOut, SlideLeftFadeIn, SlideLeftFadeOut, SlideRightFadeIn, SlideRightFadeOut	Commonly used to transition in transient UIs.
Rotate	Rotates the screen in a specific direction and angle.	In90Clockwise, In90CounterClockwise, In180Clockwise, In180CounterClockwise, Out90Clockwise, Out90CounterClockwise, Out180Clockwise, Out180CounterClockwise	Not used much for transitions; mostly used for orientation.

4. Animate the transient UI when it transitions in and when it's dismissed so that it blends with the rest of your application's UX (if applicable).

Code walkthrough: To see a transient UI screen, check *TransientUISample.xaml.cs* in the accompanying code download, with step-by-step instructions as comments.

Most of the preceding requirements should be intuitive; the only one that needs explanation is No. 2.a. To meet the certification requirements, a *PhoneApplicationPage* showing a transient screen shouldn't navigate back when the transient UI is showing. Instead, it should dismiss the transient UI.

There are multiple ways to create and display a transient screen.

In part 1 of this article, we mentioned that you should prevent hardware Back navigations by handling *OnBackKeyPress* or the *BackKeyPress* event. If your page is hosting a control that has two states and one of them is transient, then a bit of coordination (or at least awareness) between the page and the control is required. To implement the handshake, we recommend:

1. Custom controls that have multiple states (for example, *ListPicker*) should subscribe to *BackKeyPress* on their hosting page and handle it appropriately. If they're in a transient state, they should cancel the Back key press and dismiss their transient state.
 - a. These controls should subscribe only when needed. Delay subscription to the *BackKeyPress* event until needed.
 - b. We prefer the controls walk up the visual tree to find their *PhoneApplicationPage*, but we've seen experts (for example, the control toolkit team) do it by reaching out to the *Application RootVisual* (which we know is a

PhoneApplicationFrame) and peeking into its content. This is, of course, more performant and still safe because we know there's only one UI thread in a Silverlight app (so it's unlikely the *RootVisual* is changing inside an event handler for your control or your page).

2. Custom controls and transient UIs should expose properties so that the host page can query into their state and know if they're in a state that can consume the Back button press. For example, *ListPicker* on the control toolkit has a *ListPickerMode* property, *ContextMenu* has an *IsOpen* property and so on.
3. Your host *PhoneApplicationPage* will need to be aware of any transient controls or UI that can be displayed and check on all of these before it does any navigation within *OnBackKeyPress*. Remember, *OnBackKeyPress* is called before any handlers to *BackKeyPress* event are called. If you're relying on the hosted controls to handle their transient state, make sure you don't call the *Navigate* method and get ahead of the handlers.

Home Button Navigation (and Clearing the Back Stack)

A common UX pattern for other mobile platforms is to have a "Home" button that shortcuts navigation and sends you to a specific page. The Windows Phone navigation APIs don't directly handle this; in fact, the UX guidelines strongly discourage having a Home button. The recommendation is to have a well-designed, relatively flat navigation structure. If your navigations are two or three levels deep, it's probably just as easy to press the hardware Back button a couple times as it is to make a more calculated gesture finding a Home button on the screen and clicking it. That said, if you choose to have a Home button, there are a few things you must consider to implement the navigation:

- Remember that Windows Phone allows one navigation at a time. This means if you're four levels deep in a navigation,

Figure 2 The Four Page Transitions that Can Be Specified

Transition (Property Name)	Description
NavigationInTransition.Forward	Called as you Navigate to this page, with a forward navigation.
NavigationInTransition.Backward	Called when the user triggers a back navigation that's navigating to this page.
NavigationOutTransition.Forward	Called as you Navigate away from this page in a forward navigation.
NavigationOutTransition.Backward	Called as you Navigate away from this page in a back navigation.

you'll have to call `GoBack` three times in a row to get home. Because concurrent navigations aren't allowed, you must wait until a navigation is completed (when the `Navigated` event fires) to start the next navigation. This "navigation sequence" can lead to brief delays if your page is taking a lot of time to load, or it can lead to "flickers" if your page is visible while you pop the back stack.

- The animations for the `AppBar` are implemented by the OS. This means if you're navigating in a loop across pages, your UI can again flicker if the `AppBar` is being animated to be shown on a page in your loop.

So, here's a recipe to solve the "Home" navigation challenge. You can see all this code implemented via the `BackNavigationHelper` class in the accompanying download:

1. Hide the `PhoneApplicationFrame` (`RootVisual` for the app) from the screen while you pop the back stack. This will prevent flickering of the UI (it will flicker once, but you won't see all your screens go back).
 - a. If your pages take too much time, you can show a full-screen `Popup` with an animation so the user knows what's happening.
 - b. You should also set a flag before a home navigation begins so your pages know not to do a lot of work in their `OnNavigatedTo` callback. If you're navigating back, these pages will be unloaded as the navigation is completed (so any UI work they're doing is thrown away).
2. Hide the `AppBar` (by setting its `IsVisible` property to `false`) from each page that's being unwound (if applicable).

Figure 3 Replacing the `RootVisual` Frame for Your Application by Editing the `App.xaml.cs`

```
private void InitializePhoneApplication()
{
    if (phoneApplicationInitialized)
        return;

    // Create the frame but don't set it as RootVisual yet; this allows the splash
    // screen to remain active until the application is ready to render.
    RootFrame = new Microsoft.Phone.Controls.TransitionFrame();
    RootFrame.Navigated += CompleteInitializePhoneApplication;

    // Handle navigation failures
    RootFrame.NavigationFailed += RootFrame_NavigationFailed;

    // Ensure we don't initialize again
    phoneApplicationInitialized = true;
}
```

- a. Note that this is only required for pages where the `AppBar` is 100 percent opaque (its `Opacity = 1.0`).
3. Unwind the stack.

- a. Call the `GoBack` method.
- b. Listen for `Navigated` (on the `RootVisual`) and call `GoBack` again, if you're not already on the "Home" page.

Reset everything back to normal when you've unwound the stack.

Code walkthrough: You can see examples of "take me home" navigation in the pages in the `HomeNavigationSamplePages` folder in the accompanying code download. The interesting code is, of course, on the `BackNavigationHelper` class, but our sample has a `BackNavConfig` page where you can select the options you want to enable in `BackNavigationHelper` (such as hiding the frame, hiding the application bar, validating journaling and so on).

Page Transitions

The final and most advanced navigation concept you might care about concerns page transitions. In the context of navigation, we think of a transition as the perception of a hand-off between a page that you're navigating from and the page that you're navigating to. The navigation is still happening and it's using the underlying navigation framework we've discussed earlier, but as the navigation happens, the content in the `PhoneApplicationPages` that are participating in the navigation is animated to simulate a hand-off.

The Windows Phone
Controls Toolkit has support
for transitions, and it has
Storyboards for the most
common transitions.

The recipe for transitions is a bit more complicated. In fact, just like grandma's lasagna, there isn't a single recipe—it will require some "tweaking" to your taste (and application needs). That said, we have a few tenets you should follow, along with a sample and code walkthrough, of course.

The tenets:

1. Follow all the Back key press principles described earlier. Again, transition is just an extension to navigation—not an excuse to fail certification.
2. Understand the context and purpose of a transition animation. Windows Phone has a well-defined set of transitions and most of these have a specific context or usage. You must understand the intended usage so you can implement the right transitions in your app. Jeff Arnold (the lead motion designer for Windows Phone) did a brief recording of all the animations and transitions; it's a must-watch to understand the purpose of each animation. You can find the video at bit.ly/eBTkjD.

3. Keep your transitions fast and short.
 - a. Remember a transition is both an “out” from a page and an “in” on a different page. These two phases will add up. Keep them short. We would say 300 ms total is a good upper limit.
 - b. Delay as much UI work as you can during a transition. If your page can avoid data binding or expensive layout operations, consider doing that. You can transition the screen and then quickly move to populating it afterwards.

Figure 1 shows a summary of animations, their usage, directions and relevant notes.

With the three tenets listed earlier in mind, we can now move to coding page transitions. The Windows Phone Controls Toolkit has support for transitions, and it has Storyboards for the most common transitions.

Note: To follow along with the rest of the article, you'll need the Silverlight for Windows Phone Toolkit (silverlight.codeplex.com). The code in this article is written against the February release (later releases should work, too).

Toolkit transitions use a `TransitionFrame` that inherits from `PhoneApplicationFrame` but has a customized template with two content presenters (where `PhoneApplicationFrame` has only one content presenter). `TransitionFrame` listens to changes to its `Content` property and transitions in the new `Content` (page) and transitions out the old content.

Each `PhoneApplicationPage` determines what transitions it wants using an attached property in the toolkit's `TransitionService`

Figure 4 Applying Transition Properties to Your Pages

```
<phone:PhoneApplicationPage
  x:Class="LWP.TransitionSamples.Turnstile"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  FontFamily="{StaticResource PhoneFontFamilyNormal}"
  FontSize="{StaticResource PhoneFontSizeNormal}"
  Foreground="{StaticResource PhoneForegroundBrush}"
  SupportedOrientations="Portrait" Orientation="Portrait"
  mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
  shell:SystemTray.IsVisible="True" xmlns:toolkit=
    "clr-namespace:Microsoft.Phone.Controls;assembly=
    Microsoft.Phone.Controls.Toolkit"
>

<toolkit:TransitionService.NavigationInTransition>
  <toolkit:NavigationInTransition>
    <toolkit:NavigationInTransition.Backward>
      <toolkit:TurnstileTransition Mode="BackwardIn"/>
    </toolkit:NavigationInTransition.Backward>
    <toolkit:NavigationInTransition.Forward>
      <toolkit:TurnstileTransition Mode="ForwardIn"/>
    </toolkit:NavigationInTransition.Forward>
  </toolkit:NavigationInTransition>
</toolkit:TransitionService.NavigationInTransition>
<toolkit:TransitionService.NavigationOutTransition>
  <toolkit:NavigationOutTransition>
    <toolkit:NavigationOutTransition.Backward>
      <toolkit:TurnstileTransition Mode="BackwardOut"/>
    </toolkit:NavigationOutTransition.Backward>
    <toolkit:NavigationOutTransition.Forward>
      <toolkit:TurnstileTransition Mode="ForwardOut"/>
    </toolkit:NavigationOutTransition.Forward>
  </toolkit:NavigationOutTransition>
</toolkit:TransitionService.NavigationOutTransition>
```

class. You can specify up to four transitions for each page, as shown in **Figure 2**.

These transitions are instances of an extensible `NavigationTransition` class. The toolkit includes five built-in `NavigationTransition`s: `TurnstileTransition`, `SlideTransition`, `SwivelTransition`, `RotateTransition` and `RollTransition`. Again, the system is extensible so you can add your own.

Windows Phone Silverlight applications have a Web-like navigation model in which you can transition from one page and have a journaling (or history) so you can go back to previous pages.

Step-by-step directions to implement transitions using the toolkit are:

1. Download and add a reference to the Silverlight control toolkit.
2. Replace the `RootVisual` frame for your application by editing the `App.xaml.cs` and replacing it with a `TransitionFrame` (see **Figure 3**).
3. Apply transition properties to your pages (see **Figure 4**).

Actionable Recipes

To sum up, Windows Phone Silverlight applications have a Web-like navigation model in which you can transition from one page and have a journaling (or history) so you can go back to previous pages. Out-of-the-box, the navigation model is easy to use and relatively complete. There are a few advanced navigation tasks—such as transient UIs, transitions or “jump to home” functionality—that aren't implemented out of the box, but this two-part article has explained the design considerations you should take into account to implement these more advanced tasks and has given you concise, actionable recipes to implement them. ■

YOCHAY KIRIATY is a senior technical evangelist at Microsoft, focusing on client technologies such as Windows and Windows Phone. He coauthored the books “Introducing Windows 7 for Developers” (Microsoft Press, 2009) and “Learning Windows Phone Programming” (O'Reilly Media, 2011).

JAIME RODRIGUEZ is a principal evangelist at Microsoft, driving adoption of emerging client technologies such as Silverlight and Windows Phone. Follow him on Twitter at twitter.com/jaimerodriguez or on blogs.msdn.com/jaimerr.

THANKS to the following technical expert for reviewing this article: Peter Torr

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



Lissajous Animations in Silverlight

We commonly think of software as more flexible and versatile than hardware. It's certainly true in many cases, because hardware is often stuck in one configuration while software can be reprogrammed to perform completely different tasks.

Yet some rather prosaic pieces of hardware are actually quite versatile. Consider the common—or these days, not so common—cathode ray tube (CRT). This is a device that shoots a stream of electrons on the inside of a glass screen. The screen is coated with a fluorescent material that reacts to these electrons by briefly glowing.

In old-time TV sets and computer monitors, the electron gun moves in a steady pattern, repetitively sweeping horizontally across the screen while traveling more slowly from top to bottom. The intensity of electrons at any time determines the brightness of a dot at that point. For color displays, separate electron guns are used to create the primary red, green and blue colors.

The direction of the electron gun is controlled by electromagnets, and it can actually be aimed at any arbitrary location on the two-dimensional surface of the glass. This is how the CRT is used in an oscilloscope. Most commonly, the beam sweeps horizontally across the screen at a constant rate, usually in synchronization with a particular input waveform. The vertical deflection shows the amplitude of that waveform at that point. The fairly long persistence of the fluorescent material used in oscilloscopes allows the entire waveform to be displayed—in effect “freezing” the waveform to be visually examined.

Oscilloscopes also have an X-Y mode that allows the horizontal and vertical deflection of the electron gun to be controlled by two independent inputs, usually waveforms such as sine curves. With two sine curves as input, at any point in time the point (x, y) is illuminated, where x and y are given by the parametric equations:

$$x(t) = A_x \sin(\omega_x t + k_x)$$

$$y(t) = A_y \sin(\omega_y t + k_y)$$

The A values are amplitudes, the ω values are frequencies and the k values are phase offsets.

The pattern resulting from the interaction of these two sine waves is a Lissajous curve, named after French mathematician Jules Antoine Lissajous (1822 - 1880) who first visually created these curves by bouncing light between a pair of mirrors attached to vibrating tuning forks.

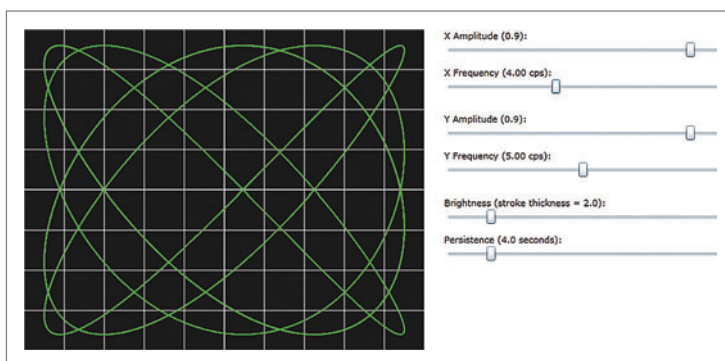


Figure 1 The Web Version of the LissajousCurves Program

You can experiment with a Silverlight program that generates Lissajous curves on my Web site (charlespetzold.com/silverlight/LissajousCurves/LissajousCurves.html). **Figure 1** shows a typical display.

Although it's not quite obvious in a static screenshot, a green point is moving around the dark gray screen and leaving behind a trail that fades out over four seconds. The horizontal position of this point is governed by one sine curve, and the vertical position by another. Repetitive patterns result when the two frequencies are simple integral ratios.

The pattern resulting from the interaction of these two sine waves is a Lissajous curve.

It's now a truth universally acknowledged that a Silverlight program of any good fortune must be ported to Windows Phone 7, and subsequently reveal any performance problems previously masked by high-powered desktop computers. That was certainly the case with this program, and I'll be discussing these performance issues later in this article. **Figure 2** shows the program running on the Windows Phone 7 emulator.

The downloadable code consists of a single Visual Studio solution named LissajousCurves. The Web application consists of the projects LissajousCurves and LissajousCurves.Web. The Windows Phone 7 application has the project name LissajousCurves.Phone. The solution also contains two library projects: Petzold.Oscilloscope.Silverlight and Petzold.Oscilloscope.Phone, but these two projects share all the same code files.

Code download available at code.msdn.microsoft.com/mag201104UIFrontiers.

PRECISELY PROGRAMMED FOR SPEED

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

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



DynamicPDF

[WWW.DYNAMICPDF.COM](http://www.DynamicPDF.com)

TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.681.5008 | +1 410.772.8620

ceTe software

Push or Pull?

Aside from the TextBlock and Slider controls, the only other visual element in this program is a class named Oscilloscope that derives from UserControl. Providing the data for Oscilloscope are two instances of a class named SineCurve.

SineCurve has no visuals itself, but I derived the class from FrameworkElement so I could put the two instances in the visual tree and define bindings on them. In fact, everything in the program is connected with bindings—from the Slider controls to the SineCurve elements and from SineCurve to Oscilloscope. The MainPage.xaml.cs file for the Web version of the program has no code beyond what's provided by default, and the equivalent file in the phone application only implements Tombstoning logic.

SineCurve defines two properties (backed by dependency properties) named Frequency and Amplitude. One SineCurve instance provides the horizontal values for Oscilloscope, and the other the vertical values.

The SineCurve class also implements an interface I called IProvideAxisValue:

```
public interface IProvideAxisValue {  
    double GetAxisValue(DateTime dateTime);  
}
```

SineCurve implements this interface with a rather simple method that references two fields as well as the two properties:

```
public double GetAxisValue(DateTime dateTime) {  
    phaseAngle += 2 * Math.PI * this.Frequency *  
        (dateTime - lastDateTime).TotalSeconds;  
    phaseAngle %= 2 * Math.PI;  
    lastDateTime = dateTime;  
  
    return this.Amplitude * Math.Sin(phaseAngle);  
}
```

The Oscilloscope class defines two properties (also backed by dependency properties) named XProvider and YProvider of type IProvideAxisValue. To get everything moving, Oscilloscope installs a handler for the CompositionTarget.Rendering event. This event is fired in synchronization with the refresh rate of the video display and thus serves as a convenient tool for performing animations. On each call to the CompositionTarget.Rendering handler, Oscilloscope calls GetAxisValue on the two SineCurve objects set to its XProvider and YProvider properties.

In other words, the program implements a pull model. The Oscilloscope object determines when it needs data and then pulls the data from the two data providers. (How it displays that data is something I'll discuss soon.)

As I began to add more features to the program—in particular, two instances of an additional control that displayed the sine curves, but that I eventually removed as an unenlightening distraction—I began to doubt the wisdom of this model. I had three objects pulling the same data from two providers, and I thought maybe a push model would be better.

I restructured the program so that the SineCurve class installed a handler for CompositionTarget.Rendering and pushed data to the

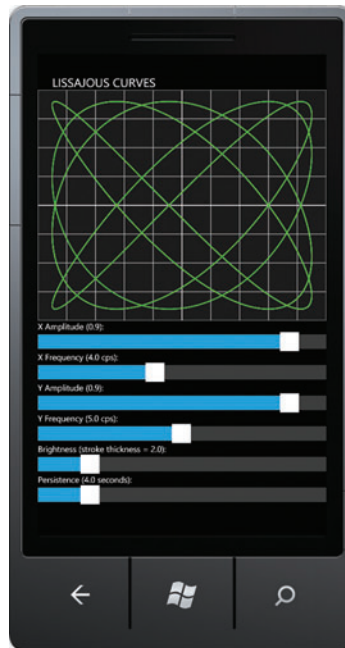


Figure 2 The LissajousCurves Program for Windows Phone 7

Oscilloscope control through properties now named simply X and Y of type double.

I probably should've anticipated the fundamental flaw in this particular push model: The Oscilloscope was now receiving two separate changes in X and Y and constructing not a smooth curve but a series of stair steps, as shown in Figure 3.

Making the decision to go back to the pull model was easy!

Rendering with WriteableBitmap

From the moment I conceived this program, there was absolutely no question in my mind that using WriteableBitmap was the best solution to implement the actual Oscilloscope screen.

WriteableBitmap is a Silverlight bitmap that supports pixel addressing. All the pixels of the bitmap are exposed as an array of 32-bit integers. Programs can obtain and set these pixels in an arbitrary manner. WriteableBitmap also has a Render method that allows rendering the visuals of any object of type FrameworkElement onto the bitmap.

If Oscilloscope just needed to display a simple static curve, I'd use Polyline or Path and wouldn't even consider WriteableBitmap. Even if that curve needed to change shape, Polyline or Path would still be preferable. But the curve displayed by Oscilloscope needs to grow in size, and it needs to be colored oddly. The line needs to fade out progressively: Recently displayed parts of the line are brighter than older parts of the line. If I used a single curve, it would need various colors along its length. This is not a concept supported under Silverlight!

A Silverlight program of any good fortune must be ported to Windows Phone 7.

Without WriteableBitmap, the program would need to create several hundred different Polyline elements, all colored differently and juggled around, and triggering layout passes after each CompositionTarget.Rendering event. Everything I knew about Silverlight programming indicated that WriteableBitmap would definitely offer much better performance.

An early version of the Oscilloscope class processed the CompositionTarget.Rendering event by obtaining new values from the two SineCurve providers, scaling those to the size of the WriteableBitmap, and then constructing a Line object from the previous point to the current point. That was simply passed to the Render method of WriteableBitmap:

```
writeableBitmap.Render(line, null);
```

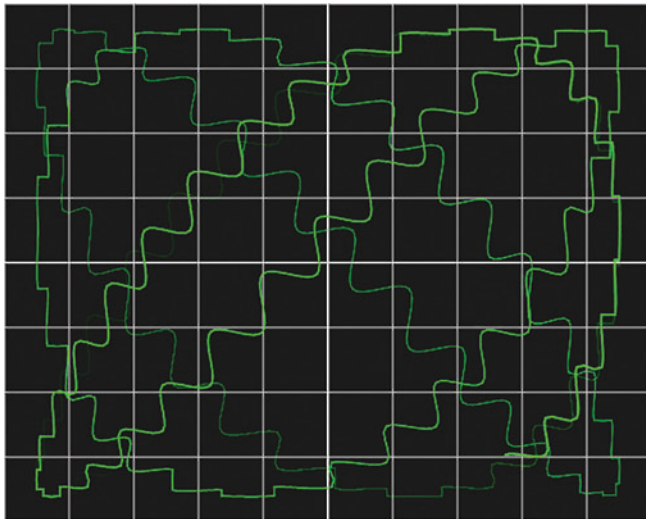



Figure 3 The Disastrous Result of a Push-Model Experiment

The Oscilloscope class defines a Persistence property that indicates the number of seconds for any color or alpha component of a pixel to decrease from 255 to 0. Making those pixels fade out involved direct pixel addressing. The code is shown in **Figure 4**.

At this point in the development of the program, I took the steps necessary to also get it running on the phone. On both the Web and the phone, the program seemed to run well, but I knew it was not quite finished. I wasn't seeing curves on the Oscilloscope screen: I was seeing a bunch of connected straight lines. And nothing destroys the illusion of digitally simulated analog faster than a bunch of extremely straight lines!

Interpolation

The CompositionTarget.Rendering handler is called in synchronization with the video display refresh. For most video displays—

Figure 4 Code to Fade out Pixel Values

```
accumulatedDecrease += 256 *
    (dateTime - lastDateTime).TotalSeconds / Persistence;
int decrease = (int)accumulatedDecrease;

// If integral decrease, sweep through the pixels
if (decrease > 0) {
    accumulatedDecrease -= decrease;

    for (int index = 0; index <
        writeableBitmap.Pixels.Length; index++) {

        int pixel = writeableBitmap.Pixels[index];

        if (pixel != 0) {
            int a = pixel >> 24 & 0xFF;
            int r = pixel >> 16 & 0xFF;
            int g = pixel >> 8 & 0xFF;
            int b = pixel & 0xFF;

            a = Math.Max(0, a - decrease);
            r = Math.Max(0, r - decrease);
            g = Math.Max(0, g - decrease);
            b = Math.Max(0, b - decrease);

            writeableBitmap.Pixels[index] = a << 24 | r << 16 | g << 8 | b;
        }
    }
}
```

including the display on Windows Phone 7—this is usually in the region of 60 frames per second. In other words, the CompositionTarget.Rendering event handler is called approximately every 16 or 17 ms. (Actually, as you'll see, that's only the optimum situation.) Even if the sine waves are a leisurely one cycle per second, for a 480-pixel-wide oscilloscope, two adjacent samples might have pixel coordinates some 35 pixels apart.

The Oscilloscope needed to interpolate between consecutive samples with a curve. But what kind of curve?

My first choice was a canonical spline (also known as a cardinal spline). For a sequence of control points p_1 , p_2 , p_3 and p_4 , the canonical spline provides a cubic interpolation between p_2 and p_3 with a degree of curviness based on a “tension” factor. It's a general-purpose solution.

The canonical spline was supported in Windows Forms, but never made it into Windows Presentation Foundation (WPF) or Silverlight. Fortunately, I had some WPF and Silverlight code for the canonical spline that I developed for a 2009 blog entry called, appropriately enough, “Canonical Splines in WPF and Silverlight” (bit.ly/bDaWgt).

After generating a Polyline with interpolation, the CompositionTarget.Rendering processing now concluded with a call like this:

```
writeableBitmap.Render(polyline, null);
```

The Oscilloscope needed to interpolate between consecutive samples with a curve.

The canonical spline worked, but it was not quite right. When the frequencies of the two sine curves are simple integral multiples, the curve should stabilize into a fixed pattern. But that wasn't happening, and I realized that the interpolated curve was slightly different depending on the actual sampled points.

This problem was exacerbated on the phone, mostly due to the tiny phone processor having trouble keeping up with all the demands I was putting on it. At higher frequencies, the Lissajous curves on the phone looked smooth and curvy, but seemingly moving in almost random patterns!

Only slowly did I realize that I could interpolate based on time. Two consecutive calls to the CompositionTarget.Rendering event handler are about 17 ms apart. I could simply loop through all these intermediate millisecond values and call the GetAxisValue method in the two SineCurve providers to construct a smoother polyline.

That approach worked much better.

Improving Performance

One piece of essential reading for all Windows Phone 7 programmers is the documentation page “Performance Considerations in Applications for Windows Phone” at bit.ly/fdvh7Z. Aside from many helpful hints about improving performance in your phone applications, it will also tell you the meaning of those numbers that are displayed at the side of the screen when you run the program under Visual Studio, as shown in **Figure 5**.

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



New in Team 2.11

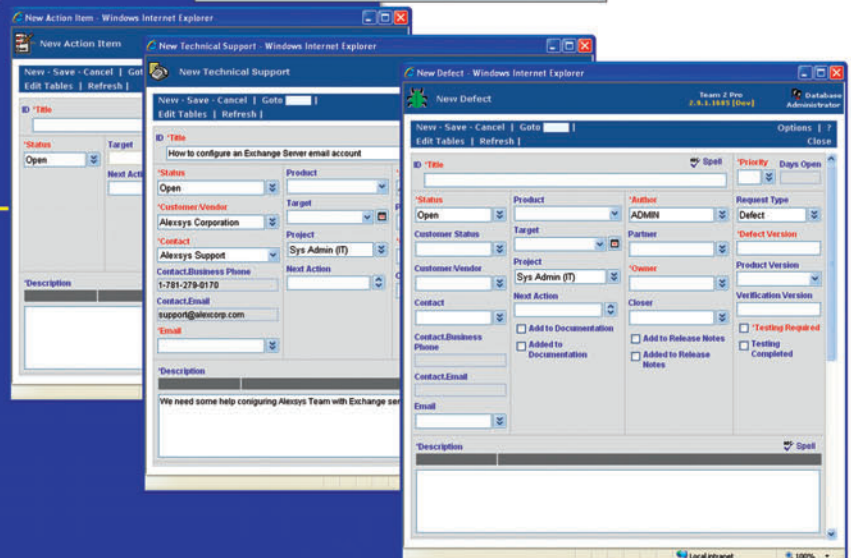
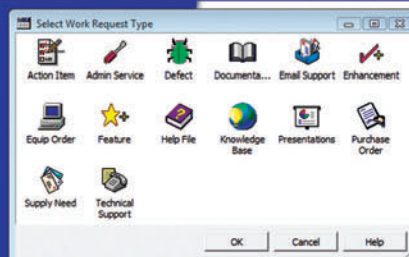
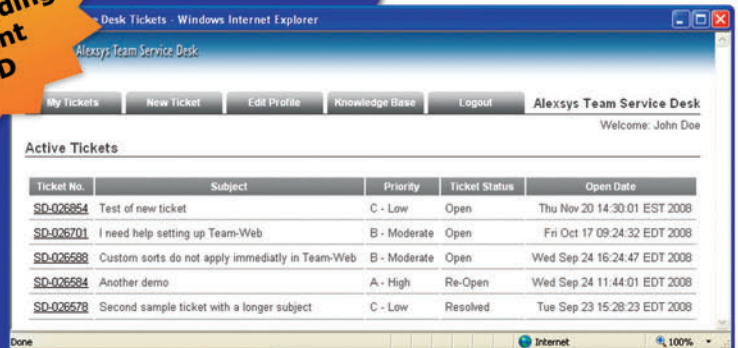
- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment

Service Desk Features

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

Team 2 Features

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



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

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

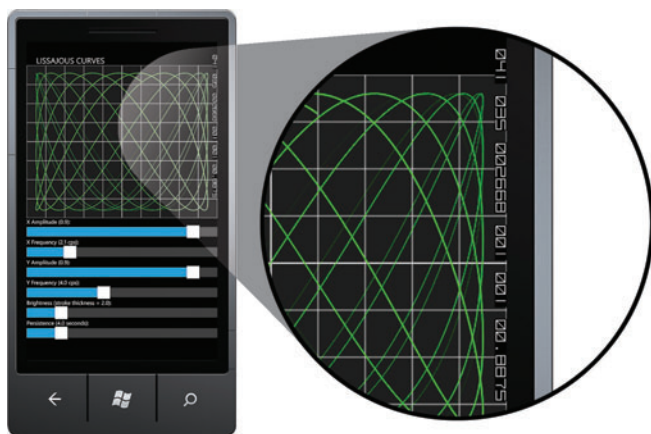


Figure 5 Performance Indicators in Windows Phone 7

This row of numbers is enabled by setting the `Application.Current.Host.Settings.EnableFrameRateCounter` property to true, which the standard `App.xaml.cs` file does if the program is running under the Visual Studio debugger.

The first two numbers are the most significant: Sometimes, if nothing is going on, these two numbers display zero, but they're both intended to display frame rates—which means they display a number of frames per second. I mentioned that most video displays are refreshed at the rate of 60 times per second. However, an application program might attempt to perform animations where each new frame requires more than 16 or 17 ms of processing time.

For example, suppose a `CompositionTarget.Rendering` handler requires 50 ms to do whatever job it's doing. In that case, the program will be updating the video display at the rate of 20 times per second. That's the program's frame rate.

Once the frame rate drops to 15 or 10, it's going to start being noticeable.

Now 20 frames per second is not a terrible frame rate. Keep in mind that movies run at 24 frames per second, and standard television has an effective frame rate (taking interlacing into account) of 30 frames per second in the United States, and 25 in Europe. But once the frame rate drops to 15 or 10, it's going to start being noticeable.

Silverlight for Windows Phone is able to offload some animations to the Graphics Processing Unit (GPU), so it has a secondary thread (sometimes referred to as the composition or GPU thread) that interacts with the GPU. The first number is the frame rate

Figure 6 Frame Rates for the Four Oscilloscope Updating Methods

	Composition Thread	UI Thread
WriteableBitmap with Polyline render	22	11
WriteableBitmap with manual outline fills	0	0
Canvas with Polyline with animation fade-out	20	20
Canvas with Polyline with manual fade-out	31	15

associated with that thread. The second number is the UI frame rate, which refers to the application's primary thread. That's the thread that any `CompositionTarget.Rendering` handlers run in.

Running the `LissajousCurves` program on my phone, I saw numbers of 22 and 11, respectively, for the GPU and UI threads, and they dropped down a bit when I increased the frequency of the sine curves. Could I do better?

I began to wonder how much time this crucial statement in my `CompositionTarget.Rendering` method required:

```
writeableBitmap.Render(polyline, null);
```

This statement should have been called 60 times per second with a polyline consisting of 16 or 17 lines, but it was actually being called more like 11 times per second with 90-segment polylines.

For my book, "Programming Windows Phone 7" (Microsoft Press, 2010), I wrote some line-rendering logic for XNA, and I was able to adapt that for Silverlight for this `Oscilloscope` class. Now I wasn't calling the `Render` method of `WriteableBitmap` at all, but instead directly altering pixels in the bitmap to draw the polylines.

Unfortunately, both frame rates plunged to zero! This suggested to me that Silverlight knew how to render lines on a bitmap much faster than I did. (I should also note that my code was not optimized for polylines.)

At this point, I began wondering if an approach other than `WriteableBitmap` might be reasonable. I substituted a `Canvas` for the `WriteableBitmap` and `Image` element, and as each `Polyline` was constructed, I simply added that the `Canvas`.

Of course, you can't do this indefinitely. You don't want a `Canvas` with hundreds of thousands of children. And besides, these `Polyline` children needed to fade out. I tried two approaches: The first involved attaching a `ColorAnimation` to each `Polyline` to decrease the alpha channel of the color, and then removing the `Polyline` from the `Canvas` when the animation completed. The second was a more manual approach of enumerating through the `Polyline` children, decreasing the alpha channel of the color manually, and removing the child when the alpha channel got down to zero.

These four methods still exist in the `Oscilloscope` class, and they're enabled with four `#define` statements at the top of the C# file. **Figure 6** shows the frame rates with each approach.

Figure 6 tells me that my original instinct about `WriteableBitmap` was wrong. In this case, it's really better to put a bunch of `Polyline` elements in a `Canvas`. The two fade-out techniques are interesting: When performed by an animation, the fading out occurs in the composition thread at 20 frames per second. When performed manually, it's in the UI thread at 15 frames per second. However, adding new `Polyline` elements always occurs in the UI thread, and that frame rate is 20 when fading-out logic is off-loaded to the GPU.

In conclusion, the third method has the best overall performance.

So what have we learned today? Clearly, to eke out the best performance, it's necessary to experiment. Try different approaches, and never, ever trust your initial instincts. ■

CHARLES PETZOLD is a longtime contributing editor to MSDN Magazine. His new book, "Programming Windows Phone 7" (Microsoft Press, 2010), is available as a free download at bit.ly/cpebookpdf.

THANKS to the following technical expert for reviewing this article: Jesse Liberty

AGENDA

VISUAL STUDIO LIVE! LAS VEGAS TRACKS

Silverlight/WPF	Programming Practices	Visual Studio 2010/.NET 4	WCF	Cloud Computing	Data Management	Web/HTML 5	"Simplification" Tools	Mobile Development
-----------------	-----------------------	---------------------------	-----	-----------------	-----------------	------------	------------------------	--------------------

Visual Studio Live! Pre-Conference Workshops: Monday, April 18, 2011

(Separate entry fee required)

MWK1 An Introduction to Multi-Platform Mobile Development Using C#: iPhone, Android, and Windows Phone 7 *Ken Getz & Brian Randell*

MWK2 Workshop: Making Effective Use of Silverlight and WPF *Billy Hollis & Rockford Lhotka*

MWK3 Workshop: Programming with WCF in One Day *Miguel Castro*

Visual Studio Live! Day 1: Tuesday, April 19, 2011

T1 Easing in to Windows Phone 7 Development *Walt Ritscher*

T2 Getting Started with ASP.NET MVC *Philip Japikse*

T3 Azure Platform Overview *Vishwas Lele*

T4 Best Kept Secrets in Visual Studio 2010 and .NET 4.0 *Deborah Kurata*

T5 Silverlight in 75 Minutes *Ken Getz*

T6 Test Driving ASP.NET MVC2 *Philip Japikse*

T7 Building Azure Applications *Vishwas Lele*

T8 How We Do Language Design at Microsoft *Lucian Wischik*

TCT1 Chalk Talk: Silverlight, WCF RIA Services, and Your Business Objects *Deborah Kurata*

TCT2 Chalk Talk: Building N-Tier Applications With Entity Framework 4 *Leonard Lobel*

TCT3 Chalk Talk: Join the XAML Revolution *Billy Hollis*

T9 Transitioning from Windows Forms to WPF *Miguel Castro*

T10 HTML5/IE9 inspire

T11 Building Compute-Intensive Apps in Azure *Vishwas Lele*

T12 Turn Your Development Up to 11: Debugging to Win with Visual Studio 2010 *Brian Randell*

T13 Programming for Windows 7 with WPF *Miguel Castro*

T14 Improving Your ASP.NET Application Performance with Asynchronous Pages and Actions *Tiberiu Covaci*

T15 Using C# and Visual Basic to Build a Cloud Application for Windows Phone 7 *Lucian Wischik & Srivatsn Narayanan*

T16 Designing and Developing for the Rich Mobile Web *Joe Marini*

Visual Studio Live! Day 2: Wednesday, April 20, 2011

W1 Bind Anything to Anything in Silverlight and WPF *Rockford Lhotka*

W2 HTML 5 and Your Web Sites *Robert Boedigheimer*

W3 How to Make Your Application Awesome with JSON, REST, WCF and MVC *James Bender*

W4 Visual Studio LightSwitch—Beyond the Basics *Robert Green*

W5 Design, Don't Decorate *Billy Hollis*

W6 Styling Web Pages with CSS 3 *Robert Boedigheimer*

W7 RESTBuilding RESTful Services in the Microsoft Platform: When to Use What? *Jesus Rodriguez*

W8 Advanced LightSwich Development *Michael Washington*

WCT1 Chalk Talk: CSLA .NET *Rockford Lhotka*

WCT2 Chalk Talk: Busy Developer's Guide to (ECMA/Java)Script *Ted Neward*

W9 Leveraging the MVVM Pattern in Silverlight, WPF and Windows Phone *Rockford Lhotka*

W10 HTML5 Messaging, Web Workers and Web Sockets with JavaScript *Jeffrey McManus*

W11 WCF Workflow Services *Rob Daigneau*

W12 The Almighty @—A Razor Primer *Charles Nurse*

W13 Top 7 Lessons Learned On My First Big Silverlight Project *Benjamin Day*

W14 jQuery Application Development *Jeffrey McManus*

W15 WCF Tips & Tricks – From the Field *Christian Weyer*

W16 WebMatrix Real World Data-Centric Applications *Charles Nurse*

Visual Studio Live! Day 3: Thursday, April 21, 2011

TH1 Multi-touch Madness! *Brian Peek*

TH2 The Best of jQuery *Robert Boedigheimer*

TH3 Making WCF Simple: Best Practices for Testing, Deploying and Managing WCF Solutions in the Big Enterprise *Jesus Rodriguez*

TH4 Digging Deeper in Windows Phone 7 *Walt Ritscher*

TH5 XAML Primer Clarifying the UI Markup Language *Walt Ritscher*

TH6 Single Sign-On for ASP.NET Applications *Dominick Baier*

TH7 How to Take WCF Data Services to the Next Level *Rob Daigneau*

TH8 C# on Android: Building Android Apps with .NET *Christian Weyer*

TH9 Silverlight Security *Dominick Baier*

TH10 The Scrum vs. Kanban Cage Match *Benjamin Day & David Starr*

TH11 Busy .NET Developer's Guide to Parallel Extensions for .NET 4 *Ted Neward*

TH12 C# on Android: Building Android Apps with .NET *Christian Weyer*

TH13 LINQ Programming Model *Marcel de Vries*

TH14 Patterns of Healthy Teams using Visual Studio and TFS *David Starr*

TH15 Designing Applications in the Era of Many-Core Computing *Tiberiu Covaci*

TH16 XNA Games for Windows Phone 7 *Brian Peek*

TH17 So Many Choices, So Little Time: Understanding Your .NET 4.0 Data Access Options *Leonard Lobel*

TH18 Produce Better Quality Code by Leveraging the Visual Test Tools You Never Discovered Before *Marcel de Vries*

TH19 Building Event-Driven Applications with Microsoft StreamInsight *Torsten Grabs*

TH20 Windows Azure and PHP *Jeffrey McManus*

Visual Studio Live! Post-Conference Workshops: Friday, April 22, 2011

(Separate entry fee required)

FWK1 Architectural Katas Workshop *Ted Neward*

FWK2 SQL Server 2011 *Andrew Brust & Leonard Lobel*

VISIT US ONLINE AT **VSLIVE.COM/LV** FOR DETAILS ON SESSIONS, SPEAKERS, AND MORE!

JOIN US IN LAS VEGAS THIS MONTH.

If you're looking for hard-hitting .NET development training, look no further than Visual Studio Live! Las Vegas. Our goal? To arm you with the knowledge to build better applications.

CHECK OUT THE FULL 50+ SESSION SCHEDULE NOW!

VISUAL STUDIO LIVE! LAS VEGAS is 5 days packed with full day workshops, keynotes from industry heavy weights and your choice of 50 hard-hitting sessions.

You'll learn tips, tricks and fixes from .NET pros like Billy Hollis, Rockford Lhotka, Andrew Brust, Deborah Kurata and Dave Mendlen, Senior Director, Developer Platform and Tools at Microsoft.

**DOWNLOAD
THE AGENDA
NOW!**



Get the free mobile app at
<http://gettag.mobi>



**REGISTER NOW
WITH CODE APRAD**

SUPPORTED BY:



PRODUCED BY:



WWW.VSLIVE.COM/LV

The Cat Butt Factor

You probably don't think about my butt when you design software. But you should.

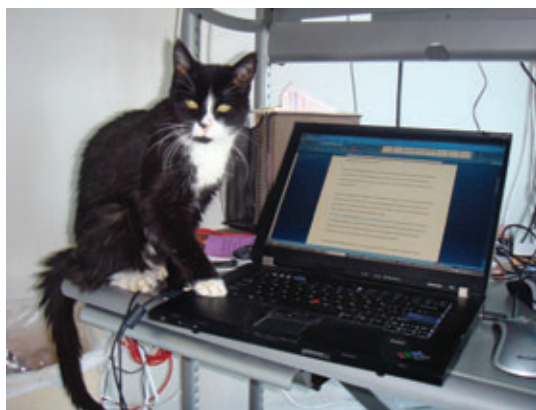
Hi, I'm Simba, Dave's long-suffering cat. Dave's really busy this month, so he asked me to help out. He said I could either write this column or he'd get me a job in the exciting field of medical research. Anyone who knows Dave will understand what a tough call that was, either living with him or being dissected alive. But I've always wanted to see my name in print, so I decided to tolerate him a little longer.

My picture doesn't show it, but I'm almost 20 years old. I remember when Dave got his first 386 PC from Gateway, with cow spots on the box. It ran this funny "Windows" thing when it started up. I thought the mouse was really cool. But when I bit its tail, I found a really shocking experience. Dave tried to uninstall Windows, but stopped when I showed him Solitaire.

That was back when Windows only ran on PCs. Cell phones were these big clunky things with horrible voice quality that dropped calls all the time. Now Windows runs on sleek, light phones that play music and games, give driving directions and even show movies. And they still have horrible voice quality and drop calls all the time; but now, at least, we have apps that use GPS to locate the nearest working pay phone when we really have to make a call. That's progress, no?

What does this have to do with my butt, I hear you wondering. The answer is that, like many users of Windows, Dave leaves his laptop on overnight because it takes so long to boot up. Naturally, the keyboard is where I like to sleep. It's warm from the heat sink, and the keys make a soft bed for my old joints. But I do it mostly because I know he doesn't want me there.

Simultaneously holding down multiple keys doesn't just trash whatever document Dave foolishly left open; it often scrambles the connection between the hardware and the OS in ways that are very hard to diagnose and fix after he chases me off it (which serves him right, don't you think?). One time the keyboard emitted weird clicks but wouldn't echo any characters. Another time the screen image turned upside down and stayed that way. Still another time, nothing initially seemed wrong, but the keyboard layout changed



You might think that my not having opposable thumbs would make this column difficult to write. You would be wrong.

to what Dave eventually identified as Indonesian. He was tearing out his face fur. I haven't seen him so angry since he called tech support for my self-scooping USB-controlled litter box and they put him on hold. I thought I'd have to take him to the people-vet.

He's supposed to know this stuff. He's written more than a dozen books on programming Windows, he teaches it at a major university and Microsoft designated him a Software Legend back in 2002. And a cat's butt has made him repave his hard drive, more than once. He thinks it's wrong. I think it shows how we cats really run the world.

Think Dave's just being extra dumb, even for him? Nope. Approximately 38 million U.S. households—about a third of the total—own at least one cat. The average number owned per household is about 2.5, for a total of 94 million owned cats in the United States. And of those 10⁺ feline beasts, exactly zero owners have any control whatsoever over where the cat goes or what the cat does in the house. This isn't an obscure edge case.

I thought the mouse was really cool. But when I bit its tail, I found a really shocking experience.

Legend says that the Internet was designed to survive a nuclear war. Your programs ought to survive the attack of a cat's butt. Maybe Dave will rent me to you as a tester. I could use the money for pâté and caviar instead of the cheap kibble he feeds me.

It's amazing to realize that a simple beast like me has witnessed the entire rise of Windows over the last 20 years. I wish I could stay around to see the next 20. Thanks for listening to me. Bye. ■

SIMBA adopted David Platt in 1991. She has obstructed the writing of 11 programming books by jumping on the keyboard. Watch for her forthcoming tell-all book, "Why Being Dave's Cat Sucks," and her subsequent appearance on Oprah. A New England snowshoe cat with polydactyl thumbs, she can almost count in octal.

Can Your **GRID** Do This?



SPREAD⁵... DOES IT ALL!

Start building smarter applications
GCPowerTools.com/Videos

WE ARE
SPREADSHEETS



ActiveReports

Spread

DataDynamicsReports

ActiveAnalysis

GrapeCity PowerTools
Report & Analyze & Excel



The Dashboard Framework for Developers like you



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



www.dundas.com
(416) 467-5100 • (800) 463-1492

Silverlight is a trademark of Microsoft Corporation in the United States and/or other countries.