THE MICROSOFT JOURNAL FOR DEVELOPERS

MARCH 2011 VOL 26 NO 3

msdn magazine

PROCESS AND DATA INTEGRATION

COLUMNS

Microsoft

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.

msdn® magazine

MARCH 2011 VOLUME 26 NUMBER 3

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

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

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

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

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

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

Microsoft

BPA WORLDWIDE

Printed in the USA

Your best source for software development tools!

programmer's paradise

LEADTOOLS Document Imaging SDK v17.0

by LEAD Technologies

LEADTOOLS Document Imaging has every component you need to develop powerful image-enabled business applications including specialized bi-tonal image processing, document clean up, annotations, high-speed scanning, advanced compression (CCITT G3/G4, JBIG2, MRC, ABC), and Win32/64 binaries for C/C++, .NET, Silverlight, WPF, WCF, & WF. Available add-ons include:

- Multi-threaded OCR/ICR/OMR/MICR/Barcodes (1D/2D)
- Forms Recognition/Processing
- Print Capture and Document Writers
- PDF, PDF/A and XPS

programmers.com/LEAD

Paradise # L05 03301A01

$2,007.99

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

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

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!

programmers.com/embarcadero

NEW RELEASE!

Paradise # CGI 15401A01

$1,383.99

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

VMware Workstation 7

VMware Workstation 7 is the gold-standard virtualization software for desktop and laptop computers, with the richest feature set and broadest platform support available. VMware Workstation enables users to create and host multiple virtual machines on a single desktop, expanding the power of desktop systems for IT administrators; software development and test engineers; technical sales, training and support staff; and PC enthusiasts.

VMware Workstation transforms the way technical professionals develop, test, demo, and deploy software. Workstation's innovative features for VMware environments help to reduce hardware cost, save time, minimize risk, and streamline tasks that save time and improve productivity.

for Linux & Windows Paradise # V55 22301A04

$153.99

programmers.com/vmware

Spread for Windows Forms 5

by GrapeCity

- World's best selling .NET Spreadsheet
- Import/export native Microsoft Excel files with full formatting
- Extremely flexible printing and export options including PDF
- Extensible formula support, including Microsoft Excel functions
- Hundreds of chart styles for enhanced data visualization
- Powerful user interface and flexible data connectivity
- WYSIWYG spreadsheet designers, quick-start wizard and chart designers
- Royalty-free licensing

programmers.com/grapecity

Upgrade Paradise # F02 01101A01

$936.99

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

TX Text Control 16.0

Word Processing Components

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

New Version Released!

- .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

Professional Edition Paradise # T79 12101A01

$1,109.99

Download a demo today.

programmers.com/textcontrol

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.

programmers.com/intel

NEW RELEASE!

for Windows Single (SSR) Paradise # I23 86101E03

$263.99

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

$41.99

programmers.com/microsoft

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 TRWD03** When You Place Your Order Online or with Your Programmer's Paradise Representative.

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

Prices subject to change. Not responsible for typographical errors.

EDITOR'S NOTE

KEITH WARD

How to Get Rejected

Last month, I talked about some of the best ways to get published in *MSDN Magazine*. That naturally led me to think of some equally good ways to *not* get published in the magazine. If February was the list of dos, this is the list of don'ts. Follow these rules (or is that "don't follow these rules"—I get confused with negatives) to give your query the best shot at not getting a positive response from me and the team.

The best way to ensure your query doesn't get accepted, or even considered, is to put it in the wrong format. We've written a guide to the proper formatting of queries, which you can find at bit.ly/eZcovQ. One of the critical pieces of information in the document is the title of your query. It should state "MSDN Article Query" in the subject line—and nothing else. Please don't get cute with the title; clever subject lines don't thrill me. Like you, I get tons of e-mail each day, and I need to quickly separate the wheat from the chaff. If you've got the right title, I'll look over the query; if not, it's ye olde Delete key for your e-mail.

Another way to hurt your chances: Make your query novella length. My eyes glaze over when I open an article query and notice that it will take quite a lot of scrolling to get through it all. Brevity is key here—short, sweet and clear. If you can't do this in your query, I'll have serious doubts about whether you can do it in an article.

That brings up another crucial point: Your query, if I haven't worked with you before, is your initial writing audition. Writing for *MSDN Magazine* isn't like writing a blog entry—you don't need to have been previously published, but you *do* need to demonstrate at least basic writing chops. If your query is full of misspellings, sloppy grammar and missing information, you'll get turned down, guaranteed. After all, if I can't trust you to write a proper query, I'm surely not going to trust you to write an entire article.

One last item about queries: Please don't make your pitch in an attached Word document. I'm not keen on downloading and opening attachments just to read a query. Include the query in the body of the e-mail—please!

Now, a few words about how to get your article rejected after it's been accepted for publication. The first, best (worst?) thing you can do is not communicate with the staff, beginning with me. It's happened numerous times that an author is late with an article. Hey, stuff happens, and delays can be caused by any number of circumstances. What drives me to distraction is when an author doesn't inform me that an article or requested bit of information will be late. I can almost always work around delays; to do that, however, I need to hear from you. Even if you come to the conclusion that you won't be able to turn in an article at all (it happens sometimes), let me know so I can make alternative arrangements. If I don't know, however, it throws a shiny steel monkey wrench right into the middle of our processes, and makes me (and my staff) miserable. Please don't make us miserable.

Other reasons your submitted article can be rejected:

- Plagiarism. You can't copy and paste from published documents. At all. Not nowhere, not nohow. "That's obvious!" you say. You'd be surprised at how un-obvious it is to some writers.
- Sloppy writing. See my section about the query. You may not be the Shakespeare of dev authors, but strive to turn in your best work. That means after you write the article, get away from it for a day and go back and edit it. Then do it again. If information is unclear or incomplete, fix it. Don't expect us to do it all for you. If we do that, and still publish your article anyway because we love the topic, be assured you won't get assigned a second article.
- Failure to follow writer's guidelines. All authors get a copy of our writing guidelines. I suspect some of them barely glance at the document. Don't let this be you. There are specific rules that need to be followed. Learn them. Love them.

One of the best parts of my job is working with authors. I try to make it a relaxed, enjoyable process. You can do your part to help me by avoiding these don'ts. Send your (properly formatted!) article ideas to me at mmeditor@microsoft.com.

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

Cloud-based Scrum Tool for Developers

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

$10 / month for 10 users • Free 30-day Team Trial

Local Scrum Tool for Developers

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

Starts @ $395 for teams of 10 • Free for 1 user

Subversion for Windows and VS

RocketSVN Server represents a state-of-the-art, fast & easy to install source code management (SCM) server for Windows on IIS.

RocketSVN for Visual Studio is an SVN client that integrates with the Visual Studio IDE

100% Free for Unlimited Users

www.axosoft.com

(800) 653-0024 • (480) 362-1900

TERRENCE DORSEY

Data Integration Tools and Resources

It seems that all but the most trivial applications these days deal with data. Often, lots of data. So I guess that means, as a developer, you need to add "Database Expert" to the many hats you wear. Or maybe not: What if there were tools and resources out there that gave you a leg up and did some of the heavy lifting for you?

A good place to start is the **MSDN Data Developer Center** (msdn.microsoft.com/data), where you'll find links to a huge selection of Microsoft tools and technologies for integrating data access into your apps. From SQL Server to ADO.NET to MSXML, you'll find it all there.

In fact, it's such a comprehensive resource I probably could leave it at that. Happy databinding!

But wait! There's more ...

Data Basics

What's that? You're not already savvy in the ways of SQL Server? Tables, rows and records sounds more like party planning than dev speak? If you're just getting started with data-centric development—or need a refresher—here are a few resources that will get you up to speed.

Geekgirl's Plain-English Computing database guides (geekgirls.com/category/office/databases) provide a series of "Databasics" tutorials explaining the core concepts of database design and use. If you're starting from scratch, you can't do much better than this.

Once you understand those basics, a solid next step would be something like the **Code Project** article "**SQL for Developers: Basic Data Retrieval**" (bit.ly/gurX8Y). Here you'll learn how to use simple SQL queries to get selected data out of a database and into your app, where you can do something useful with it. (That task is left to the reader ... but read on for some tips.)

For a deeper look at the syntax for communicating with a database engine like SQL Server 2008, check out the **Transact-SQL Reference** (bit.ly/hDhdvz) in SQL Server Books Online.

Once you get coding, you'll want to check frequently for tips and tricks at **SQL for Programmers**, the .NET Answers archive of SQL programming tips (bit.ly/ejD7Zg).

LINQ

Language Integrated Query (LINQ) is a feature of the Microsoft .NET Framework that extends data access using native language constructs in C# and Visual Basic (and F# to some extent, as well). Learn more about it at the **LINQ Developer Center** (bit.ly/fl9xpg).

One of the strengths of LINQ is that it enables you to write SQL-like queries using strongly typed syntax. Then, LINQ providers such as LINQ to SQL or LINQ to Objects can handle the fine details of the actual data source. For a practical overview of how this works, see the Data Points column by John Papa, "**Standard Query Operators with LINQ**" (bit.ly/huKhxa).

LINQPad (linqpad.net) has become a crucial tool for developers to learn LINQ, prototype queries, or interactively query a wide variety of data sources. LINQPad is a free tool written by Joseph Albahari (albahari.com) and supported by a huge community of programmers and data experts.

If you're just getting started with LINQ, check out Dan Wahlin's blog post, "**Learn How to Use LINQ with LINQPad**" (bit.ly/hlOyMh), and Al Tenhundfeld's article, "**Master LINQ with LINQPad**" (bit.ly/fSUij4). Both will get you up and running quickly.

Not only is LINQPad a great tool for LINQ queries, it also lets you interactively run and verify functions in C# and Visual Basic. Rich Strahl demonstrates this handy feature in his article, "**LINQPad as a Code Snippet Execution Engine**" (bit.ly/eCD60C).

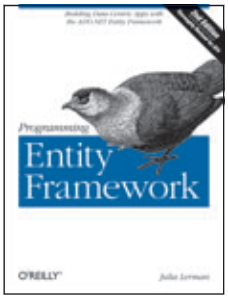
LINQPad has become a crucial tool for developers to learn LINQ, prototype queries, or interactively query a wide variety of data sources.

Entity Framework

The ADO.NET Entity Framework is a .NET object-relational mapping (O/RM) framework meant to simplify access to relational databases from your code. Simply stated, the Entity Framwork lets you map your database schema to programmatic entities that you can query via properties. To learn more about the Entity Framework, see the **ADO.NET Entity Framework Developer Center** (bit.ly/e0mtC1).

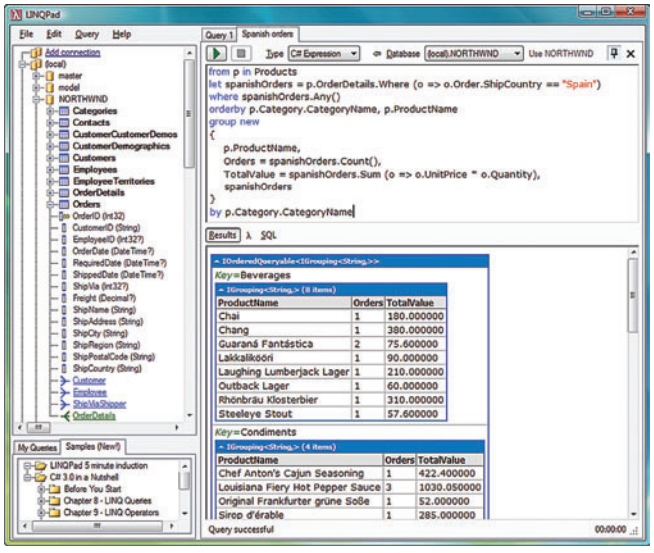
If you're new to the Entity Framework, get up to speed by working through the Microsoft "**Getting Started with Entity Framework**" tutorials (bit.ly/gcrXyU) for WebForms. If you prefer to use ASP.NET MVC, there's also a "**Creating Model Classes with the Entity Framework**" tutorial (bit.ly/dXJAjx).

Julie Lerman, our regular Data Points columnist, is an Entity Framework expert, having written the comprehensive "**Programming Entity Framework, Second Edition**" (O'Reilly, 2010). Learn more about her book at learnentityframework.com. Want to get a taste of some advanced tasks you can achieve with the Entity Framework? Read Lerman's December 2010 column, "**Profiling Database Activity in the Entity Framework**" (bit.ly/flLwdw).



Julie Lerman's Book

Of course, LINQPad is a great tool for learning the Entity Framework, too. Check out "**Using LINQPad with Entity Framework**," (bit.ly/hUBRu0) for a full tutorial.



LINQPad

WCF Data Services and OData

WCF Data Services—formerly known as ADO.NET Data Services—lets you share, consume and update data via HTTP using the **OData** protocol (odata.org). Like the Entity Framework, WCF Data Services uses an Entity Data Model (EDM) to bridge between data source and program entities. You can read more about **WCF Data Services on MSDN** at bit.ly/hnuvwv.

To get you started, Shayne Burgess walks you through the basics of using OData and Data Services in the article, "**Building Rich Internet Apps with the Open Data Protocol**," in the June 2010 issue of *MSDN Magazine* (bit.ly/gPZGDc).

Not sure which WCF services to use for your data-centric app? Tony Sneed wrote an in-depth appraisal of **WCF Data Services vs. WCF Soap Services** (bit.ly/icbLnR) that will help you understand the strengths of each approach.

How about putting a bunch of these technologies together into one interesting sample? Shawn Wildermuth, in his article "**WCF Data Services and jQuery**" (bit.ly/hVCMWd), builds a Web app that uses jQuery to retrieve data in JSON format, exposes the data as entities via Entity Framework, and then uses WCF Data Services to expose the entities via REST. I think we have a "Buzzword Bingo" winner here.

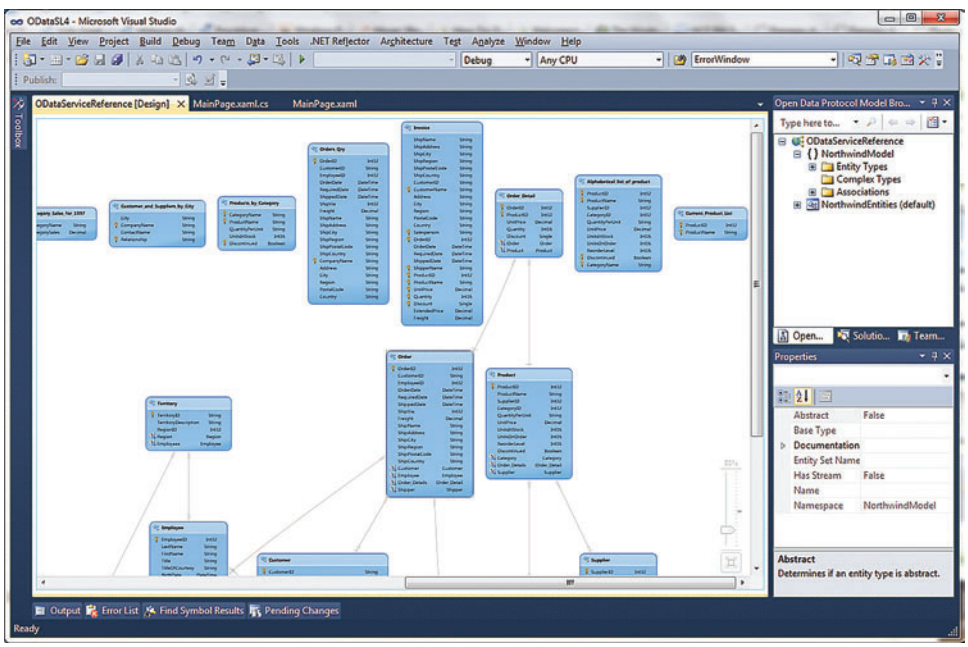
A highly rated tool for making sense of your data sources is the **Open Data Protocol Visualizer** extension for Visual Studio 2010 (bit.ly/dWt19X), which displays the types and relationships provided by WCF Data Services in simplified diagram form. Read Audrey Petit's "**Open Data Protocol Visualizer Extension for Visual Studio 2010**" blog post (bit.ly/hKSKRx) to learn how it works.

NHibernate

NHibernate (nhforge.org) is an open source O/RM framework for development with the .NET Framework. Like the Entity Framework, NHibernate lets you map databases to entities and allows simplified programmatic access to the data.

Because it's a community-driven effort, there are lots of useful resources available for learning and using NHibernate in your projects. One great way is to jump in and start coding. Gabriel Schenker's "**Your Very First NHibernate Application**" series of articles on dotnetslackers.com (direct link: bit.ly/exFATb) is one such tutorial. Another is Mitch Fincher's "**Learning with Code Samples**" for NHibernate (bit.ly/e91Nzv).

Would you rather see the movie? Well then, check out the **Summer of NHibernate** (summerofnhibernate.com) screencast series, which walks you step-by-step from getting set up to writing your first queries to advanced topics like modeling inheritance and managing session state. There's a lot to watch, so get your popcorn ready and settle in for a few evenings of learning.



Open Data Protocol Visualizer

Zentity

Many business and social apps run on data, but researchers are increasingly creating and sorting through huge data sources from clinical studies, experimental results and even celestial observations. In response, Microsoft Research released **Zentity 2.0** (bit.ly/fiFPb3), a data library framework that can be used for storing, accessing and analyzing data using SQL Server 2008. The new version of Zentity leverages the .NET Framework 4 with support for WCF Data Services and OData, LINQ, Entity Framework, Windows PowerShell and more. ■

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

DINO ESPOSITO

Application Extensibility: MEF vs. IoC

There's an interesting new component in the Microsoft .NET Framework 4 specifically designed to provide an effective answer to an evergreen question: How would you write extensible applications that can discover at run time all the parts they're made of?

As Glenn Block explained in his February 2010 article, "Building Composable Apps in .NET 4 with the Managed Extensibility Framework" (msdn.microsoft.com/magazine/ee291628), the Managed Extensibility Framework (MEF) can be used to streamline building composable and plug-in-based applications. As one who started approaching the problem back in 1994 (yes, it was one of my first real challenges as a developer), I definitely welcome any proposed solutions in this problem space.

The MEF doesn't require you to buy, download and reference any additional libraries, and it offers a simple programming interface because it's focused on solving the problem of general, third-party extensibility of existing applications. Glenn's article is an excellent introduction to the MEF and should be considered required reading if you're thinking about plug-in-based applications.

In this article, I'll walk through the steps required to build an extensible application using the MEF as the underlying glue to keep the main body and external parts of the application together.

From IoC to the MEF and Back

Before I get to the sample application, however, I'd like to share some thoughts about the MEF and another popular family of frameworks: Inversion of Control (IoC).

In a nutshell, it's correct to say that the functionality of the MEF and of a typical IoC framework overlap, but don't coincide. With most IoC frameworks you can perform tasks that the MEF just doesn't support. You could probably employ a functionally rich IoC container and, with some effort on your own, emulate some MEF-specific capabilities. In light of this, the question that I'm frequently asked when I mention the MEF in classes and everyday work is: What's the difference between the MEF and an IoC tool? And when do I really need the MEF?

My thought is that, at its core, the MEF is an IoC framework built right into the .NET Framework. It's not as powerful as many of the popular IoC frameworks today, but it can perform the basic tasks of a typical IoC container quite well.

Today, IoC frameworks have three typical capabilities. First, they can act as the factory of a graph of objects and walk through the chain of object relationships and dependencies to create an instance of any required and registered type. Second, an IoC framework can manage the lifetime of created instances and offer caching and pooling capabilities. Third, most IoC frameworks support interception and offer to create dynamic proxies around instances of specific types so that developers can pre- and post-process the execution of methods. I covered interception in Unity 2.0 in January (msdn.microsoft.com/magazine/gg535676).

The MEF, in a way, can serve as the factory of a graph of objects, meaning that it can recognize and handle members on a class that need be resolved at run time. The MEF also provides minimal support for caching instances, meaning that some caching capabilities exist, but they're not as functionally rich as in some other IoC frameworks. Finally, in the version shipped with the .NET Framework 4, the MEF lacks interception capabilities entirely.

Having said that, when should you use the MEF? If you've never used an IoC framework and just need to clean up the design of your system by adding a bit of dependency injection, then the MEF can be an easy start. As long as you can quickly achieve your goals with it, the MEF is preferable to an IoC framework.

On the other hand, if you've spent years working with one or more IoC frameworks and can squeeze any bit of functionality out of them, then there's probably nothing that the MEF can give you except, perhaps, its ability to scan various types of catalogs to find matching types. It should be noted, however, that some IoC frameworks such as StructureMap (structuremap.net/structuremap/ScanningAssemblies.htm) already offer to scan directories and assemblies to look for specific types or implementations of given interfaces. With the MEF, this is probably easier and more direct to do than with StructureMap (and a few others).

In summary, the first question to answer is whether you're looking for general extensibility. If the answer is yes, then the MEF must be considered—perhaps in addition to an IoC tool if you also need to handle dependencies, singletons and interception. If the

Figure 1 **Definitions for the Application SDK**

```
public interface IFindTheNumberPlugin {
  void ShowUserInterface(GuessTheNumberSite site);
  void NumberEntered(Int32 number);
  void GameStarted();
  void GameStopped();
}

public interface IFindTheNumberApi {
  Int32 MostRecentNumber { get; }
  Int32 NumberOfAttempts { get; }
  Boolean IsUserPlaying { get; }
  Int32 CurrentLowerBound { get; }
  Int32 CurrentUpperBound { get; }
  Int32 LowerBound { get; }
  Int32 UpperBound { get; }
  void SetNumber(Int32 number);
}

public class FindTheNumberFormBase : Form, IFindTheNumberApi {
  ...
}
```

answer is no, then the best approach is using an IoC framework unless you have basic needs that the MEF can address as well. All things being equal, the MEF is preferable to an IoC framework because it's built right into the .NET Framework and you don't need to take any additional dependencies.

The MEF and Extensible Applications

While the MEF helps in the building of an extensible application, the most delicate part of the job is designing the application for extensibility. This is design and has little to do with the MEF, IoC or other technologies. In particular, you must figure out which parts of your application you intend to make available to plug-ins.

A plug-in is often a visual element and needs to interact with the UI of the main application, add or extend menus, create panes, display dialog boxes, or even add or resize the main windows. Depending on how you envision the plug-ins of your specific application, the amount of information to share with plug-ins may consist of just business data (essentially a segment of the application's current state) or reference to visual elements such as containers, menus, toolbars and even specific controls. You group this information in a data structure and pass it down to the plug-in at initialization time. Based on that information, the plug-in should be able to adjust its own UI and implement its own additional custom logic.

Next comes the interface for the plug-ins. The interface depends on the injection points you've identified in your main application. By "injection point" I mean the places in the application's code from which you'd invoke plug-ins to give them a chance to kick in and operate.

As an example of an injection point, consider Windows Explorer. As you may know, Windows Explorer allows you to extend its UI via shell extensions. These plug-ins are invoked at very specific moments—for example, when the user right-clicks to display the properties of a selected file. As the application's architect, it's your responsibility to identify these injection points and what data you intend to pass to registered plug-ins at that point.

Once every design aspect has been cleared up, you can look around for frameworks that can simplify the task of building a plug-in-based application.

A Sample Plug-In-Based Application

Even a simple application such as "Find the number" can be made richer and functionally appealing using plug-ins.

You might want to create a separate project to define the SDK of your application. It will be a class library where you define all classes and interfaces required to implement plug-ins. **Figure 1** shows an example.

All plug-ins are required to implement the IFindTheNumberPlugin interface. The main application form will inherit from the specified form class, which defines a list of public helper members useful to pass information down to plug-ins.

As you may guess from IFindTheNumberPlugin, registered plug-ins are invoked when the application displays its UI, when the user makes a new attempt to guess the number, and when the game is started and stopped. GameStarted and GameStopped are just notification methods and don't need any input. NumberEntered is a notification that brings in the number the user just typed and submitted for a new try. Finally, ShowUserInterface is invoked when the plug-in must show up in the window. In this case, a site object is passed, as defined in **Figure 2**.

Figure 2 **The Site Object for Plug-Ins**

```
public class FindTheNumberSite {
  private readonly FindTheNumberFormBase _mainForm;

  public FindTheNumberSite(FindTheNumberFormBase form) {
    _mainForm = form;
  }

  public T FindElement<T>(String name) where T:class { ... }
  public void AddElement(Control element) { ... }

  public Int32 Height {
    get { return _mainForm.Height; }
    set { _mainForm.Height = value; }
  }

  public Int32 Width { ... }
  public Int32 NumberOfAttempts { ... }
  public Boolean IsUserPlaying { ... }
  public Int32 LowerBound { ... }
  public Int32 UpperBound { ... }
  public void SetNumber(Int32 number) { ... }
}
```

The site object represents the point of contact between the plug-in and the host application. The plug-in must gain some visibility of the host state and must even be able to modify the host UI, but it never gains knowledge of the host's internal details. That's why you might want to create an intermediate site object (part of your SDK assembly) that plug-in projects must reference.

I omitted for brevity the implementation of most methods in **Figure 2**, but the constructor of the site object receives a reference to the application's main window, and using helper methods in **Figure 1** (exposed by the main window object) it can read and write the application's state and visual elements. For example, the Height member shows how the plug-in may read and write the height of the host window.

In particular, the FindElement method allows the plug-in (in the sample application) to retrieve a particular visual element in the form. It's assumed that you unveil as part of your SDK some technical details of how to access certain containers such as toolbars, menus and the like. In such a simple application, it's assumed that you document the ID of the physical controls. Here's the implementation of FindElement:

Figure 3 **Initializing the MEF**

```
private void InitializeMef() {
  try {
    _pluginCatalog = new DirectoryCatalog(@"\My App\Plugins");
    var filteredCatalog = new FilteredCatalog(_pluginCatalog,
      cpd => cpd.Metadata.ContainsKey("Level") &&
      !cpd.Metadata["Level"].Equals("Basic"));

    // Create the CompositionContainer with the parts in the catalog
    _container = new CompositionContainer(filteredCatalog);
    _container.ComposeParts(this);
  }
  catch (CompositionException compositionException) {
    ...
  }
  catch (DirectoryNotFoundException directoryException) {
    ...
  }
}
```

THE COMMENTS

"Outstanding!"

"Second-to-none!"

"No one comes close!"

"One of my best decisions!"

"Superb!"

"You guys really rock!"

"Your products are excellent!"

"Amazing components!"

```
public T FindElement<T>(String name) where T:class {
  var controls = _mainForm.Controls.Find(name, true);
  if (controls.Length == 0)
    return null;
  var elementRef = controls[0] as T;
  return elementRef ?? null;
}
```

With the design of the application's extensibility model completed, we're now ready to introduce the MEF.

Defining Imports for Plug-ins

The main application will certainly expose a property that lists all currently registered plug-ins. Here's an example:

```
public partial class FindTheNumberForm :
  FindTheNumberFormBase {
  public FindTheNumberForm() {
    InitializeMef();
    ...
 }

 [ImportMany(typeof(IFindTheNumberPlugin)]
 public List<IFindTheNumberPlugin> Plugins {
    get; set;
  }
  ...
}
```

Initializing the MEF means preparing the composition container specifying the catalogs you intend to use and optional export providers. A common solution for plug-in-based applications is loading plug-ins from a fixed folder. **Figure 3** shows the startup code of the MEF in my example.

You use a DirectoryCatalog to group available plug-ins and use the FilteredCatalog class (which is not in the MEF, but an example is shown in the MEF documentation at bit.ly/gf9xDK) to filter out some of the selected plug-ins. In particular, you can request that all loadable plug-ins have a metadata attribute that indicates the level. Missing the attribute, the plug-in is ignored.

The call to ComposeParts has the effect of populating the Plugins property of the application. The next step is just invoking plug-ins from the various injection points. The first time you invoke plug-ins is right after the application loads to give them a chance to modify the UI:

```
void FindTheNumberForm_Load(Object sender, EventArgs e) {
  // Set up UI
  UserIsPlaying(false);

  // Stage to invoke plugins
  NotifyPluginsShowInterface();
}

void NotifyPluginsShowInterface() {
  var site = new FindTheNumberSite(this);
  if (Plugins == null)
    return;

  foreach (var p in Plugins) {
    p.ShowUserInterface(site);
  }
}
```

Similar calls will appear in the event handlers that signal when the user just started a new game, quit the current game or just made a new attempt to guess the mysterious number.

Writing a Sample Plug-In

A plug-in is just a class that implements your app's extensibility interface. An interesting plug-in for the application is one that shows the number of attempts the user made so far. The number of attempts is being tracked by the business logic of the application, and it's exposed to plug-ins via the site object. All a plug-in must do is prepare its own UI, bind it to the number of attempts and attach it to the main window.

Figure 4 **The Counter Plug-In**

```
[Export(typeof(IFindTheNumberPlugin))]
[PartMetadata("Level", "Advanced")]
public class AttemptCounterPlugin : IFindTheNumberPlugin {
  private FindTheNumberSite _site;
  private Label _attemptCounterLabel;

  public void ShowUserInterface(FindTheNumberSite site) {
    _site = site;
    var numberToGuessLabelRef = _host.FindElement<Label>("NumberToGuess");
    if (numberToGuessLabelRef == null)
      return;

    // Position of the counter label in the form
    _attemptCounterLabel = new Label {
      Name = "plugins_AttemptCounter",
      Left = numberToGuessLabelRef.Left,
      Top = numberToGuessLabelRef.Top + 50,
      Font = numberToGuessLabelRef.Font,
      Size = new Size(150, 30),
      BackColor = Color.Yellow,
      Text =  String.Format("{0} attempt(s)", _host.NumberOfAttempts)
    };
    _site.AddElement(_attemptCounterLabel);
  }

  public void NumberEntered(Int32 number = -1) {
    var attempts = _host.NumberOfAttempts;
    _attemptCounterLabel.Text = String.Format("{0} attempt(s)", attempts);
    return;
  }

  public void GameStarted() {
    NumberEntered();
  }

  public void GameStopped() {
  }
}
```

Plug-ins of the sample application will create new controls in the UI of the main window. **Figure 4** shows a sample plug-in.

The plug-in creates a new Label control and places it just below an existing UI element. Next, whenever the plug-in receives the notification that a new number has been entered, the counter is updated to show the current number of attempts according to the state of the business logic.

Plugging In

At the end of the day, the most delicate task of designing extensible apps is the design of the host and the interface of plug-ins. This is a pure design task and has to do with the feature list and user's requirements.

When it comes to implementation, though, you have quite a few practical tasks to accomplish regardless of the plug-in interface, such as selecting, loading and verifying plug-ins. In this regard, the MEF gives you significant help in simplifying creation of the catalog of plug-ins to load, and automatically loading them up in much the same way an IoC framework would do.

Note that the MEF is under continual development, and you can find the latest bits, documentation and example code at mef.codeplex.com. ■

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

THANKS to the following technical expert for reviewing this article: *Glenn Block*

JULIE LERMAN

Server-Side Paging with the Entity Framework and ASP.NET MVC 3

In my January Data Points column, I showed off the jQuery DataTables plug-in and its ability to seamlessly handle huge amounts of data on the client side. This works well with Web applications where you want to slice and dice large amounts of data. This month, I'll focus on using queries that return smaller payloads to enable a different type of interaction with the data. This is especially important when you're targeting mobile applications.

I'll take advantage of features introduced in ASP.NET MVC 3 and demonstrate how to use these together with efficient server-side paging against the Entity Framework. There are two challenges with this task. The first is to provide an Entity Framework query with the correct paging parameters. The second is to mimic a feature of client-side paging by providing visual clues to indicate that there's more data to retrieve, as well as links to trigger the retrieval.

ASP.NET MVC 3 has a slew of new features, such as the new Razor view engine, validation improvements and a ton more JavaScript features. The launch page for MVC is at asp.net/mvc, where you can download ASP.NET MVC 3 and find links to blog posts and training videos to help you get up to speed. One of the new features that I'll use is the ViewBag. If you've used ASP.NET MVC previously, ViewBag is an enhancement to the ViewData class and lets you use dynamically created properties.

Another new element that ASP.NET MVC 3 brings to the table is the specialized System.Web.Helpers.WebGrid. Although one of the grid's features is paging, I'll use the new grid but not its paging in this example, because that paging is client-side—in other words, it pages through a set of data provided to it, similar to the DataTables plug-in. I'm going to be using server-side paging instead.

For this little app, you'll need an Entity Data Model to work with. I'm using one created from the Microsoft AdventureWorksLT sample database, but I only need the Customer and SalesOrderHeaders brought into the model. I've moved the Customer rowguid, PasswordHash and PasswordSalt properties into a separate entity so that I don't have to worry about them when editing. Other than this small change, I haven't modified the model from its default.

I created a project using the default ASP.NET MVC 3 project template. This prepopulates a number of controllers and views, and I'll let the default HomeController present the Customers.

I'll use a simple DataAccess class to provide interaction with the model, context and, subsequently, the database. In this class, my GetPagedCustomers method provides server-side paging. If the goal of the ASP.NET MVC application was to allow the user to interact with all of the customers, that would be a lot of customers returned in a single query and managed in the browser. Instead, we'll let the app present 10 rows at a time and the GetPagedCustomers will provide that filter. The query that I'll eventually need to execute looks like this:

```
context.Customers.Where(c =>
c.SalesOrderHeaders.Any()).Skip(skip).Take(take).ToList()
```

The view will know which page to request and give that information to the controller. The controller will be in charge of knowing how many rows to supply per page. The controller will calculate the "skip" value using the page number and the rows per page. When the controller calls the GetPagedCustomers method, it will pass in the calculated skip value as well as the rows per page, which is the "take" value. So if we're on page four and presenting 10 rows per page, skip will be 40 and take will be 10.

The paging query first creates a filter that requests only those customers who have any SalesOrders. Then, using LINQ Skip and Take methods, the resulting data will be a subset of those customers. The full query, including the paging, is executed in the database. The database returns only the number of rows specified by the Take method.

The query is composed of a few parts to enable some tricks I'll add down the road. Here's a first pass at the GetPagedCustomers method that will be called from the HomeController:

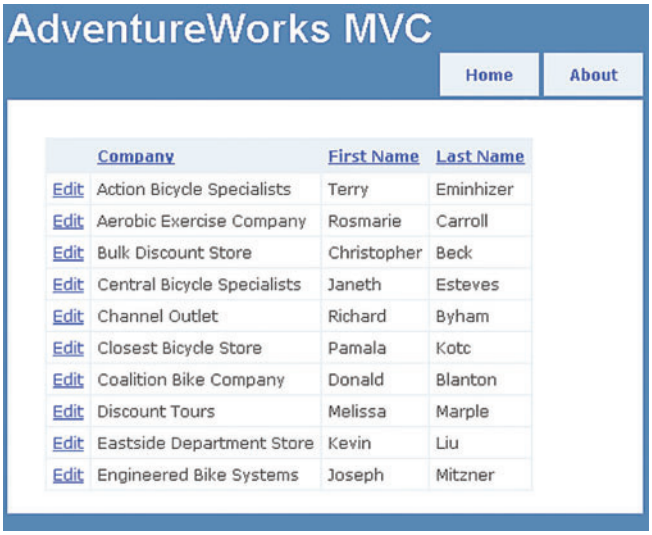


Figure 1 **Providing Edit ActionLinks in the WebGrid**

Code download available at code.msdn.microsoft.com/mag201103DataPoints.

Figure 2 **The New Version of GetPagedCustomers**

```
public static PagedList<Customer> GetPagedCustomers(int skip, int take)
{
  using (var context = new AdventureWorksLTEntities())
  {
    var query = context.Customers.Include("SalesOrderHeaders")
      .Where(c => c.SalesOrderHeaders.Any())
      .OrderBy(c => c.CompanyName + c.LastName + c.FirstName);

    var customerCount = query.Count();

    var customers = query.Skip(skip).Take(take).ToList();

    return new PagedList<Customer>
    {
      Entities = customers,
      HasNext = (skip + 10 < customerCount),
      HasPrevious = (skip > 0)
    };
  }
}
```

```
public static List<Customer> GetPagedCustomers(int skip, int take)
{
  using (var context = new AdventureWorksLTEntities())
  {
    var query = context.Customers.Include("SalesOrderHeaders")
      .Where(c => c.SalesOrderHeaders.Any())
      .OrderBy(c => c.CompanyName + c.LastName + c.FirstName);

    return query.Skip(skip).Take(take).ToList();
  }
}
```

The controller Index method that calls this method will determine the number of rows to return using a variable I'll call pageSize, which becomes the value for Take. The Index method will also specify where to begin based on a page number that will be passed in as a parameter, as shown here:

```
public ActionResult Index(int? page)
{
  const int pageSize = 10;
  var customers=DataAccess.GetPagedCustomers((page ?? 0)*pageSize, pageSize);
  return View(customers);
}
```

This gets us a good part of the way. The server-side paging is completely in place. With a WebGrid in the Index view markup, we can display the customers returned from the GetPagedCustomers method. In the markup, you need to declare and instantiate the grid, passing in Model, which represents the List<Customer> that was provided when the controller created the view. Then, using the WebGrid GetHtml method, you can format the grid, specifying which columns to display. I'll only show three of the Customer properties: CompanyName, FirstName and LastName. You'll be happy to find full IntelliSense support as you type this markup whether you use syntax associated with ASPX views or with the new MVC 3 Razor view engine syntax (as with the following example). In the first column, I'll provide an Edit ActionLink so that the user can edit any of the Customers that are displayed:

```
@{
  var grid = new WebGrid(Model);
}
<div id="customergrid">
  @grid.GetHtml(columns: grid.Columns(
    grid.Column(format: (item) => Html.ActionLink
      ("Edit", "Edit", new { customerId = item.CustomerID })),
  grid.Column("CompanyName", "Company"),
  grid.Column("FirstName", "First Name"),
  grid.Column("LastName", "Last Name")
   ))
</div>
```

The result is shown in **Figure 1**.

So far, so good. But this doesn't provide a way for the user to navigate to another page of data. There are a number of ways to achieve this. One way is to specify the page number in the URI—for example, http://adventureworksmvc.com/Page/3. Surely you don't want to ask your end users to do this. A more discoverable mechanism is to have paging controls, such as page number links "1 2 3 4 5 …" or links that indicate forward and backward, for example, "<< >>."

The current roadblock to enabling the paging links is that the Index view page has no knowledge that there are more Customers to be acquired. It knows only that the universe of customers is the 10 that it's displaying. By adding some additional logic into the data-access layer and passing it down to the view by way of the controller, you can solve this problem. Let's begin with the data-access logic.

In order to know if there are more records beyond the current set of customers, you'll need to have a count of all of the possible customers that the query would return without paging in groups of 10. This is where composing the query in the GetPagedCustomers will pay off. Notice that the first query is returned into _customerQuery, a variable that's declared at the class level, as shown here:

```
_customerQuery = context.Customers.Where(c => c.SalesOrderHeaders.Any());
```

You can append the Count method to the end of that query to get the count of all of the Customers that match the query before paging is applied. The Count method will force a relatively simple query to be executed immediately. Here's the query executed in SQL Server, from which the response returns a single value:

```
SELECT
[GroupBy1].[A1] AS [C1]
FROM ( SELECT
       COUNT(1) AS [A1]
       FROM [SalesLT].[Customer] AS [Extent1]
       WHERE  EXISTS (SELECT
              1 AS [C1]
              FROM [SalesLT].[SalesOrderHeader] AS [Extent2]
              WHERE [Extent1].[CustomerID] = [Extent2].[CustomerID]
       )
)  AS [GroupBy1]
```

Once you have the count, you can determine if the current page of customers is the first page, the last page or something in between. Then you can use that logic to decide which links to display. For example, if you're beyond the first page of customers, then it's logical to display a link to access earlier pages of customer data with a link for the previous page, for example, "<<."

We can calculate values to represent this logic in the data-access class and then expose it in a wrapper class along with the customers. Here's the new class I'll be using:

```
public class PagedList<T>
{
  public bool HasNext { get; set; }
  public bool HasPrevious { get; set; }
  public List<T> Entities { get; set; }
}
```

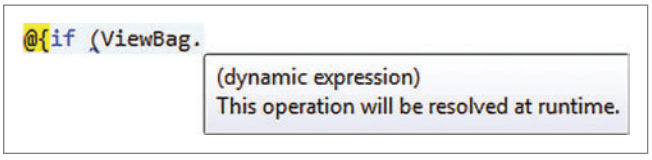


Figure 3 **ViewBag Properties Aren't Available Through IntelliSense Because They're Dynamic**

LEADTOOLS® version 17

The World Leader in Imaging SDKs

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

- **Silverlight:** 100% pure Silverlight 3, 4 and Windows Phone Imaging SDK.
- **Image Formats & Compression:** Supports 150+ image formats and compressions including TIFF, EXIF, PDF, JPEG2000, JBIG2 and CCITT G3/G4.
- **Scanning:** TWAIN & WIA (32 & 64-bit), auto-detect optimum driver settings for high speed scanning.
- **Viewer Controls:** Win Forms, Web Forms, WPF, Silverlight, ActiveX and COM.
- **Image Processing:** 200+ filters, transforms, color conversion and drawing functions supporting region of interest and extended grayscale data.
- **Document Cleanup/Preprocessing:** Auto-deskew, despeckle, hole punch, line and border removal, inverted text correction and more for optimum results in OCR and Barcode recognition.
- **Barcode:** Auto-detect, read and write 1D and 2D barcodes for multithreaded 32 & 64 bit development.
- **OCR/ICR/OMR:** Full page or zonal recognition for multithreaded 32 and 64 bit development with PDF, PDF/A, XPS, DOC, XML and Text output.
- **Forms Recognition & Processing:** Automatically identify and classify forms and extract user filled data.
- **PDF & PDF/A:** Read, write and view searchable PDF with text, images, bookmarks and annotations.
- **Annotations:** Interactive UI for document mark-up, redaction and image measurement (including support for DICOM annotations).
- **DICOM:** Full support for all IOD classes and modalities defined in the DICOM standard (including Encapsulated PDF/CDA and Raw Data).
- **PACS:** Full support for DICOM messaging and secure communication enabling quick implementation of any DICOM SCU and SCP services.
- **Medical Image Viewer:** High level display control with built-in tools for image mark-up, window level, measurement, zoom/pan, cine, and LUT manipulation.
- **Medical Web Viewer Framework:** Plug-in enabled framework to quickly build high-quality, full-featured, web-based medical image delivery and viewer applications.
- **Medical Workstation Framework:** Set of .NET medical and PACS components that can be used to build a full featured PACS Workstation application.
- **3D:** Construct 3D volumes from 2D DICOM medical images and visualize with a variety of methods including MIP, MinIP, MRP, VRT and SSD.
- **Multimedia:** Capture, play, stream and convert MPEG, AVI, WMV, MP4, MP3, OGG, ISO, DVD and more. Stream from RTSP Servers.
- **DVD:** Play, create, convert and burn DVD images.
- **DVR:** Pause, rewind and fast-forward live capture and UDP or TCP/IP streams.
- **MPEG Transport Stream:** With DVR for UDP and TCP/IP streams & auto-live support.

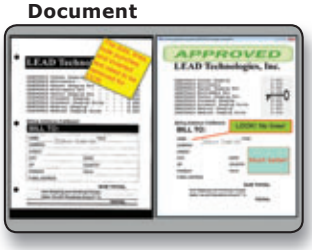
Develop your application with the same robust imaging technologies used by **Microsoft, HP, Sony, Canon, Kodak, GE, Siemens,** the **US Air Force** and **Veterans Affairs Hospitals.**

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

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

LEAD TECHNOLOGIES INCORPORATED

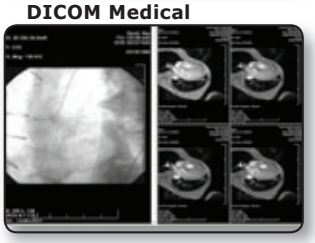
Document



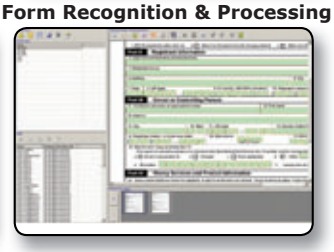
Barcode

Multimedia

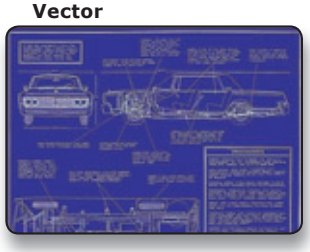
DICOM Medical



Form Recognition & Processing



Vector



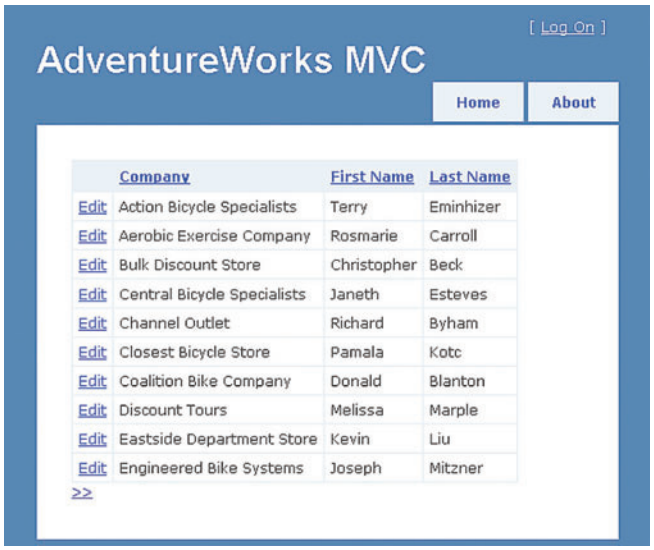


Figure 4 **The First Page of Customer Has Only a Link to Navigate to the Next Page**

GetPagedCustomers method will now return a PagedList class rather than a List. **Figure 2** shows the new version of GetPagedCustomers.

With the new variables populated, let's take a look at how the Index method in the HomeController can push them back to the View. Here's where you can use the new ViewBag. We'll still return the results of the customers query in a View, but you can additionally stuff the values to help determine what the markup will look like for the next and previous links in the ViewBag. They will then be available to the View at run time:

```
public ActionResult Index(int? page)
{
  const int pageSize = 10;
  var customers=DataAccess.GetPagedCustomers((page ?? 0)*pageSize, pageSize);
  ViewBag.HasPrevious = DataAccess.HasPreviousCustomers;
  ViewBag.HasMore = DataAccess.HasMoreCustomers;
  ViewBag.CurrentPage = (page ?? 0);
  return View(customers);
}
```

Figure 5 **A Single Page of Customers with Navigation Links to Go to Previous or Next Page of Customers**

It's important to understand that the ViewBag is dynamic, not strongly typed. ViewBag doesn't really come with HasPrevious and HasMore. I've just made them up as I'm typing the code. So don't be alarmed that IntelliSense doesn't suggest this to you. You can create any dynamic properties you'd like.

If you've been using the ViewPage.ViewData dictionary and are curious how this is different, ViewBag *does* do the same job. But in addition to making your code a little prettier, the properties are typed. For example, HasNext is a dynamic{bool} and CurrentPage is a dynamic{int}. You won't have to cast the values when you retrieve them later.

In the markup, I still have the customer list in the Model variable, but there's a ViewBag variable available as well. You're on your own as you type in the dynamic properties into the markup. A tooltip reminds you that the properties are dynamic, as shown in **Figure 3**.

Here's the markup that uses the ViewBag variables to determine whether or not to display the navigation links:

```
@{ if (ViewBag.HasPrevious)
  {
    @Html.ActionLink("<<", "Index", new { page = (ViewBag.CurrentPage - 1) })
  }
}

@{ if (ViewBag.HasMore)
   { @Html.ActionLink(">>", "Index", new { page = (ViewBag.CurrentPage + 1) })
  }
}
```

This logic is a twist on markup used in the NerdDinner Application Tutorial, which you can find at nerddinnerbook.s3.amazonaws.com/Intro.htm.

Now when I run the app, I have the ability to navigate from one page of customers to the next.

When I'm on the very first page, I have a link to navigate to the next page but nothing to go to a previous page because there is none (see **Figure 4**).

When I click the link and navigate to the next page, you can see that there are now links to go to the previous or next page (see **Figure 5**).

The next step, of course, will be to work with a designer to make this paging more attractive.

Critical Piece of Your Toolbox

Summing up, while there are a number of tools to streamline client-side paging, such as the jQuery DataTables extension and the new ASP.NET MVC 3 WebGrid, your application needs may not always benefit from bringing back large amounts of data. Being able to perform efficient server-side paging is a critical piece of your toolbox. The Entity Framework and ASP.NET MVC work together to provide a great user experience and at the same time simplify your development task to pull this off. ■

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. Lerman blogs at thedatafarm.com/blog and is the author of the highly acclaimed book, "Programming Entity Framework" (O'Reilly Media, 2009). You can follow her at Twitter.com/julielerman.*

THANKS to the following technical expert for reviewing this article: *Vishal Joshi*

YOUR GUIDE TO REPORTS

RDL

Microsoft Report Builder

XML

ComponentOne Report Designer

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

ComponentOne Studio Enterprise

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

ComponentOne

JOSEPH FULTZ

Cloud Services Mashup

Up until now, I've spent time on solutions using Microsoft Windows Azure or SQL Azure to augment solution architecture. This month I'm taking a look at how to combine multiple cloud services into a single app. My example will combine Windows Azure, Windows Azure AppFabric Access Control, Bing Maps and Facebook to provide an example of composing cloud services.

For those who are a little put off when thinking about federated identity or the real-world value of the social network, I'd like to introduce Marcelus. He's a friend of mine who owns a residential and commercial cleaning company. Similar to my father in his business and personal dealings, he knows someone to do or get just about anything you want or need, usually in some form of barter. Some might recognize this as the infamous good ol' boys' network, but I look at Marcelus and I see a living, breathing example of the Windows Azure AppFabric Access Control service (or ACS for short) combined with a powerful social network. In real life I can leverage Marcelus and others like him to help me.

However, in the virtual world, when I use a number of cloud services they often need to know who I am before they allow me to access their functionalities. Because I can't really program Marcelus to serve Web pages, I'm going to use the cloud services in **Figure 1** to provide some functionality.

The scenario is that navigation to my site's homepage will be authenticated by Facebook and the claims will be passed back to my site. The site will then pull that user's friends from Facebook and subsequently fetch information for a selected friend. If the selected friend has a hometown specified, the user may click on the hometown name and the Bing Map will show it.

Configuring Authentication Between Services

The December 2010 issue of *MSDN Magazine* had a good overview article for ACS, which can be found at msdn.microsoft.com/magazine/gg490345. I'll cover the specific things I need to do to federate my site with Facebook. To get this going properly, I'm using AppFabric Labs, which is the developer preview of Windows Azure AppFabric. Additionally, I'm using Windows Azure SDK 1.3 and I've installed Windows Identity Foundation SDK 4.0. To get started, I went to portal.appfabriclabs.com and registered. Once I had access to ACS, I followed the first part of the directions found at the ACS Samples and Documentation (Labs) CodePlex page (bit.ly/fuxkbl) to get the service namespace set up. The next goal was to get Facebook set up as an Identity Provider, but in order to do that I had to first create a Facebook application (see directions at bit.ly/e9yE3l), which results in a summary like that in **Figure 2**.

Figure 1 **Cloud Services and Their Functionalities**

Service	Functionality
Windows Azure	Host my site and serve pages
AppFabric Access Control	Manage and negotiate authentication between my site and Facebook
Facebook	Authenticate users and provide social network services
Bing Maps	Visualize friends' hometowns

This summary page is important, as I'll need to use information from it in my configuration of Facebook as an Identity Provider in ACS. In particular, I'll need the Application ID and the Application secret as can be seen in the configuration information from ACS shown in **Figure 3**.

Note that I've added friends_hometown to the Application permissions text box. I'll need that address to map it, and without specifying it here I wouldn't get it back by default. If I wanted some other data to be returned about the user by the Graph API calls, I'd

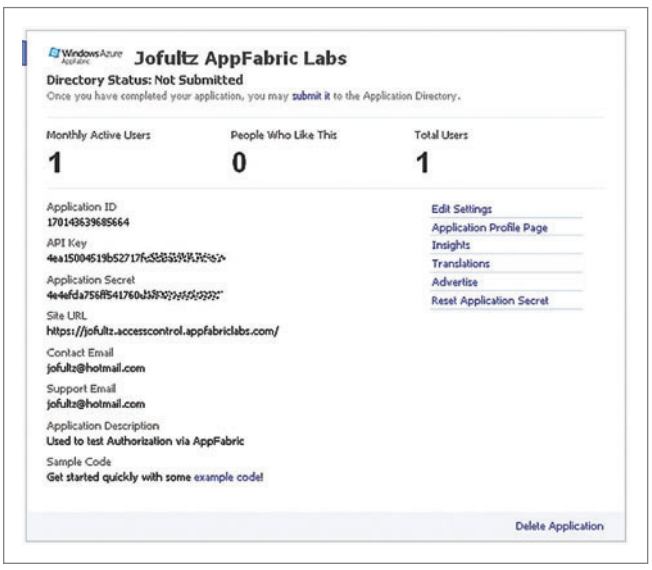


Figure 2 **Facebook Application Configuration Summary**

This article discusses a prerelease version of AppFabric Labs. All information is subject to change.

Code download available at code.msdn.microsoft.com/mag201103Cloudy.

Database Connectivity Solutions

devart

dotConnect

- fast
- comprehensive
- flexible

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

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

WWW.DEVART.COM

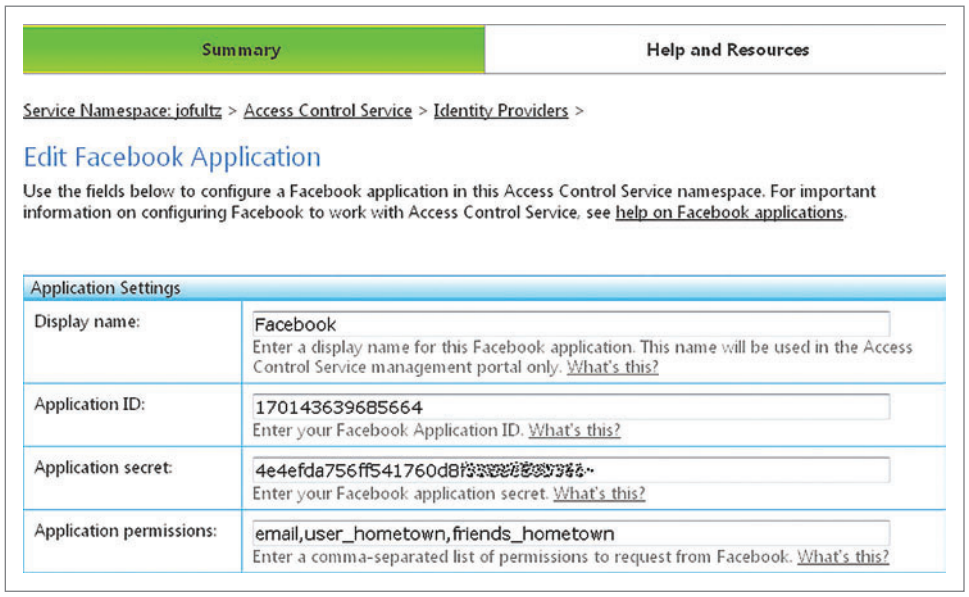


Figure 3 **ACS Facebook Identity Provider Configuration**

need to look it up at the Facebook Developers site (bit.ly/c8UoAA) and include the item in the Application permissions list.

Something worth mentioning when working with ACS: You specify the Relying Parties that will use each Identity Provider. If my site exists at jofultz.cloudapp.net, it will be specified as a relying party on the Identity Provider configuration. This is also true for my localhost. So, in case I don't want to push to the cloud to test it, I'll need to configure a localhost relying party and select it, as illustrated in **Figure 4**.

Something worth mentioning when working with ACS: You specify the Relying Parties that will use each Identity Provider.

Figure 3 and **Figure 4** are both found on the same page for configuring the Identity Provider. By the same token, if I only had it configured for localhost, but then attempted to authenticate from my Web site, it wouldn't work. I can create a custom login page, and there's guidance and a sample for doing so under Application Integration in the ACS management site. In this sample, I'm just taking the default ACS-hosted page.

So far I've configured ACS and my Facebook application to get them talking once invoked. The next step is to configure this Identity Provider for my site as a means of authentication. The easiest way to do this is to install the Windows Identity Foundation SDK 4.0 found at bit.ly/ew6K5z. Once installed, there will be a right-click menu option available to Add STS reference, as illustrated in **Figure 5**.

In my sample I used a default ASP.NET site created in Visual Studio by selecting a new Web Role project. Once it's created, I right-click on the site and go about stepping through the wizard. I'll configure the site to use an existing Security Token Service (STS) by choosing that option in the wizard and providing a path to the WS-Federation metadata. So, for my access control namespace, the path is:

```
jofultz.accesscontrol.appfabriclabs.com/
  FederationMetadata/2007-06/
  FederationMetadata.xml
```

Using this information, the wizard will add the config section <microsoft.identityModel/> to the site configuration. Once this is done, add <httpRuntime requestValidationMode="2.0" /> underneath the <system.web/> element. Providing that I specified localhost as a relying party, I should be able to run the application, and upon startup be presented with an ACS-hosted login page that will present Facebook—or Windows Live or Google, if so configured. The microsoft.identityModel element is dependent upon existence of the Microsoft.Identity assembly, so you have to be sure to set that DLL reference in the site to Copy Always. If it isn't, once it's pushed to Windows Azure it won't have the DLL and the site will fail to run. Referring to my previous statement about needing to have configuration for localhost and the Windows Azure hosted site, there's one more bit of configuration once the wizard is complete. Thus, if the wizard was configured with the localhost path, then a path for the Windows Azure site will need to be added to the <audienceUris> element as shown here:

```
<microsoft.identityModel>
  <service>
    <audienceUris>
      <add value="http://jofultz.cloudapp.net/" />
      <add value="http://localhost:81/" />
    </audienceUris>
```

Additionally, the realm attribute of the wsFederation element in the config will need to reflect the current desired runtime location. Thus, when deployed to Windows Azure, it looks like this for me:

```
<federatedAuthentication>
  <wsFederation passiveRedirectEnabled="true" issuer=
   "https://jofultz.accesscontrol.appfabriclabs.com/v2/wsfederation"
   realm="http://jofultz.cloudapp.net/" requireHttps="false" />
  <cookieHandler requireSsl="false" />
</federatedAuthentication>
```

Figure 4 **ACS Facebook Identity Provider Configuration: Relying Parties**

However, if I want to debug this and have it work properly at run time on my localhost (for local debugging), I'll change the realm to represent where the site is hosted locally, such as the following:

```
<federatedAuthentication>
  <wsFederation passiveRedirectEnabled="true"
   issuer="https://jofultz.accesscontrol.
   appfabriclabs.com/v2/wsfederation"
   realm="http://localhost:81/"
   requireHttps="false" />
  <cookieHandler requireSsl="false" />
</federatedAuthentication>
```

With everything properly configured, I should be able to run the site, and upon attempting to browse to the default page I'll be redirected to the ACS-hosted login page, where I can choose Facebook as the Identity Provider. Once I click Facebook, I'm sent to the Facebook login page to be authenticated (see **Figure 6**).

Because I haven't used my application before, Facebook presents me with the Request Permission dialog for my application, as seen in **Figure 7**.

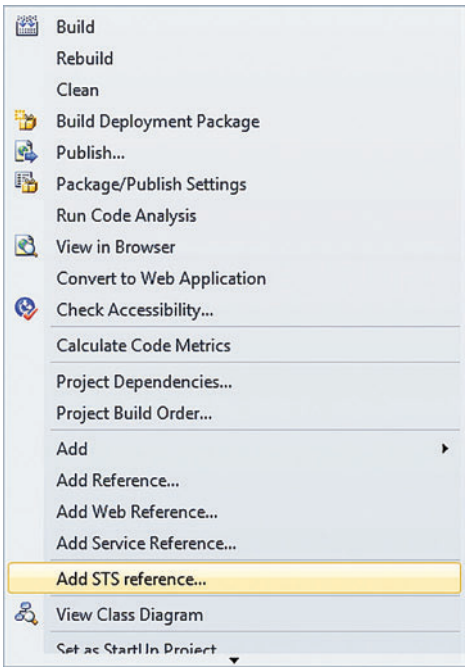


Figure 5 **Add STS Reference Menu Option**

and use. In order to make the requests, I'll need an Access Token. Fortunately, it was passed back in the claims, and with the help of the Windows Identity Foundation SDK, the claims have been placed into the principal identity. The claims look something like this:

```
http://schemas.xmlsoap.org/ws/2005/05/
  identity/claims/nameidentifier
http://schemas.microsoft.com/ws/2008/06/
  identity/claims/expiration
http://schemas.xmlsoap.org/ws/2005/05/
  identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/
  identity/claims/name
http://www.facebook.com/claims/AccessToken
http://schemas.microsoft.com/
  accesscontrolservice/2010/07/claims/
  identityprovider
```

What I really want out of this is the last part of the full name (for example, nameidentifier, expiration and so on) and the related value. So I create the ParseClaims method to tease apart the claims and place them and their values into a hash table for further use, and then call that method in the page load event:

```
protected void ParseClaims()
{
  string username = default(string);
  username = Page.User.Identity.Name;

  IClaimsPrincipal Principal = (IClaimsPrincipal) Thread.CurrentPrincipal;
  IClaimsIdentity Identity = (IClaimsIdentity) Principal.Identity;

  foreach (Claim claim in Identity.Claims)
  {
    string[] ParsedClaimType = claim.ClaimType.Split('/');
    string ClaimKey = ParsedClaimType[ParsedClaimType.Length - 1];

    _Claims.Add(ClaimKey, claim.Value);
  }
}
```

I create an FBHelper class where I'll create the methods to access the Facebook information that I desire. To start, I create a method to help make all of the needed requests. I'll make each

> For my application, I need to fetch the friends that comprise my social network and then subsequently retrieve information about those friends.

Not wanting to be left out of the inner circle of those who use such a fantastic app, I quickly click Allow, after which Facebook, ACS and my app exchange information (via browser redirects) and I'm finally redirected to my application. At this point I've simply got an empty page, but it does know who I am and I have a "Welcome Joseph Fultz" message at the top right of the page.

Facebook Graph API

For my application, I need to fetch the friends that comprise my social network and then subsequently retrieve information about those friends. Facebook has provided the Graph API to enable developers to do such things. It's pretty well-documented, and best of all, it's a flat and simple implementation, making it easy to understand

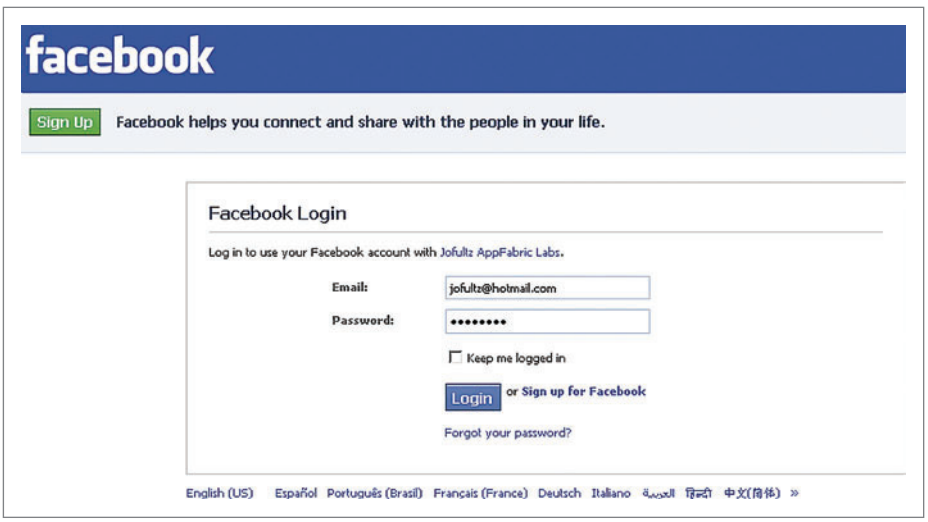


Figure 6 **Facebook Login**

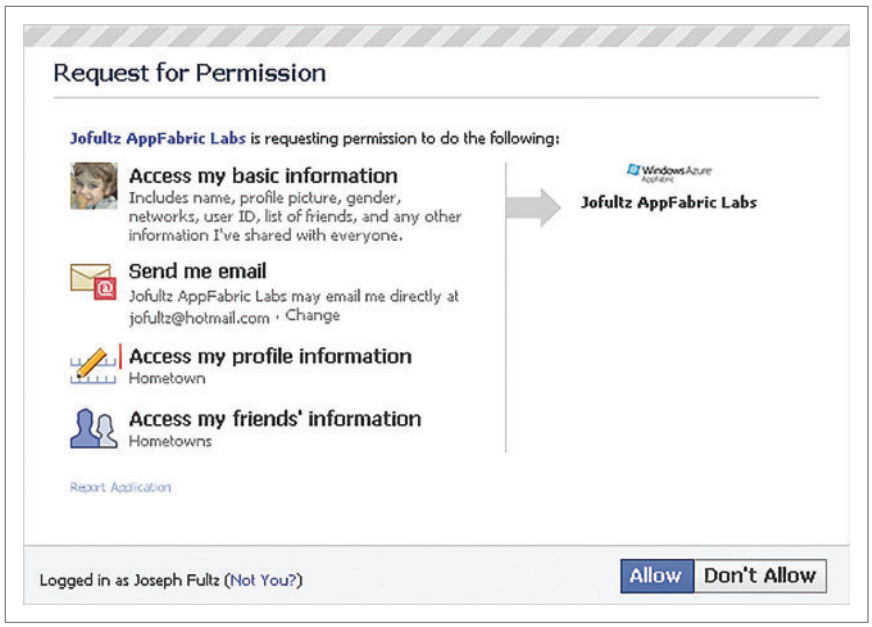


Figure 7 **Application Permission Request**

request using the WebClient object and parse the response with the JavaScript Serializer:

```
public static Hashtable MakeFBRequest(string RequestUrl)
{
  Hashtable ResponseValues = default(Hashtable);

  WebClient WC = new WebClient();
  Uri uri = new Uri(String.Format(RequestUrl, fbAccessToken));

  string WCResponse = WC.DownloadString(uri);
  JavaScriptSerializer JSS = new JavaScriptSerializer();
  ResponseValues = JSS.Deserialize<Hashtable>(WCResponse);

  return ResponseValues;
}
```

> Using ACS, I was able to create a sample application from a composite of cloud technology.

As seen in this code snippet, each request will need to have the Access Token that was passed back in the claims. With my reusable request method in place, I create a method to fetch my friends and parse them into a hash table containing each of their Facebook IDs and names:

```
public static Hashtable GetFBFriends(string AccessToken)
{
  Hashtable FinalListOfFriends = new Hashtable();
  Hashtable FriendsResponse = MakeFBRequest(_fbFriendsListQuery, AccessToken);
  object[] friends = (object[])FriendsResponse["data"];

  for (int idx = 0; idx < friends.Length;idx++ )
  {
    Dictionary<string, object> FriendEntry =
      (Dictionary<string, object>)friends[idx];
    FinalListOfFriends.Add(FriendEntry["id"], FriendEntry["name"]);
  }
  return FinalListOfFriends;
}
```

The deserialization of the friends list response results in a nested structure of Hashtable->Hashtable->Dictionary. Thus I have to do a little work to pull the information out and then place it into my own hash table. Once it's in place, I switch to my default.aspx page, add a ListBox, write a little code to grab the friends and bind the result to my new ListBox:

```
protected void GetFriends()
{
  _Friends = FBHelper.GetFBFriends((string)_
    Claims["AccessToken"]);
  this.ListBox1.DataSource = _Friends;
  ListBox1.DataTextField = "value";
  ListBox1.DataValueField = "key";
  ListBox1.DataBind();
}
```

If I run the application at this point, once I'm authenticated I'll see a list of all of my Facebook friends. But wait—there's more! I need to get the available information for any selected friend so that I can use that to show me their hometown on a map. Flipping back to my FBHelper class, I add a simple method that will take the Access Token and the ID of the selected friend:

```
public static Hashtable GetFBFriendInfo(string AccessToken, string ID)
{
  Hashtable FriendInfo =
    MakeFBRequest(String.Format(_fbFriendInfoQuery, ID) +
    "?access_token={0}", AccessToken);
  return FriendInfo;
}
```

Note that in both of the Facebook helper methods I created, I reference a constant string that contains the needed Graph API query:

```
public const string _fbFriendsListQuery =
  "https://graph.facebook.com/me/friends?access_token={0}";
public const string _fbFriendInfoQuery = "https://graph.facebook.com/{0}/";
```

With my final Facebook method in place, I'll add a GridView to the page and set it up to bind to a hash table, and then—in the code-behind in the SelectedIndexChanged method for the ListBox—I'll bind it to the Hashtable returned from the GetFBFriendInfo method, as shown in **Figure 8**.

Now that I've got my friends and their info coming back from Facebook, I'll move on to the part of showing their hometown on a map.

Figure 8 **Adding a GridView**

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
  Debug.WriteLine(ListBox1.SelectedValue.ToString());
  Hashtable FriendInfo =
    FBHelper.GetFBFriendInfo((string)_Claims["AccessToken"],
    ListBox1.SelectedValue.ToString());
  GridView1.DataSource = FriendInfo;
  GridView1.DataBind();
  try
  {
    Dictionary<string, object> HometownDict =
      (Dictionary<string, object>) FriendInfo["hometown"];
      _Hometown = HometownDict["name"].ToString();
  }
  catch (Exception ex)
  {
    _Hometown = "";//Not Specified";
  }
}
```

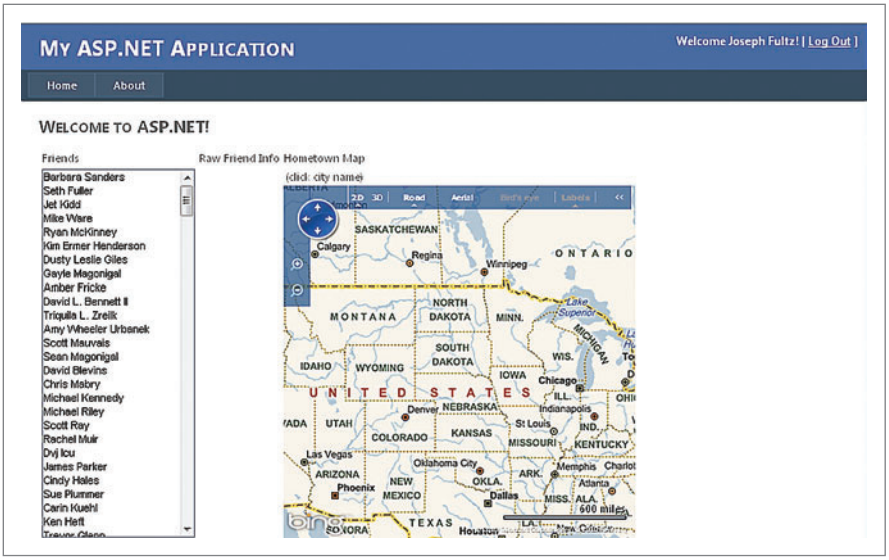


Figure 9 **Demo Homepage**

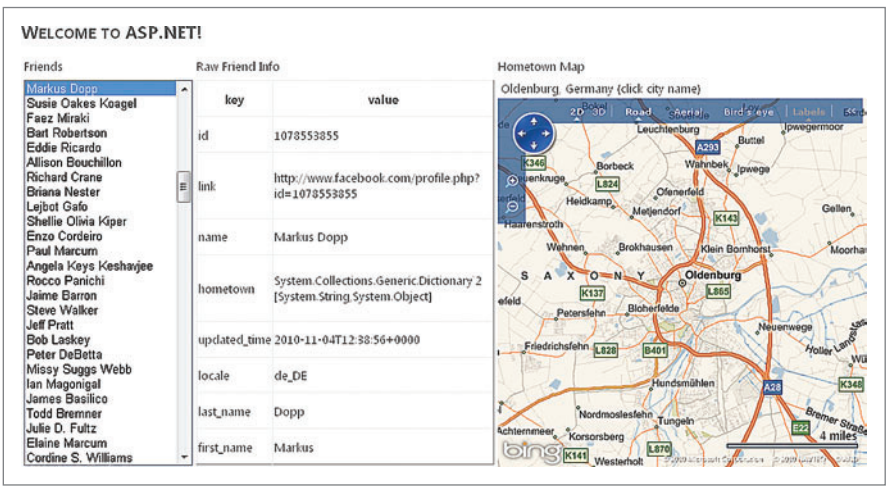


Figure 10 **Hometown in Bing Maps**

There's No Place Like Home

For those of my friends who have specified their hometown, I want to be able to click on the hometown name and have the map navigate there. The first step is to add the map to the page. This is a pretty simple task and, to that end, Bing provides a nice interactive SDK that will demonstrate the functionality and then allow you to look at and copy the source. It can be found at microsoft.com/maps/isdk/ajax/. To the default.aspx page, I add a div to hold the map, like this:

```
<div  id="myMap" style="position:relative; width:400px; height:400px;" ></div>
```

However, to get the map there, I add script reference and a little bit of script to the SiteMaster page:

```
<script type="text/javascript" src="http://ecn.dev.virtualearth.net/
  mapcontrol/mapcontrol.ashx?v=6.2"></script>
  <script type="text/javascript">
    var map = null;
    function GetMap() {
      map = new VEMap('myMap');
      map.LoadMap();
    }
  </script>
```

With that in place, when I pull up the page I'll be presented with a map on the default position—but I want it to move to my friend's hometown when I select it. During the SelectedIndexChanged event discussed earlier, I also bound a label in the page to the name and added a client-side click event to have the map find a location based on the value of the label:

```
onclick="map.Find(null, hometown.innerText,
  null, null, null, null, true, null, true);
  map.SetZoomLevel(6);"
```

In the map.Find call, most of the trailing parameters could be left off if so desired. The reference for the Find method can be found at msdn.microsoft.com/library/bb429645. That's all that's needed to show and interact with the map in this simple example. Now I'm ready to run it in all of its glory.

If I've configured the identityModel properly to work with my localhost as mentioned earlier, I can press F5 and run it locally in debug. So, I hit F5, see a browser window pop up, and there I'm presented with my login options. I choose Facebook and I'm taken to the login page shown in **Figure 6**. Once logged in, I'm directed back to my default.aspx page, which now displays my friends and a default map like that in **Figure 9**.

Next, I'll browse through my friends and click one. I'll get the information available to me based on his security settings and the application permissions I requested when I set up the Identity Provider as seen in **Figure 2**. Next, I'll click in the hometown name located above the map and the map will move to center on the hometown, as seen in **Figure 10**.

Final Thoughts

I hope I've clearly articulated how to bring together several aspects of the Windows Azure platform, Bing Maps and Facebook—and that I've shown how easy it is. Using ACS, I was able to create a sample application from a composite of cloud technology. With a little more work, it's just as easy to tie in your own identity service to serve as it's needed. The beauty in this federation of identity is that using Windows Azure enables you to develop against and incorporate services from other vendors and other platforms—versus limiting you into a single choice of provider and that provider's services, or having to figure out a low-fidelity integration method. There's power in the Microsoft Windows Azure platform, and part of that power is how easily it can be mashed together with other cloud services. ■

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

THANKS to the following technical expert for reviewing this article: *Steve Linehan*

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.

Windows CE · android · Linux

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

Wi Fi · Bluetooth · Made in Canada

Learn more at www.devtouchpro.com

Windows 7 · Certified for Windows Server 2008

More Development Tools Available at:

www.amyuni.com

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

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

AMYUNI Technologies

Microsoft Silverlight Microsoft Visual Studio

Your Data in a Whole New Light

ComponentArt

Data Visualization for .NET

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

Charting **Gauges** **Maps** **GridView** **TimeNavigator** **CalcEngine**

www.componentart.com

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

ComponentArt
Build Something Amazing

Cloud-Based Collaboration with SharePoint Online

Chris Mayo

With the release of Office 365, Microsoft will present the next version of Microsoft Online Services, a cloud-based collaboration and communication service based on SharePoint 2010, Exchange 2010 and Lync Server 2010. Office 365, currently in beta, will provide SharePoint, Exchange and Lync as a subscription-based Software as a Service (SaaS) offering, hosted in cloud datacenters managed by Microsoft.

SharePoint Online—the cloud-based version of SharePoint 2010—will provide users with many of the same features of SharePoint 2010, but without the need to manage the hardware or software required for a scalable and secure collaboration solution. In this article, I'll provide an overview of how SharePoint Online development is similar to and different from SharePoint 2010 development by building solutions that run in SharePoint Online.

With the next release of SharePoint Online, SharePoint developers will be able to develop collaboration solutions using the same skills and tools they use in developing for SharePoint 2010, including Visual Studio 2010, SharePoint Designer 2010, C# or Visual Basic and the SharePoint APIs and SDKs. There are many similarities between developing for SharePoint on-premises and in the cloud, but there are also significant differences that will impact how you build solutions.

Understanding these differences will help you understand what solutions can be created to run in SharePoint Online and how to develop those solutions.

This article is based on prerelease versions of Office 365 and SharePoint Online. All information is subject to change.

This article discusses:

- SharePoint Online customization similarities and differences
- Developing for SharePoint Online with sandboxed solutions
- Creating client-side solutions with Silverlight

Technologies discussed:

SharePoint Online, SharePoint 2010

Code download available at:

code.msdn.microsoft.com/mag201102SPOnline

SharePoint Online Customization Similarities

In SharePoint 2010 development, you have the ability to customize SharePoint using the browser and SharePoint Designer 2010 and by building solutions using Visual Studio 2010. With SharePoint Online, customization with the browser and SharePoint Designer 2010 is largely the same as SharePoint 2010 (given the feature differences mentioned in the next section). Developing SharePoint Online solutions using Visual Studio 2010 is also largely the same. Development is done in Visual Studio 2010 against a local instance of SharePoint 2010 (either running locally in Windows 7 or Windows Server 2008 R2 or in a virtual machine, or VM), leveraging the integrated debugging experience for iterative development. When

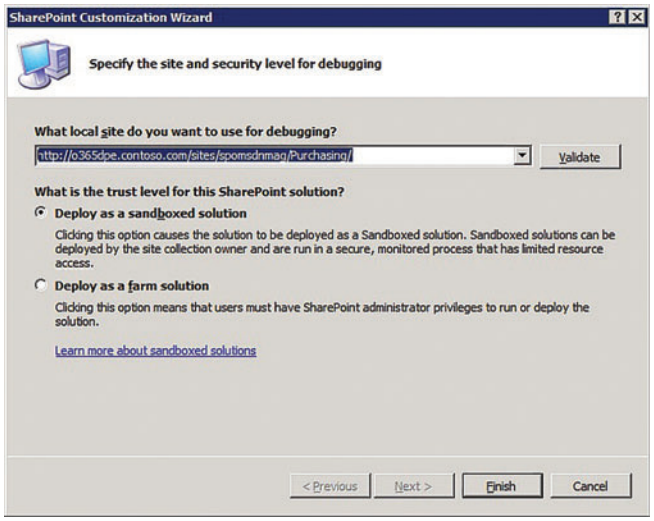


Figure 1 **Specify the Site and Trust Level for PurchasingMgr**

development is complete, the solution is uploaded to SharePoint Online using the same Solution Gallery provided in SharePoint 2010.

SharePoint Online Customization Key Differences

Although SharePoint Online is based on SharePoint 2010, there are some key differences to keep in mind as you develop solutions that will run in the former. First, SharePoint Online only supports Site- and Web-scoped solutions. It runs in a multi-tenant cloud, where multiple tenancies run on a shared datacenter infrastructure, so it make sense that solutions with Farm scope (where a feature is activated for the entire farm) aren't supported. Likewise, in SharePoint Online, the highest level of access to your SharePoint tenancy is at the site-collection level, so WebApplication-scoped features (where a feature runs in every Web site in a Web application) aren't supported either.

Second, only partial-trust solutions are supported in SharePoint Online. Full-trust solutions, where your solution would have access beyond the site-collection level or could be granted permission to run with admin-level privileges on the farm, also aren't supported.

Finally, although SharePoint Online is based on SharePoint 2010, it doesn't have 100 percent feature parity with its on-premises counterpart. For a complete feature-by-feature comparison between SharePoint 2010 and SharePoint Online, refer to the Microsoft SharePoint Online Beta Service Description, available from the Office 365 Beta Service Descriptions page at bit.ly/bBckol.

The feature-by-feature list shows that a majority of SharePoint customization features are supported. The lack of support for Business Connectivity Services (BCS), External Lists and the ability to call Web services outside SharePoint Online (which isn't supported in partial-trust solutions) will have a significant impact on building solutions that run in SharePoint Online. BCS support is planned for a future release, however.

With these similarities and differences in mind, let's look at some examples of the types of solutions you can build to run in SharePoint

Figure 2 **Item Templates Supported in Sandboxed Solutions**

Item Template	Sandbox Compatible?	Notes
Visual Web Part	No	Requires ASCX file be installed on SharePoint Servers
Visual Web Part (Sandboxed)	Yes	Provided by installing the Visual Studio 2010 SharePoint Power Tools
Web Part	Yes	
Sequential Workflow	No	Requires workflow solution be deployed as Farm Solution
State Machine Workflow	No	Requires workflow solution be deployed as Farm Solution
Business Data Connectivity Model	No	Requires BCS solution be deployed as a full-trust solution; feature not supported in SharePoint Online
Application Page	No	Requires ASPX page be deployed to SharePoint Server
Event Receiver	Yes	
Module	Yes	
Content Type	Yes	
List Definition from Content Type	Yes	
List Definition	Yes	
List Instance	Yes	
Empty Element	Yes	
User Control	No	Requires ASCX file to be installed on SharePoint Servers

Figure 3 **NonStandBusPurchaseRequestsCT Definition via Elements.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <Field SourceID="http://schemas.microsoft.com/sharepoint/v3"
    ID="{A74E67E5-8905-4280-90C9-DEBFFC30D43D}"
    Name="RequestDescription"
    DisplayName="Description"
    Group="Purchasing Manager Custom Columns"
    Type="Note"
    DisplaceOnUpgrade="TRUE" />
  <Field SourceID="http://schemas.microsoft.com/sharepoint/v3"
    ID="{CB5054F5-0C60-4DBE-94D2-CEFBFB793C7F}"
    Name="Price"
    DisplayName="Price"
    Group="Purchasing Manager Custom Columns"
    Type="Currency"
    DisplaceOnUpgrade="TRUE" />

  <!-- Parent ContentType: Item (0x01) -->
  <ContentType ID="0x010078a81c8413f54917856495e56e7c09ed"
    Name="Purchasing Manager - Non-Standard Business Purchase Requests Content Type"
    Group="Purchasing Manager Content Types"
    Description=
      "Non-Standard Business Purchase Requests Content Type
       for the Purchasing Manager Solution"
    Inherits="TRUE"
    Version="0">
    <FieldRefs>
      <FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" Name="Title"
        DisplayName="Title" />
      <FieldRef ID="{A74E67E5-8905-4280-90C9-DEBFFC30D43D}"
        Name="RequestDescription"
        Required="TRUE" />
      <FieldRef ID="{CB5054F5-0C60-4DBE-94D2-CEFBFB793C7F}" Name="Price"
        Required="TRUE" />
    </FieldRefs>
  </ContentType>
</Elements>
```

Figure 4 **NonStandBusPurchaseRequestsListDefn Definition via Elements.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- Do not change the value of the Name attribute below.
    If it does not match the folder name of the List Definition project item,
    an error will occur when the project is run. -->
    <ListTemplate
      Name="NonStandBusPurchaseRequestsListDefn"
      Type="10051"
      BaseType="0"
      OnQuickLaunch="TRUE"
      SecurityBits="11"
      Sequence="410"
      DisplayName="Purchasing Manager -
        Non-Standard Business Purchase Requests List Definition"
      Description=
        "Non-Standard Business Purchase Requests List Definition
         for the Purchasing Manager Solution"
      Image="/_layouts/images/itgen.png"/>
</Elements>
```

Figure 5 **Adding Custom Columns to the NonStandBusPurchaseRequestsListDefn Default View**

```
<View BaseViewID="1" Type="HTML" WebPartZoneID="Main"
  DisplayName="$Resources:core,objectiv_schema_mwsidcamlidC24;"
  DefaultView="TRUE" MobileView="TRUE" MobileDefaultView="TRUE"
  SetupPath="pages\viewpage.aspx" ImageUrl="/_layouts/images/generic.png"
  Url="AllItems.aspx">
  <Toolbar Type="Standard" />
  <XslLink Default="TRUE">main.xsl</XslLink>
  <RowLimit Paged="TRUE">30</RowLimit>
  <ViewFields>
    <FieldRef Name="Attachments">
    </FieldRef>
    <FieldRef Name="LinkTitle">
    </FieldRef>
    <FieldRef ID="{A74E67E5-8905-4280-90C9-DEBFFC30D43D}"
      Name="RequestDescription" />
    <FieldRef ID="{CB5054F5-0C60-4DBE-94D2-CEFBFB793C7F}" Name="Price" />
  </ViewFields>
  <Query>
    <OrderBy>
      <FieldRef Name="ID">
      </FieldRef>
    </OrderBy>
  </Query>
  <ParameterBindings>
    <ParameterBinding Name="NoAnnouncements"
      Location="Resource(wss,noXinviewofY_LIST)" />
    <ParameterBinding Name="NoAnnouncementsHowTo"
      Location="Resource(wss,noXinviewofY_DEFAULT)" />
  </ParameterBindings>
</View>
```

Online, including sandboxed solutions and the SharePoint client object model (OM). Other solution types, such as automating business processes via declarative workflow solutions, will be covered in future articles.

Developing for SharePoint Online with Sandboxed Solutions

From the previous section, you know that SharePoint Online solutions must be scoped to site or Web features, are restricted to data in the site collection and must run in partial trust. Developing solutions that run as sandboxed solutions meet all these criteria while letting SharePoint Online administrators easily deploy a solution by uploading it directly to the Solution Gallery.

Visual Studio 2010 provides great support for sandboxed solutions, including project template and project item template support, the SharePoint Customization Wizard for creating new projects as sandboxed solutions, IntelliSense support for the site collection-scoped SharePoint APIs, and debugging and packaging support. To get started building a solution for SharePoint Online, you'll develop and debug the solution locally against SharePoint 2010. You'll need either Windows 7 64-bit or Windows Server 2008 R2 installed along with SharePoint 2010 and Visual Studio 2010. Another great way to get started is to use the 2010 Information Worker Demonstration and Evaluation Virtual Machine (RTM), which provides a virtualized SharePoint 2010 development environment (download it from bit.ly/ezfe2Y). I also recommend the Visual Studio 2010 SharePoint Power Tools (bit.ly/azq882), which add compile-time support for the sandbox and a sandboxed Visual Web Part project item template.

In the examples in this article, I'll build a solution using the simple scenario of providing the employees of the fictional Contoso Corp. with the ability to request purchases that aren't supported in their procurement system. To get started, I'll create a site collection and site in my on-premises SharePoint 2010 development environment. I'm using the VMs mentioned previously, so I've created http://o365dpe.contoso.com/sites/spomsdnmag/purchasing. My first solution will deploy the list used to track these non-standard purchases. I'll open Visual Studio 2010, select File | New Project, and in the New Project dialog I'll select Empty SharePoint Project and name the project PurchasingMgr.

In the SharePoint Customization Wizard dialog, for "What local site …," I'll enter the URL of my site, http://o365dpe.contoso.com/sites/spomsdnmag/Purchasing/, select "Deploy as a sandboxed solution" and click Finish as seen in **Figure 1**.

Next, I'll select the PurchasingMgr project in Solution Explorer, right-click and select Add | New Item. In the Add New Item dialog, I'll select SharePoint 2010 in the Installed Templates node to the supported

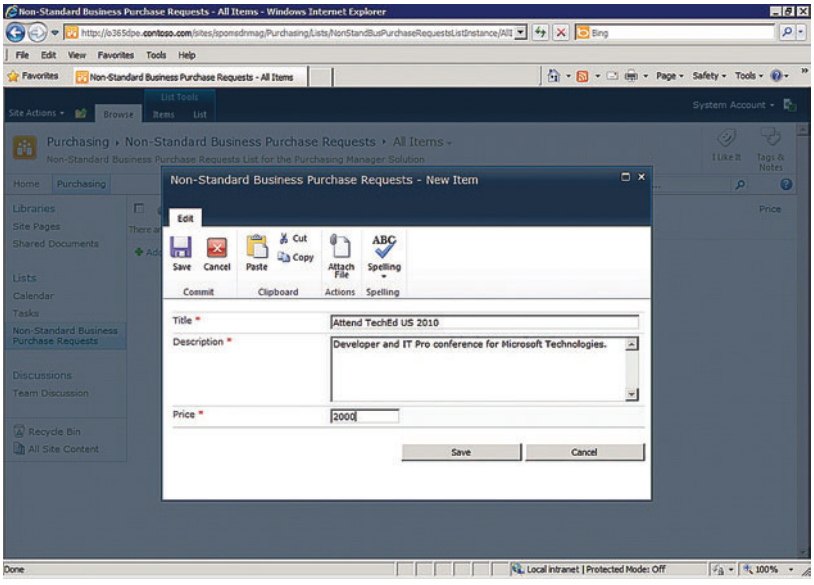


Figure 6 **Debugging the PurchasingMgr Solution**

RadControls *for*

ASP.NET AJAX

The industry's leading AJAX components.

Try it for yourself: www.telerik.com/ajaxMSDN

Many Fortune 500 companies chose RadControls for ASP.NET AJAX. For its long-lasting history of leading the market, for its comprehensiveness, for its vast .NET community.
Try it for yourself and feel the Telerik difference at **www.telerik.com/ajaxMSDN**

Windows IT Pro | SQL Server
Best of Microsoft tech·ed North America 2010
SharePoint Pro | DevProConnections

2010 Editors Choice Visual Studio Magazine

SD Times 2010 100

Members Choice Winner The Code Project 2010

Microsoft Partner Network

2010
Microsoft Central & Eastern Europe
PARTNER OF THE YEAR
Winner

telerik
deliver more than expected

SharePoint item templates. Not all of these templates are supported under sandboxed solutions and therefore in SharePoint Online. **Figure 2** shows the item templates supported under sandboxed solutions.

To build my list, I'll define Site Columns and a Content Type for the list by selecting the Content Type item template and entering NonStandBusPurchaseRequestsCT for the name.

In the SharePoint Customization Wizard, I'll select Item as the base content type and click Finish. The Content Type will have a Title column, a Description column and a Price column, which I'll define declaratively by replacing the contents of the Elements.xml created with the XML in **Figure 3**.

Next, I'll define a list based on that content type by right-clicking PurchasingMgr in Solution Explorer and selecting Add New Item. I'll select the List Definition from Content Type item template and name the list definition NonStandBusPurchaseRequestsListDefn and click Add.

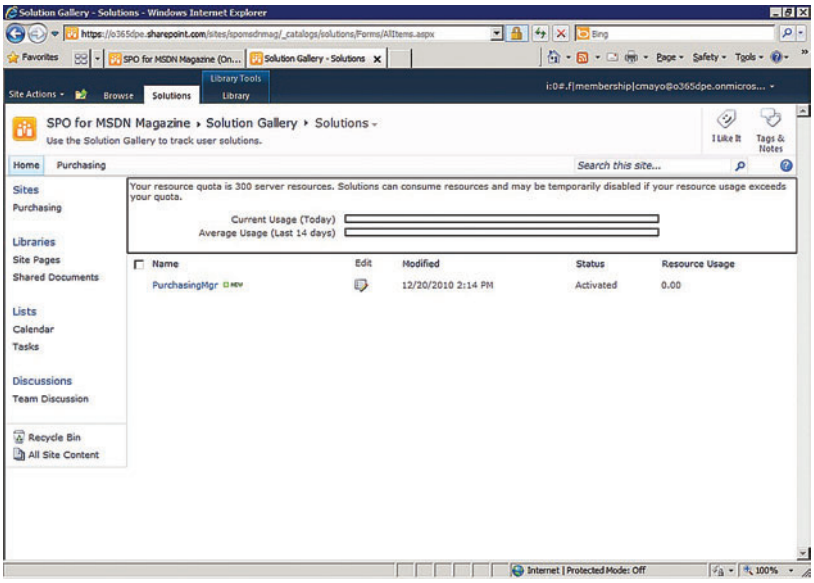


Figure 7 **PurchasingMgr Solution Deployed to SharePoint Online**

In the SharePoint Customization Wizard, I'll select the content type created earlier and check the "Add a list instance" box. The Elements.xml created for the NonStandBusPurchaseRequestsListDefn is shown in **Figure 4**.

Note that each list definition in my feature needs to be identified by a unique Type value greater than 10,000 (to avoid conflicts with lists defined by SharePoint), and that I use that value in defining any list instance based on that definition.

To add the custom columns to the list view, I'll open Schema.xml created and add the FieldRef elements to the default view, as seen in **Figure 5**.

Finally, I'll define an instance of the list by selecting ListInstance1 under NonStandBusPurchaseRequestsListDefn and rename it NonStandBusPurchaseRequestsListInstance. I'll open Elements.xml and add the following XML to base the list on the content type and to provide helpful descriptions for users:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ListInstance Title="Non-Standard Business Purchase Requests"
    OnQuickLaunch="TRUE"
    TemplateType="10051"
    Url="Lists/NonStandBusPurchaseRequestsListInstance"
    Description=
      "Non-Standard Business Purchase Requests List
       for the Purchasing Manager Solution">
  </ListInstance>
</Elements>
```

In Visual Studio 2010, I'll select Debug and then Start Debugging to test the solution. The solution is packaged and deployed to my on-premises site, as seen in **Figure 6**.

Now that I've tested the PurchasingMgr solution, I'm ready to deploy it to SharePoint Online. I'll create a new site collection in SharePoint Online named Purchasing using the Team Site template. Back in Visual Studio 2010, I'll package the solution by right-clicking on the PurchasingMgr project in the Solution Explorer and selecting Package. To deploy the solution to SharePoint Online, I'll just need to upload it to the Solution Gallery and activate the site features (I'll need site collection administrator privileges to do so). To do this, I'll log in to SharePoint Online and navigate to my site Collection and select Site Actions | Site Settings, and then Solutions to access the Solutions Gallery. In the Solution Gallery, I'll click the Solutions tab and select Upload Solution in the ribbon, then browse to the PurchasingMgr.wsp file in bin\Debug and click OK and then Activate. You'll see your solution in the Solution Gallery as seen in **Figure 7**.

Next, to activate the feature that contains my site columns, content type and list, I'll navigate to the Purchasing site and select Site Actions | Site Settings | Manage Site Features. I'll select the

Figure 8 **NonStandBusPurchaseReqsSLOM MainPage.xaml**

```
<UserControl xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/
presentation/sdk"
 x:Class="NonStandBusPurchaseReqsSLOM.MainPage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 mc:Ignorable="d"
 d:DesignHeight="300" d:DesignWidth="400">

 <Grid x:Name="LayoutRoot" Background="White">
   <Grid.ColumnDefinitions>
     <ColumnDefinition />
     <ColumnDefinition />
   </Grid.ColumnDefinitions>
     <Grid.RowDefinitions>
       <RowDefinition Height="Auto"/>
       <RowDefinition Height="Auto"/>
       <RowDefinition Height="Auto"/>
       <RowDefinition Height="Auto"/>
     </Grid.RowDefinitions>

     <sdk:Label Content="Title:" Grid.Column="0" Grid.Row="0" Margin="3"/>
     <sdk:Label Content="Description:" Grid.Column="0" Grid.Row="1" Margin="3"/>
     <sdk:Label Content="Price:" Grid.Column="0" Grid.Row="2" Margin="3"/>

     <TextBox Name="Title" Grid.Column="1" Grid.Row="0" Margin="3"/>
     <TextBox Name="Description" Grid.Column="1" Grid.Row="1" Margin="3"/>
     <TextBox Name="Price" Grid.Column="1" Grid.Row="2" Margin="3"/>

     <Button Content="Add" Grid.Column="1" Grid.Row="3" Margin="3"
       Name="addNonStanPurchaseReq" HorizontalAlignment="Right"
       Height="25" Width="100" Click="addNonStanPurchaseReq_Click" />
 </Grid>
</UserControl>
```

Telerik Extensions *for*

ASP.NET MVC

Fast, clean, rich UI. Open source and free.

Free download at: www.telerik.com/razor

- *18 native ASP.NET MVC extensions*
- *jQuery-based for minimal client-side footprint*
- *MVC 3 and Razor ready*
- *14 ready-to-use themes*

Best of tech·ed 2010 | Visual Studio Editors Choice 2010 | SD Times 100 2010 | The Code Project Members Choice Winner 2010 | Microsoft Partner Network 2010 Microsoft Central & Eastern Europe PARTNER OF THE YEAR Winner

telerik
deliver more than expected

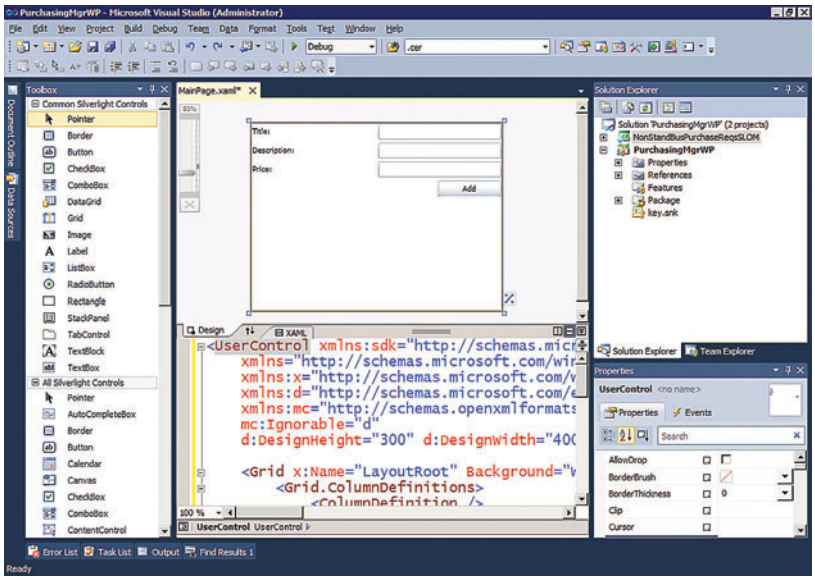


Figure 9 **MainPage.xaml in Designer**

Creating Client-Side Solutions with Silverlight

The client OM, also introduced with SharePoint 2010, provides an object-oriented, client-side API for SharePoint clients built using the Microsoft .NET Framework, Silverlight and ECMAScript (including JavaScript and JScript) that run on remote computers (including the browser for Silverlight and ECMAScript). The API is consistent with the Microsoft.SharePoint server-side namespace, so it's easy to learn. The API is also consistent across the supported client types, so it's easy to apply that knowledge across different client solutions. The client OM API is supported in SharePoint Online and is a valuable tool for cloud development.

For example, I can use the client OM to create a Silverlight 4 application to add items to my list and host the application in a sandboxed Web Part. To do this, I'll open Visual Studio 2010, select File | New Project and in the New Project dialog select Empty SharePoint Project. I'll name the project PurchasingMgrWP and click OK. Again, I'll create the solution as a sandboxed solution and point it at my on-premises Purchasing site. To create the Silverlight 4 application, I'll right-click on the PurchasingMgrWP solution, select Silverlight under Installed Templates, select Silverlight Application and name the solution NonStandBusPurchaseReqsSLOM. In the New Silverlight Application dialog, I'll uncheck "Host the Silverlight application in a new Web Site"

Purchasing Manager - Content Types and Lists feature and select Activate. At this point you should see the Non-Standard Business Purchase Requests lists in your SharePoint Online site.

The Purchasing Manager is just one example of what you can accomplish in SharePoint Online with sandboxed solutions. Keep in mind the limitations in sandboxed solutions and in the SharePoint Online supported features, and you can create solutions that will run in SharePoint 2010 or SharePoint Online.

Figure 10 **addNonStanPurchaseReq_Click**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

using Microsoft.SharePoint.Client;

namespace NonStandBusPurchaseReqsSLOM
{
  public partial class MainPage : UserControl
  {
    private string webUrl;

    public MainPage(string url)
    {
      webUrl = url;

      InitializeComponent();
    }

    private void addNonStanPurchaseReq_Click(object sender, RoutedEventArgs e)
    {
      ClientContext clientContext = new ClientContext(webUrl);

      Web webSite = clientContext.Web;
      ListCollection webLists = webSite.Lists;

      List nonStandBusPurList =
        clientContext.Web.Lists.GetByTitle(
          "Non-Standard Business Purchase Requests");

      ListItem newListItem =
        nonStandBusPurList.AddItem(new ListItemCreationInformation());
        newListItem["Title"] = Title.Text;
        newListItem["RequestDescription"] = Description.Text;
        newListItem["Price"] = Price.Text;

        newListItem.Update();

        clientContext.Load(nonStandBusPurList, list => list.Title);

        clientContext.ExecuteQueryAsync(onQuerySucceeded, onQueryFailed);
    }

    private void onQuerySucceeded(
      object sender, ClientRequestSucceededEventArgs args)
    {
      Dispatcher.BeginInvoke(() =>
      {
        MessageBox.Show("New item added.");
      });
    }


    private void onQueryFailed(object sender,
      ClientRequestFailedEventArgs args)
    {
      Dispatcher.BeginInvoke(() =>
      {
        MessageBox.Show("Request failed. " + args.Message + "\n" +
          args.StackTrace);
      });
    }
  }
}
```

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
@infragistics

(we'll test by hosting the application in SharePoint) and select Silverlight 4 for the Silverlight Version.

To reference the Silverlight client OM API, I'll add references to Microsoft.SharePoint.Client.Silverlight.dll and Microsoft.SharePoint.Client.Silverlight.Runtime.dll in C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin. Next, I'll create the Silverlight UI by opening MainPage.xaml and replacing the XAML with the code in **Figure 8**.

The XAML in **Figure 8** defines text boxes and a button to collect information to add to my list as seen in **Figure 9**.

Double-click the button in the designer and replace the class with the code in **Figure 10**.

The code in **Figure 10** follows a common pattern in client OM code. First, I'll get access to the client context via the ClientContext class (which is equivalent to the SPContext class). Next, I'll access the site and list via the Web, ListCollection and List classes, respectively.

Figure 11 **Adding the SilverlightObjectTagControl.cs Helper Class**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace PurchasingMgrWP
{
  class SilverlightObjectTagControl : WebControl
  {
    public string Source { get; set; }
    public string InitParameters { get; set; }

    protected override void CreateChildControls()
    {
      base.CreateChildControls();

      if (Source != null && Source != "")
      {
        string width = (this.Width == Unit.Empty) ? "400" :
          this.Width.ToString();
        string height = (this.Height == Unit.Empty) ? "300" :
          this.Height.ToString();

        this.Controls.Add(new LiteralControl(
          "  <div>" +
          "  <object data=\"data:application/x-silverlight-2,\" +
          "  type=\"application/x-silverlight-2\" width=\"" + width +
          "  "\" height=\"" + height + "\">" +
          "  <param name=\"source\" value=\"" + Source + "\"/>" +
          "  <param name=\"onerror\" value=\"onSilverlightError\" />" +
          "  <param name=\"background\" value=\"white\" />" +
          "  <param name=\"minRuntimeVersion\" value=\"4.0.50826.0\" />" +
          "  <param name=\"autoUpgrade\" value=\"true\" />" +
          "  <param name=\"initparams\" value=\"" + InitParameters + "\" />" +
          "  <a href=\"http://go.microsoft.com/fwlink/?LinkID=" +
          "  149156&v=4.0.50826.0\" +
          "  style=\"text-decoration: none;\">" +
          "  <img src=\"http://go.microsoft.com/fwlink/?LinkId=161376\" +
          "  alt=\"Get Microsoft Silverlight\" style=\"border-style: none\"/>" +
          "  </a>" +
          "  </object>" +
          "  <iframe id=\"_sl_historyFrame\" +
          "  style='visibility:hidden;height:0;width:0;border:0px'></iframe>" +
          "  </div>"
          ));

      }
    }
  }
}
```

Note the similarity to the SPWeb, SPListCollection and SPList classes. Finally, I'll create a ListItem by calling the List.AddItem method, populate it with data from the UI and call the ListItem.Update method. The ListItem isn't actually created until the ClientContext.Load and ClientContext.ExecuteQueryAsync methods are called to execute the query. Note that you can save roundtrips to the server by loading multiple queries via ClientContext.Load, calling the ClientContext.ExecuteQueryAsync method.

To deploy the Silverlight 4 application, I'll add a module to deploy the application with my Web Part project. I'll select PurchasingMgrWP in the Solution Explorer, right-click and select Add | New Item |Module and name the module ClientBin. I'll replace the contents of Elements.xml created with this XML:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="ClientBin">
    <File Path="ClientBin\NonStandBusPurchaseReqsSLOM.xap"
      Url="ClientBin/NonStandBusPurchaseReqsSLOM.xap" />
  </Module>
</Elements>
```

This XML deploys the NonStandBusPurchaseReqsSLOM.xap file to the ClientBin folder in my SharePoint site.

To deploy the output of the NonStandBusPurchaseReqsSLOM project with the ClientBin module, I'll select the ClientBin module in Solution Explorer and open the Project Output References property dialog. I'll click Add and select NonStandBusPurchaseReqsSLOM as the Project Name and ElementFile as the DeploymentType.

Next, I'll add a custom Web Part to my SharePoint solution to host my Silverlight 4 application. I'll select PurchasingMgrWP in the Solution Explorer, right-click and select Add | New Item, select Web Part and give the Web Part the name NonStandBusPurchaseReqsWP. I'll use a custom Web Part in order to pass parameters to my Silverlight 4 application, such as the URL of the site used to create ClientContext. In order to do this, I'll add a helper class called SilverlightObjectTagControl.cs and replace the body of that class with the code in **Figure 11**.

Figure 12 **NonStandBusPurchaseReqsWP.cs**

```
using System;
using System.ComponentModel;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;

namespace PurchasingMgrWP.NonStandBusPurchaseReqsWP
{
  [ToolboxItemAttribute(false)]
  public class NonStandBusPurchaseReqsWP : WebPart
  {
    protected override void CreateChildControls()
    {
      base.CreateChildControls();

      SilverlightObjectTagControl slhc =
        new SilverlightObjectTagControl();
      slhc.Source = SPContext.Current.Site.Url +
        "/ClientBin/NonStandBusPurchaseReqsSLOM.xap";
      slhc.InitParameters = "url=" + SPContext.Current.Web.Url;

        this.Controls.Add(slhc);
    }
  }
}
```

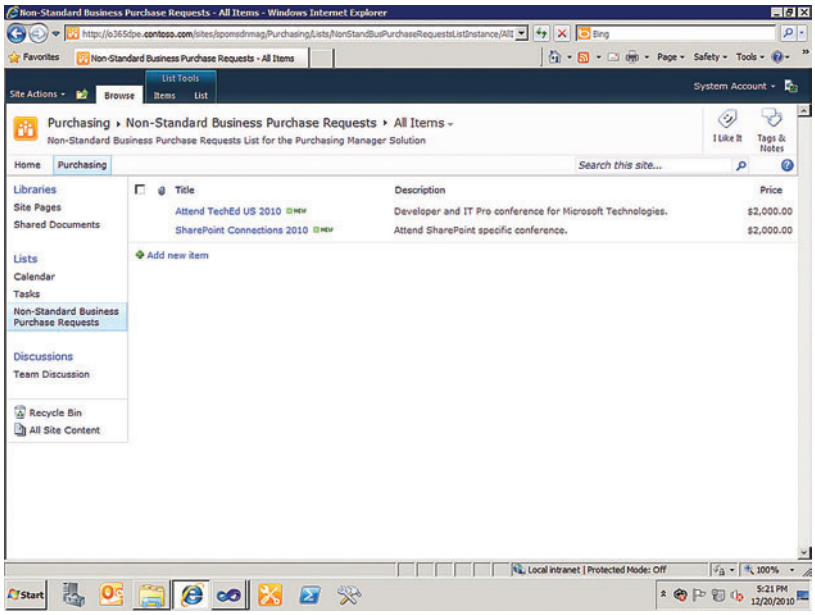


Figure 13 **The NonStandBusPurchaseReqsWP in Action**

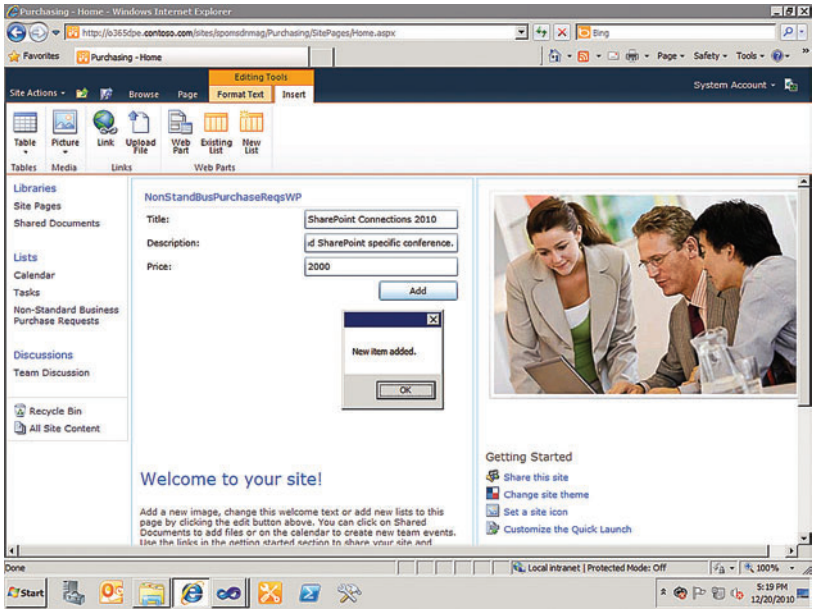


Figure 14 **The Updated Non-Standard Business Purchase Requests List**

The SilverlightObjectTagControl class in **Figure 11** has two properties: Source is used to pass the URL of the Silverlight application to load in the Web Part, and InitParameters to pass initialization parameters to the Silverlight 4 application. These properties are used to build the <object /> tag for the Silverlight application in the CreateChildControls method. To use this class, open NonStandBusPurchaseReqsWP.cs and replace the code in that class with the code in **Figure 12**.

The code in **Figure 12** creates an instance of SilverlightObjectTagControl, sets the Source property to the URL of the Silverlight application in ClientBin and sets the InitParameters property to hold the URL of the current site (where the Non-Standard Business Purchase Requests list can be found). To pass the URL to the constructor of the MainPage class in NonStandBusPurchaseReqsSLOM, open App.xaml.cs and add the following code to the Application_Startup event:

```
private void Application_Startup(object sender,
  StartupEventArgs e)
{
  string url = e.InitParams["url"];

  this.RootVisual = new MainPage(url);
}
```

To test the Web Part, deploy the PurchasingMgr.wsp package to the on-premises Purchasing site to deploy the Non-Standard Business Purchase Requests list (the list was removed when the debug session listed earlier was ended), and then debug the PurchasingMgrWP solution from Visual Studio 2010. When added to \Purchasing\Home.aspx, the Web Part allows me to add items directly to the list from Silverlight, as seen in **Figure 13** and **Figure 14**.

Developing and debugging against the on-premises site allows me to use Visual Studio 2010 to debug both the SharePoint and Silverlight 4 code until I have the solution tested completely. At that point, I'll upload the PurchasingMgrWP.wsp to the Solution Gallery in SharePoint Online.

The SharePoint client OM provides a familiar and consistent object-oriented API for accessing lists and libraries in SharePoint Online. The API is a subset of the Microsoft.SharePoint API and is scoped to the site collection and below, which is perfectly aligned to SharePoint Online development.

SharePoint Solutions in the Cloud

Summing up, SharePoint Online provides SharePoint developers a unique opportunity to build SharePoint solutions for the cloud using the skills and tools they already have. By understanding SharePoint Online customization features (including what's supported and what's not), sandboxed solutions, the SharePoint client OM and declarative workflows built using SharePoint Designer 2010, you can build SharePoint solutions that run in the cloud with SharePoint Online. To keep up on SharePoint Online development throughout the beta process, check out the SharePoint Online Developer Resource Center (msdn.com/sharepointonline). ■

CHRIS MAYO *is a technology specialist focusing on Office 365 and SharePoint Online. He has experience as both a writer and a public speaker delivering technical content to audiences ranging in size from small groups to thousands. He coauthored "Programming for Unified Communications with Microsoft Office Communications Server 2007 R2" (Microsoft Press, 2009). He's been with Microsoft for 10 years. Prior to joining Microsoft, he served as a developer and architect in the IT departments of Fortune 500 companies in the retail and finance industries. Keep up with Mayo at his blog, blogs.msdn.com/cmayo.*

THANKS to the following technical experts for reviewing this article:
George Durzi, Steve Fox, AJ May and Christina Storm

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
@infragistics

Processing Health Care Claims with BizTalk Server 2010

Mark Beckner

BizTalk Server 2010 introduces an entirely re-architected platform for managing the exchange of Electronic Data Interchange (EDI) documents between trading partners. If you've worked with earlier editions of the platform, you may have encountered some headaches with setting up document exchange between various parties. With the latest release, you have complete control over the organization and configuration of document settings related to any interchange between parties.

BizTalk Server 2010 also provides a much-improved experience when developing maps.

To illustrate these new features, I'm going to walk you through the steps required for building out an EDI solution. Every EDI solution developed in BizTalk Server 2010 follows a basic pattern:

- Adding and modifying base EDI schemas.
- Creating maps between the EDI document and internal/external messaging.
- Implementing orchestrations to handle workflow associated with processing the EDI documents.
- Configuring trading partner settings, including parties, business profiles, agreements and acknowledgments.
- Configuring ports to enable reception or delivery of documents, whether over FTP, AS2 or another protocol.

This article discusses:

- EDI schemas
- Developing EDI maps
- Trading partner configurations
- Ports and document delivery

Technologies discussed:

BizTalk Server 2010

For purposes of illustration, we'll look at the exchange of documents between a hospital and a claims-processing unit. The documents exchanged between parties will be Health Insurance Portability and Accountability Act (HIPAA)-compliant 837 Professional and Institutional files.

Working with Schemas

In most EDI solutions, you'll be working with two basic types of schema: the EDI schemas that represent the flat files exchanged with trading partners, and the internal schemas that represent document types needed for processing the data within the EDI document.

In my example, the solution exchanges the 837 documents with external parties, but uses a different document format for internal processing. The internal schema represents an ECSIF flat file: a common format for claims processors. The 837 schemas that ship with BizTalk and can be added to a Visual Studio project, but the internal schemas (such as an ECSIF) must be built by hand.

BizTalk Server 2010 ships with thousands of existing schemas that define a majority of the EDI documents in use today. To access the schemas, run the MicrosoftEdiXSDTemplates.exe executable found in the $\Program Files\Microsoft BizTalk Server 2010\XSD_Schema\EDI directory. For the purposes of this example, I'll use the 837 Professional and Institutional HIPAA-compliant schemas found in the HIPAA\00501 folder. Adding the XSD file to a Visual Studio project allows it to be referenced and used by other BizTalk components—most importantly, by maps.

Figure 1 shows the 837 Professional 5010 schema within Visual Studio 2010. Notice the number of nodes on this schema: the 837 is

one of the most complex EDI documents, and can be extremely tricky to work with. It contains hundreds of nodes that are virtually identical, representing subscriber and patient information.

Figure 2 shows the internal schema representing the ECSIF format. This schema was generated using the Flat File Schema Wizard. The wizard can be pointed to a valid flat file instance to create an XSD. A number of the fields in the FileHeader node have been promoted in this schema. Promoted fields allow for improved filtering and mapping options.

Once the schemas have been defined and added to the Visual Studio project, you can begin building out the maps. In this case, I'll look at several scenarios that are useful in mapping an 837 document.

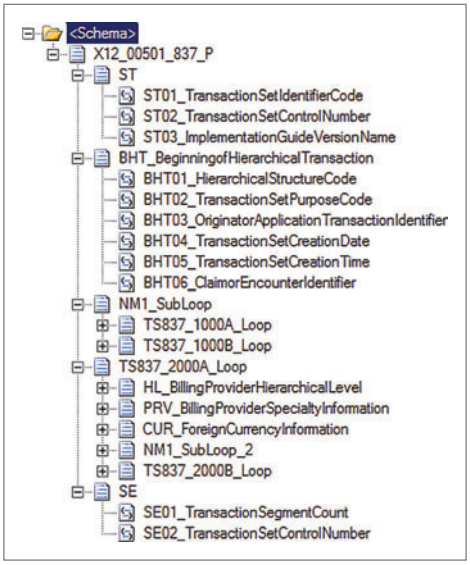


Figure 1 **The 837 Professional 5010 Schema**

Developing Maps

The mapping interface in BizTalk Server 2010 has been extensively revised and introduces a variety of new capabilities. These capabilities include zooming, automatic node matching and search capabilities. One of the most noticeable enhancements is the ability to click on a line or functoid and have all other mappings fade into the background.

As any EDI map developer will know, some maps get extremely complex, with multiple tabs and dozens (or even hundreds) of functoids. Finding out what data in the source schema maps to what data in the target schema and what functoids are being used to do this mapping can be difficult to visualize. By clicking on any of the lines or functoids used, all related mappings will be highlighted.

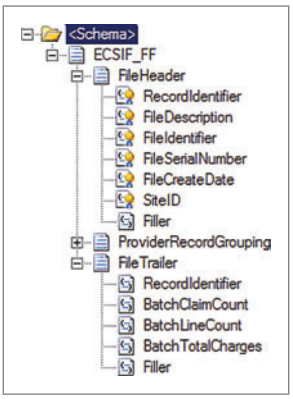


Figure 2 **The Target ECSIF Schema Format**

An example of a complex map with a specific set of mapping logic highlighted is shown in **Figure 3**. This brings up a point of BizTalk map development that's often overlooked: there are times to use the map interface, and there are times not to use it. Not all mapping that's done in BizTalk is best suited for standard mapping—sometimes using alternative approaches is in order. Alternatives include inline scripting and external components, including XSLT and Microsoft .NET Framework components.

BizTalk map development generally consists of a combination of standard functoids and inline XSLT and C#. There are even times when shelling out to an external XSLT stylesheet is in order (which bypasses the BizTalk mapper altogether). EDI maps can become complicated and require ingenuity and planning to end up with the needed solution.

To illustrate the use of inline C#, let's look at a common function of mapping an outbound 837 Professional or Institutional file: the mapping of hierarchical (HL) segments. The HL segments require incremental values for each record in the file, and denote parent/child relationships. There really is no traditional functoid combination that will allow these values to be set properly. There is, however, a simple inline C# approach that allows for the correct values. This approach requires two scripting functoids: one that stores a global variable and maps HL01, while the second maps HL02 (which is dependent on the value of HL01).

The HL01 functoid script looks like this:

```
int intHL01;
public int getHL01() {
intHL01++;
return intHL01;
}
```

Here's the code for the HL02 script:

```
public int getHL02() {
return intHL01 -1;
}
```

Figure 4 shows the functoids placed in the map.

Another situation that frequently arises in mapping EDI documents is the need to use inline XSLT. This is one of the most important skills to incorporate into BizTalk mapping, and is something you should acquaint yourself with. It allows for many options around looping and node creation that simply aren't available using standard functoid combinations.

One illustration of using Inline XSLT in a map is shown in **Figure 5**. This code demonstrates how to use the Inline XSLT Call Template functionality to pass in a parameter from the source document (Name) and create a node in the target 837 document.

When developing a BizTalk map, always think about the long-term maintainability of the map. Is this something that you'll be able to easily update? Is this something that someone else can work with in the future? Good coding practice should not be forgotten when working with BizTalk maps, and some amount of architecture

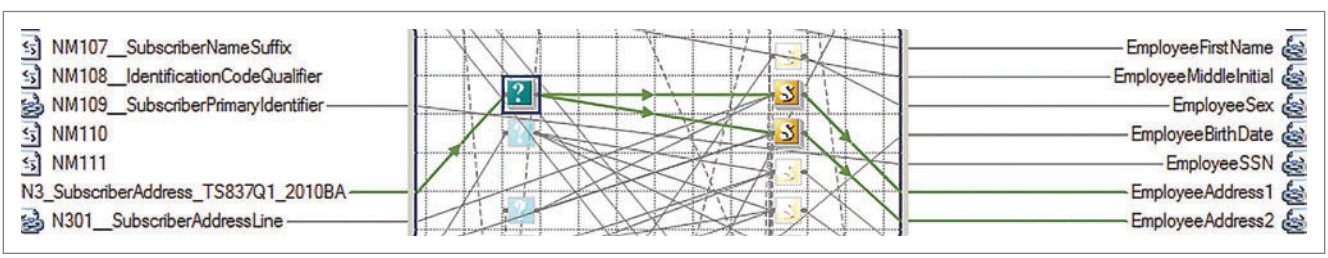


Figure 3 **Highlight Capabilities in BizTalk Server 2010 Maps**

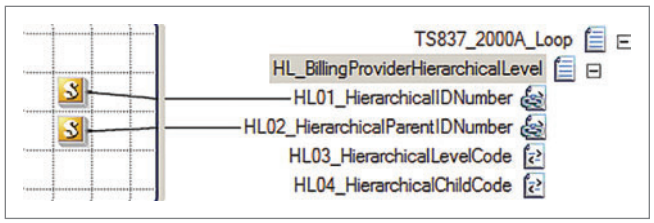


Figure 4 **Mapping HL Segment on Outbound 837**

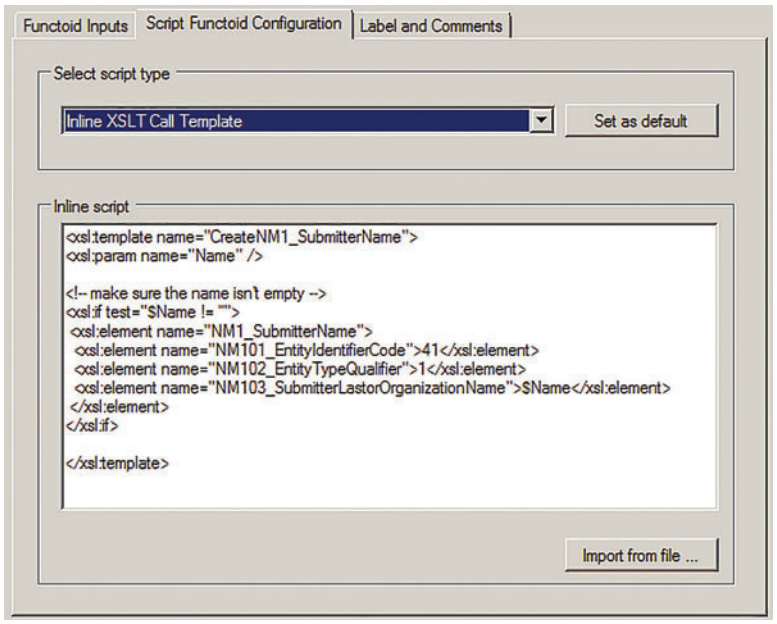


Figure 5 **Passing Parameters to an Inline XSLT Call Template Script**

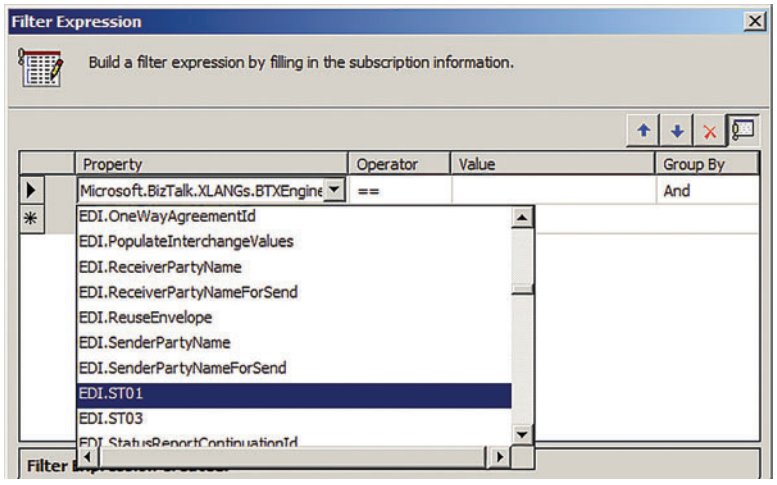


Figure 6 **Setting a Filter on an Orchestration**

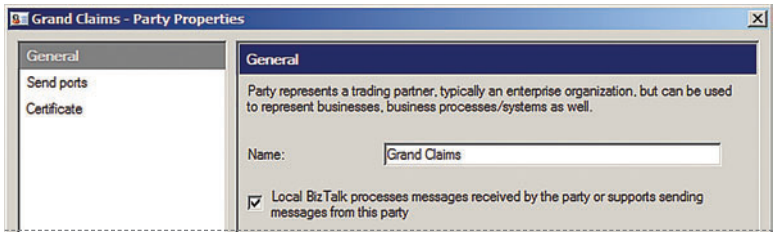


Figure 7 **Top-Level Party Configuration**

and planning should be involved when developing maps that have a lot of logic or complex functionality within them.

Orchestrations

The use of orchestrations in EDI solutions is not a requirement. Often, documents simply need to be mapped from one format to another and delivered, without the need for the inclusion of workflow.

In some cases, however, a document may need to have several steps of processing done to it before it's ready to be delivered. To illustrate this, I'll set up an orchestration to map and archive messages to a table in SQL Server. The orchestration will be configured with a filter to ensure only documents of a certain type are processed by it.

> In most EDI solutions, you'll be working with two basic types of schema.

The orchestration can be set up to subscribe to just about any of the fields within the ISA or ST segments of an EDI document (among many other properties). To configure an orchestration to subscribe to a specific field on a document, a filter can be set on the initial receive shape of the orchestration, as shown in **Figure 6**.

With the filter specified, the orchestration can now do the necessary processing of the EDI document. In this case, the orchestration is going to map the data from the 837 format to the ECSIF format and then write this information to an archive table in SQL Server. The mapping of the documents is done through a transform shape and the inclusion of a map file, but the writing of the information to SQL Server has a number of options available for it.

When thinking of SQL Server, many BizTalk developers assume that they need to use one of the adapters available for writing to SQL. The truth is that, in most cases, the SQL adapters are overly complex for basic database calls. Generally, the easiest and most supportable approach for interacting with SQL Server is through the use of a custom .NET assembly class. When developing classes that will be called from orchestration, always make sure and mark the class as serializable to ensure that BizTalk can call it from any type of transaction state:

```
namespace Demo.BizTalk.Helper {
  [Serializable]
  public class DataAccess
  { }
}
```

Developing orchestrations for EDI solutions is no different than for other types of BizTalk implementations. The main thing to remember when developing orchestrations is to keep it simple. There's an art to BizTalk development and to organizing your BizTalk

projects properly. If you've planned and developed properly, you'll end up with a solution that's easy to make updates to and deploy to a production environment.

Trading Partner Configurations

BizTalk Server 2010 introduces a new interface for managing the exchange of EDI documents between trading partners. It's made up of three basic tiers of organization: the party, the business profile and the agreement.

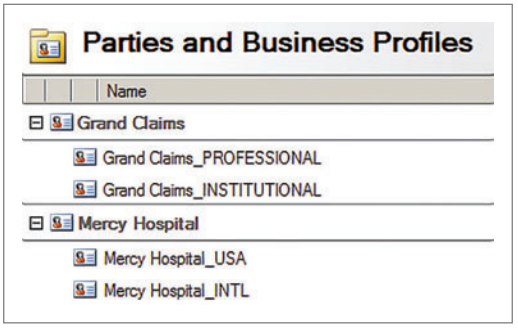


Figure 8 **Business Profiles Associated with Parties**

The party represents the top-level organization of the trading partner. Configurations on this artifact relate to things that are common across all documents being exchanged with this trading partner. For example, certificates that may be required for secure communication would be configured at this level.

> BizTalk Server 2010 introduces a new interface for managing the exchange of EDI documents.

In my example, I have two parties: the claims processor and the hospital. Each is set up as its own unique BizTalk party. All that needs to be configured is the name, as shown in **Figure 7**.

A party will have one or more business profiles associated with it. The business profile represents a department within an organization that has its own unique business identity when sending or receiving EDI documents. A business identity is the value that appears in the ISA06 or ISA08 segment of an EDI document (depending on whether the document is being sent or received) and uniquely identifies the trading partner from all other entities. Many organizations will have a single business profile, but some will require multiple profiles.

In my example, the claims processor has two business profiles: one that represents professional claims processing, and another that represents institutional claims processing. The hospital also has two business profiles: the national branch and the international branch. **Figure 8** shows the parties with their business profiles.

Because the business profile represents a unique business identity within a party, configurations at this level deal with information that's common across all documents that will be exchanged with this identity. All inbound and outbound settings that are common across all document types being exchanged will be configured: the protocols used (X12, EDIFACT, AS2), the validation settings, the transaction sets (EDI documents are allowed at this level), acknowledgments and some of the envelope settings are all configured on a business profile. In many cases, the default information can be used at this level, because the configuration of the specific agreements of a business profile will define this information and more.

A business profile can have one or more agreements. An agreement represents the way that two parties are expecting to exchange one or more document types with one another. This is where the specifics about the envelope, the acknowledgments and the transaction sets allowed are defined. One agreement could allow for certain documents to be exchanged with 997 acknowledgments configured, while another agreement could allow for other document types to forego the acknowledgments.

In my example, the claims processor and the hospital are exchanging documents. The hospital will send the claim (X12 837 Institutional version 5010) to the claims processor, and the claims processor will send an acknowledgment (X12 997) back to the hospital. **Figure 9** shows the envelope identifier configuration and **Figure 10** shows the transaction set configuration on the agreement for documents flowing from the hospital to the claims processor. Note the tabs at the top of the window indicating the flow of the document.

Figure 11 shows the configuration of the acknowledgments that are sent back from the claims processor to the hospital.

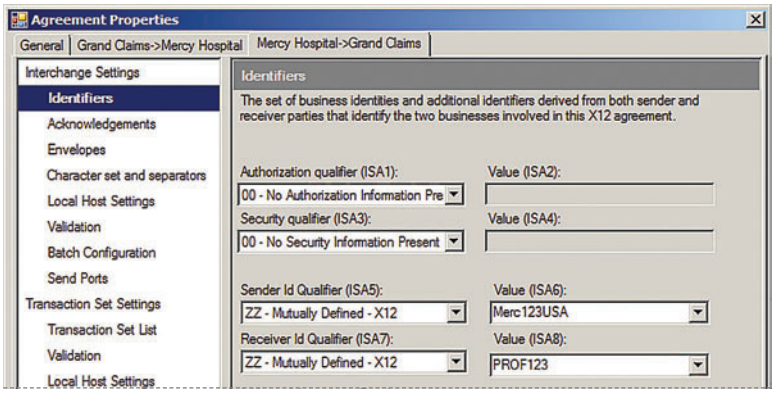


Figure 9 **Configuring ISA Envelope Settings on an Agreement**

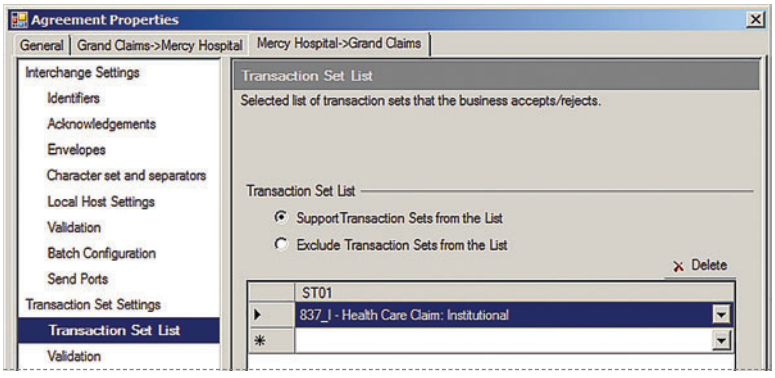


Figure 10 **Configuring Transaction Sets on an Agreement**

THESE GUYS STAND ON THEIR OWN.

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

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

If you're exchanging documents with more than a single trading partner, you'll likely find that the majority of your configurations are virtually identical—the only thing changing will be the identifiers in the ISA segment of the envelope.

To ease development, make sure to use the template functionality available from the agreement configuration screen. There are two buttons you'll be interested in: Save As Template and Load From Template. When you have a trading partner fully configured, and the EDI documents are flowing end-to-end with the correct envelope settings and acknowledgments, simply save the agreement settings as a template and use them as the starting point for future agreements.

Port Configurations and Document Delivery

The actual delivery of documents to or from BizTalk to external trading partners is done through the configuration of ports. The port defines the type of delivery mechanism (FTP, file and so on) and contains the BizTalk pipeline that transforms the XML document within BizTalk into the flat file EDI document expected by trading partners. The pipeline contains the logic to interpret (or create) the EDI envelope on a document and to determine which party the document resolves to.

> Configuring the send port to deliver documents to a specific location is straightforward.

To understand how ports process EDI documents, let's look at sending an acknowledgment. As you saw earlier, acknowledgments are configured on the agreements of a BizTalk party. When a document arrives, BizTalk can automatically generate a 997 acknowledgment. What happens is that BizTalk creates the 997 XML and drops it on the BizTalk message box, but it doesn't actually route the message anywhere. A send port must be set up and configured to convert the XML to a flat file, add the envelope and deliver it to the appropriate destination.

Setting up the send port to deliver an acknowledgment requires three basic steps:

1. The definition of the send port and delivery protocol
2. The inclusion of the EdiSend pipeline (or custom pipeline with EDI pipeline components)
3. The configuration of filters to listen for appropriate acknowledgments

Configuring the send port to deliver documents to a specific location is straightforward. **Figure 12** shows a Send Port configured with the EdiSend pipeline. The send port will write files out to a file directory with the full EDI envelope and formatting in place.

Setting the filter on the send port is also easy to do. It simply requires specifying that it should only pick up acknowledgments generated by the system for this trading partner (as opposed to incoming 997s from the partner). **Figure 13** shows a fully configured set of filters.

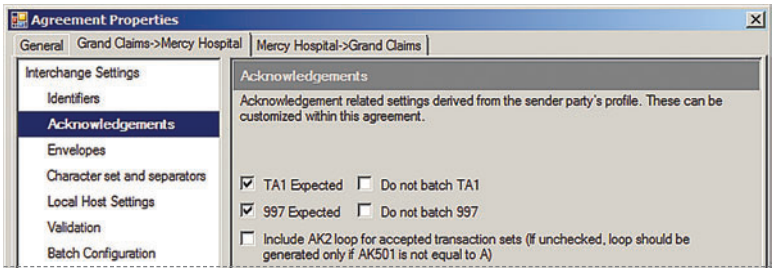


Figure 11 **Configuring Acknowledgements on an Agreement**

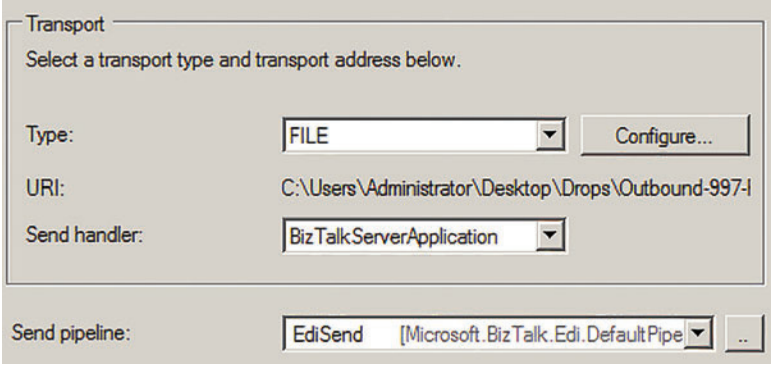


Figure 12 **Setting Pipeline and Delivery Information on a Port**

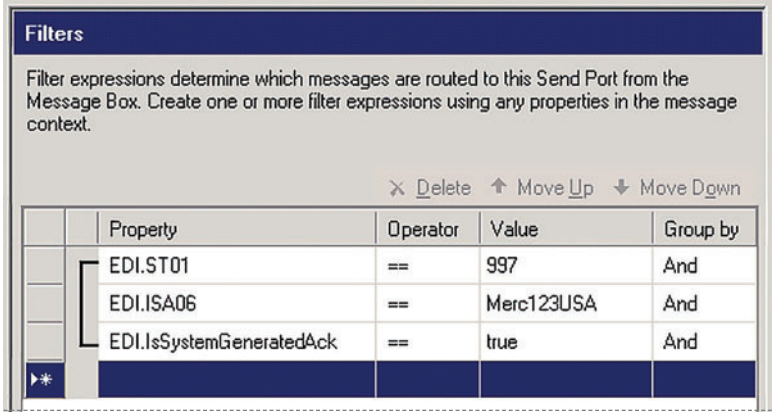
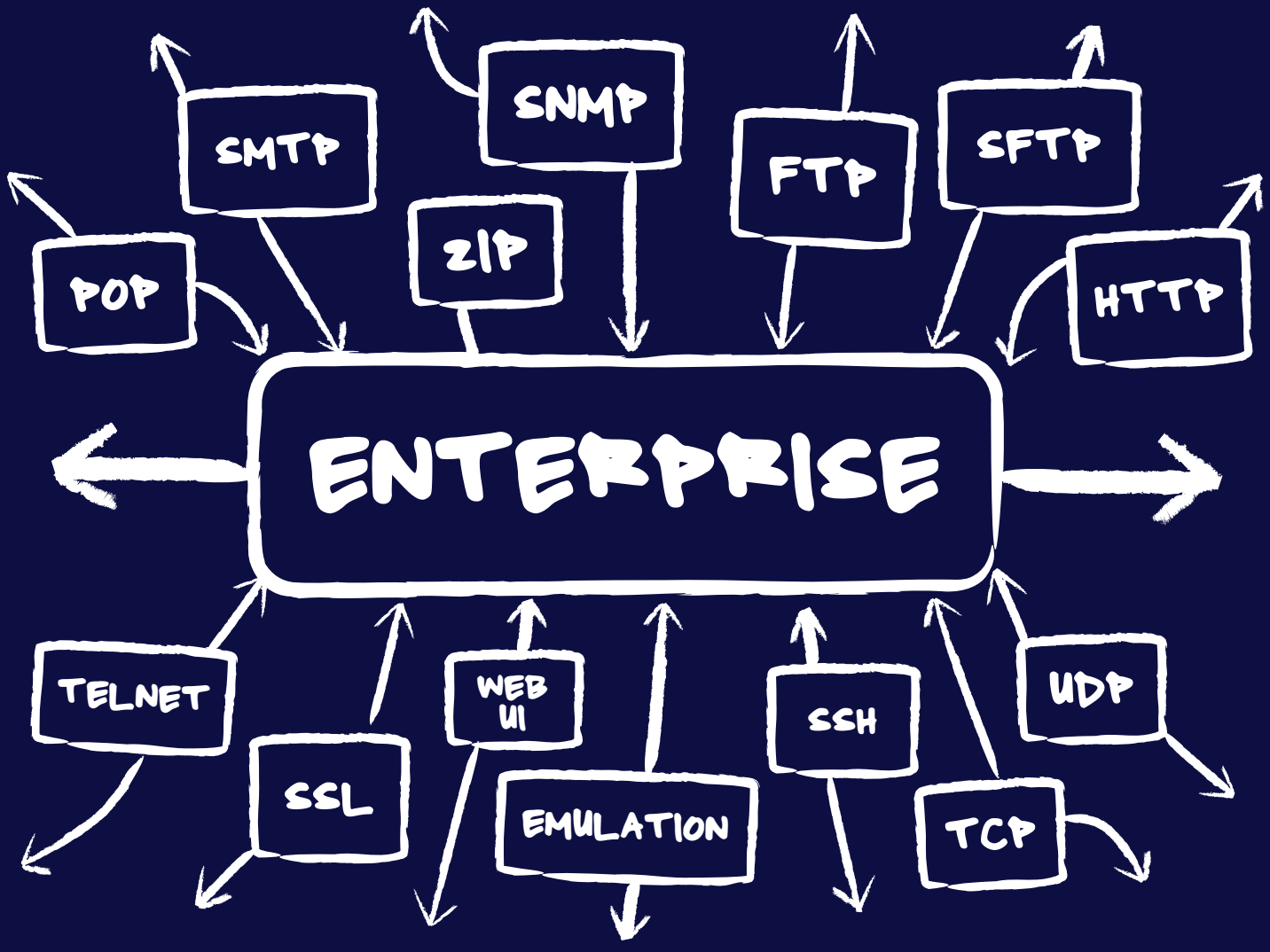


Figure 13 **Filtering on a Send Port**

Problem Solved

The new EDI functionality in BizTalk Server 2010 is concentrated primarily in the management of trading partners. Hierarchical relationships and organizations that were impossible with previous editions of the platform can now be modeled with relative ease. In addition to this, the map interface has been enhanced, and the developer experience with mapping is much improved. With the various improvements and increased functionality, any EDI solution can be modeled and developed successfully using BizTalk Server 2010. ■

Mark Beckner *is the founder of Inotek Consulting Group LLC. He works across the Microsoft stack, including BizTalk, SharePoint, Dynamics CRM and general .NET development. He can be reached at mbeckner@inotekgroup.com.*



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	**FTP**	**SMTP**	**DNS**	**Telnet**
UDP	**SFTP**	**IMAP**	**Rlogin**	**VT Emulation**
TCP	**HTTP**	**S/MIME**	**Rsh**	**ZIP Compression**
SSL	**POP**	**Ping**	**Rexec**	*more...*

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

Download a fully functional product trial today!

DART.com

Tips and Tricks for Loading Silverlight Locale Resources

Matthew Delisle

Silverlight is a great framework for creating rich Internet applications (RIAs), but it doesn't yet provide the robust support for localization that you enjoy in other components of the Microsoft .NET Framework. Silverlight does have .resx files, a simple ResourceManager class and an element in the project file. But after that you're on your own. There are no custom markup extensions, and no support for the DynamicResource class.

In this article I'll show you how to remedy all of these issues. I'll present a solution that will allow a developer to load resource sets at run time, use any format for storing resources, change resources without recompiling and demonstrate lazy loading of resources.

This article is divided into three parts. First, I'll develop a simple application using the localization process detailed by Microsoft. Next, I'll present another localization solution that has some benefits over the standard process. Finally, I'll round off the solution with a discussion of the back-end components needed to complete the solution.

This article discusses:

- Locale resource basics
- A smart resource manager
- Using a WCF service
- Loading the neutral locale

Technologies discussed:

Silverlight

Code download available at:

code.msdn.microsoft.com/mag201103Localization

The Standard Localization Process

I'll start by building a Silverlight application that uses the localization process outlined by Microsoft. A detailed description of the process is available at msdn.microsoft.com/library/cc838238(VS.95).

The UI contains a TextBlock and an Image, as shown in **Figure 1**.

The localization process described by Microsoft uses .resx files to store the resource data. The .resx files are embedded in the main assembly or a satellite assembly and loaded only once, at application startup. You can build applications targeted at certain languages by modifying the SupportedCultures element in the project file. This sample application will be localized for two languages, English and French. After adding the two resource files and two images representing flags, the project structure looks like **Figure 2**.

I changed the build action for the images to content so I can reference the images using a less verbose syntax. I'll add two entries to each file. The TextBlock is referenced via a property called Welcome, and the Image control is referenced via a property called FlagImage.

When resource files are created in a Silverlight app, the default modifier for the generated resource class is internal. Unfortunately, XAML can't read internal members, even if they're located in the same assembly. To remedy this situation, the generated class modifiers need to be changed to public. This can be accomplished

in the design view of the resource file. The Access Modifier dropdown menu lets you designate the scope of the generated class.

Once resource files are ready, you need to bind the resources in XAML. To do this you create a wrapper class with a static field referencing an instance of the resource class. The class is as simple as this:

```
public class StringResources {
  private static readonly strings strings = new strings();
  public strings Strings {  get { return strings; } }
}
```

Figure 1 **The App**

To make the class accessible from XAML, you need to create an instance. In this case, I'll create the instance in the App class so that it's accessible throughout the project:

```
<Application.Resources>
  <local:StringResources x:Key="LocalizedStrings"/>
</Application.Resources>
```

Data-binding in XAML is now possible. The XAML for the TextBlock and the Image looks like this:

```
<StackPanel Grid.ColumnSpan="2" Orientation="Horizontal"
  HorizontalAlignment="Center">
  <TextBlock Text="{Binding Strings.Welcome, Source={StaticResource
LocalizedStrings}}"
    FontSize="24"/>
</StackPanel>
<Image Grid.Row="1" Grid.ColumnSpan="2"
  HorizontalAlignment="Center"
  Source="{Binding Strings.FlagImage, Source={StaticResource
LocalizedStrings}}"/>
```

The path is the String property followed by the key of the resource entry. The source is the instance of the StringResources wrapper from the App class.

Setting the Culture

There are three application settings that must be configured for the application to pick up the browser's culture setting and display the appropriate culture.

The first setting is the SupportedCultures element in the .csproj file. Currently there's no dialog box in Visual Studio to edit the setting, so the project file must be edited manually. You can edit a project file either by opening it outside of Visual Studio, or by unloading the project and selecting edit from the context menu within Visual Studio.

To enable English and French for this application, the value of the SupportedCultures element looks like this:

```
<SupportedCultures>fr</SupportedCultures>
```

The cultures values are separated by commas. You don't need to specify the neutral culture; it's compiled into the main DLL.

These next steps are necessary to pick up the browser language setting. A parameter must be added to the embedded Silverlight object in the Web page. The parameters value is the current UI culture, taken from the server side. This requires the Web page to be an .aspx file. The parameter syntax is:

```
<param name="uiculture"
  value="<%=Thread.CurrentThread.
CurrentCulture.Name %>" />
```

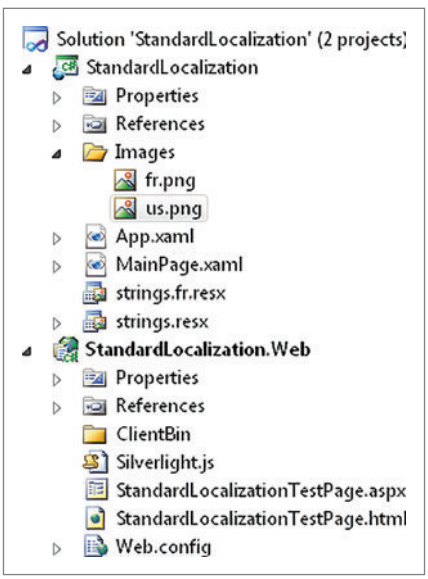


Figure 2 **Project Structure After Adding .resx Files**

The final mandatory step in this process is to edit the web.config file and add a globalization element inside of the system.web element, setting the values of its attributes to auto:

```
<globalization culture="auto" uiCulture="auto"/>
```

As mentioned earlier, a Silverlight application has a neutral language setting. The setting is reached by going to the Silverlight tab of the project properties and clicking Assembly Information. The neutral language property is located at the bottom of the dialog, as shown in **Figure 3**.

I recommend setting the neutral language to one without a region. This setting is a fallback, and it's more useful if it covers a wide range of potential locales. Setting a neutral language adds an assembly attribute to the assemblyinfo.cs file, like this:

```
[assembly: NeutralResourcesLanguageAttribute("en")]
```

After all that, what you end up with is a localized application that reads the browser language setting at startup and loads the appropriate resources.

A Custom Localization Process

The limitations of the standard localization process stem from the use of ResourceManager and .resx files. The ResourceManager class doesn't change resource sets at run time based on culture changes within the environment. Using .resx files locks the developer into one resource set per language and inflexibility in maintaining the resources.

In response to these limitations, let's look at an alternative solution that employs dynamic resources.

To make resources dynamic, the resource manager needs to send notification when the active resource set changes. To send notifications in Silverlight, you implement the INotifyPropertyChanged interface. Internally, each resource set will be a dictionary with a key and value type of string.

The Prism framework and the Managed Extensibility Framework (MEF) are popular for Silverlight development, and these frameworks break up the application into multiple .xap files. For localization, each .xap file needs its own instance of the resource manager. To send notifications to every .xap file (every instance of the resource manager), I need to keep track of each instance that gets created and iterate through that list when notifications need to be sent. **Figure 4** shows the code for this SmartResourceManager functionality.

As you can see, a static list is created to hold all instances of the resource manager. The active resource set is stored in the field resourceSet and every resource that has been loaded is stored in the ResourceSets list. In the constructor, the current instance is stored in the Instances list. The class implements INotifyPropertyChanged in the standard way. When the active resource set is changed, I iterate through the list of instances and fire each one's PropertyChanged event.

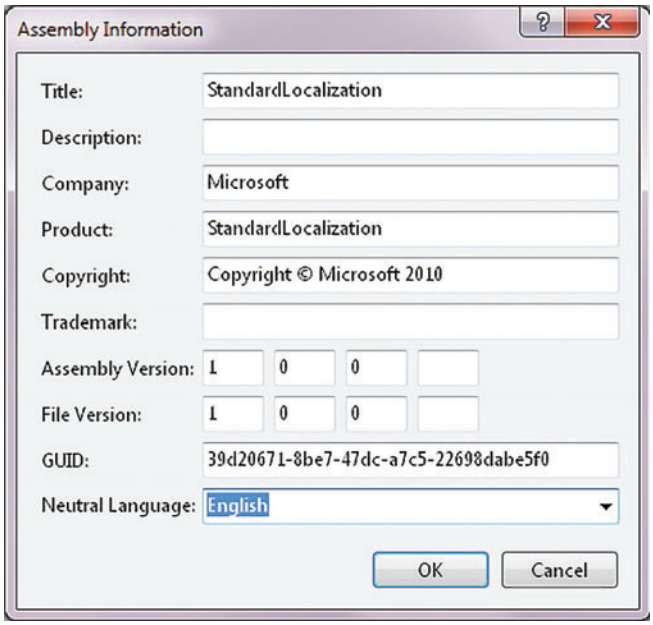


Figure 3 **Setting the Neutral Language**

The SmartResourceManager class needs a way to change the culture at run time, and it's as simple as a method that takes a CultureInfo object:

```
public void ChangeCulture(CultureInfo culture) {
  if (!ResourceSets.ContainsKey(culture.Name)) {
    // Load the resource set
  }
  else {
    ResourceSet = ResourceSets[culture.Name];
    Thread.CurrentThread.CurrentCulture =
      Thread.CurrentThread.CurrentUICulture =
      culture;
  }
}
```

This method checks to see if the requested culture has been loaded yet. If not, it loads the culture and then sets it as active. If the culture has already been loaded, the method simply sets the corresponding resource set as active. The code to load a resource is omitted for the moment.

For completeness, I'll also show you the two methods to load a resource programmatically (see **Figure 5**). The first method takes just a resource key and returns the resource from the active culture. The second method takes a resource and a culture name and returns the resource for that specific culture.

If you ran the application right now, all localized strings would be empty, because no resource sets have been downloaded. To load the initial resource set, I'm going to create a method named Initialize that takes the neutral language file and culture identifier. This method should be called only once during the application lifetime (see **Figure 6**).

Figure 4 **SmartResourceManager**

```
public class SmartResourceManager : INotifyPropertyChanged {
  private static readonly List<SmartResourceManager> Instances =
    new List<SmartResourceManager>();
  private static Dictionary<string, string> resourceSet;
  private static readonly Dictionary<string,
    Dictionary<string, string>> ResourceSets =
    new Dictionary<string, Dictionary<string, string>>();

  public Dictionary<string, string> ResourceSet {
    get { return resourceSet; }
    set { resourceSet = value;
    // Notify all instances
    foreach (var obj in Instances) {
      obj.NotifyPropertyChanged("ResourceSet");
    }
  }
}

public SmartResourceManager() {
  Instances.Add(this);
}

public event PropertyChangedEventHandler PropertyChanged;
public void NotifyPropertyChanged(string property) {
  var evt = PropertyChanged;

  if (evt != null) {
    evt(this, new PropertyChangedEventArgs(property));
  }
}
```

Binding to XAML

A custom markup extension would provide the most fluid binding syntax for the localized resources. Unfortunately, custom markup extensions aren't available in Silverlight. Binding to a dictionary is available in Silverlight 3 and later, and the syntax looks like this:

```
<StackPanel Grid.ColumnSpan="2" Orientation="Horizontal"
  HorizontalAlignment="Center">
  <TextBlock Text="{Binding Path=ResourceSet[Welcome],
Source={StaticResource
SmartRM}}" FontSize="24"/>
</StackPanel>
<Image Grid.Row="1" Grid.ColumnSpan="2"
  HorizontalAlignment="Center"
  Source="{Binding ResourceSet[FlagImage], Source={StaticResource SmartRM}}"/>
```

The path contains the name of the dictionary property with the key in square brackets. If you're using Silverlight 2, there are two options available: creating a ValueConverter class or emitting a strongly typed object using reflection. Creating a strongly typed object using reflection is beyond the scope of this article. The code for a ValueConverter would look like **Figure 7**.

The LocalizeConverter class takes the dictionary and parameter passed in and returns the value of that key in the dictionary. After creating an instance of the converter, the binding syntax would look like this:

```
<StackPanel Grid.ColumnSpan="2" Orientation="Horizontal"
  HorizontalAlignment="Center">
  <TextBlock Text="{Binding Path=ResourceSet, Source={StaticResource
SmartRM}, Converter={StaticResource LocalizeConverter},
ConverterParameter=Welcome}" FontSize="24"/>
</StackPanel>
<Image Grid.Row="1" Grid.ColumnSpan="2" HorizontalAlignment="Center"
  Source="{Binding ResourceSet, Source={StaticResource SmartRM},
Converter={StaticResource LocalizeConverter}, ConverterParameter=FlagImage}"/>
```

The syntax is more verbose using the converter, though you have more flexibility. However, for the rest of the article, I'll continue without the converter, instead using the dictionary binding syntax.

Locale Settings Redux

There are two application settings that must be configured for the culture to be picked up by the Silverlight application. These are the same settings discussed with the standard localization process. The globalization element in the web.config file needs to have culture and uiCulture values of auto:

```
<globalization culture="auto" uiCulture="auto"></globalization>
```

Also, the Silverlight object in the .aspx file needs to be passed in the thread current UI culture value as a parameter:

Figure 5 **Loading Resources**

```
public string GetString(string key) {
  if (string.IsNullOrEmpty(key)) return string.Empty;

  if (resourceSet.ContainsKey(key)) {
    return resourceSet[key];
  }
  else {
    return string.Empty;
  }
}

public string GetString(string key, string culture) {
  if (ResourceSets.ContainsKey(culture)) {
    if (ResourceSets[culture].ContainsKey(key)) {
      return ResourceSets[culture][key];
    }
    else {
      return string.Empty;
    }
  }
  else {
    return string.Empty;
  }
}
```

```
<param name="uiculture"
  value="<%=Thread.CurrentThread.CurrentCulture.Name %>" />
```

To showcase the dynamic localization of the application, I'm going to add a couple buttons that facilitate changing culture, as shown in **Figure 8**. The click event for the English button looks like this:

```
(App.Current.Resources["SmartRM"] as SmartResourceManager).ChangeCulture(
  new CultureInfo("en"));
```

With some mocked-up data, the application would run and display the appropriate language. The solution here allows for dynamic localization at run time and is extensible enough to load the resources using custom logic.

The next section focuses on filling in the remaining gap: where the resources are stored and how they're retrieved.

Server-Side Components

Now let's create a database to store resources and a Windows Communication Foundation (WCF) service to retrieve those resources. In larger applications, you'd want to create data and business layers, but for this example, I won't be using any abstractions.

Figure 6 **Initializing the Neutral Language**

```
public SmartResourceManager() {
  if (Instances.Count == 0) {
    ChangeCulture(Thread.CurrentThread.CurrentUICulture);
  }
  Instances.Add(this);
}

public void Initialize(string neutralLanguageFile,
  string neutralLanguage) {
  lock (lockObject) {
    if (isInitialized) return;
    isInitialized = true;
  }

  if (string.IsNullOrWhiteSpace(neutralLanguageFile)) {
    // No neutral resources
    ChangeCulture(Thread.CurrentThread.CurrentUICulture);
  }
  else {
    LoadNeutralResources(neutralLanguageFile, neutralLanguage);
  }
}
```

The reason I chose a WCF service is for the ease of creation and robustness offered by WCF. The reason I chose to store the resources in a relational database is for ease of maintenance and administration. An administration application could be created that would allow translators to easily modify the resources.

I'm using SQL Server 2008 Express for this application. The data schema is shown in **Figure 9**.

A Tag is a named group of resources. A StringResource is the entity representing a resource. The LocaleId column represents the name of the culture that the resource is under. The Comment column is added for compatibility with the .resx format. The CreatedDate and ModifiedDate columns are added for auditing purposes.

A StringResource can be associated with multiple Tags. The advantage of this is that you can create specific groups (for example, the resources for a single screen) and download only those resources. The disadvantage is that you can assign multiple resources with the same LocaleId, Key and Tag. In that case, you may want to write a trigger to manage creating or updating resources or use the ModifiedDate column when retrieving resource sets to determine which is the latest resource.

I'm going to retrieve the data using LINQ to SQL. The first service operation will take in a culture name and return all resources associated with that culture. Here's the interface:

```
[ServiceContract]
public interface ILocaleService {
  [OperationContract]
  Dictionary<string, string> GetResources(string cultureName);
}
```

Here's the implementation:

```
public class LocaleService : ILocaleService {
  private acmeDataContext dataContext = new acmeDataContext();

  public Dictionary<string, string> GetResources(string cultureName) {
  return (from r in dataContext.StringResources
          where r.LocaleId == cultureName
          select r).ToDictionary(x => x.Key, x => x.Value);
  }
}
```

The operation simply finds all resources whose LocaleId is equal to the cultureName parameter. The dataContext field is an instance of the LINQ to SQL class hooked up to the SQL Server database. That's it! LINQ and WCF make things so simple.

Now, it's time to link the WCF service to the SmartResourceManager class. After adding a service reference to the Silverlight application, I register to receive the completed event for the GetResources operation in the constructor:

Figure 7 **Custom ValueConverter**

```
public class LocalizeConverter : IValueConverter {
  public object Convert(object value,
    Type targetType, object parameter,
    System.Globalization.CultureInfo culture) {
    if (value == null) return string.Empty;

    Dictionary<string, string> resources =
      value as Dictionary<string, string>;
    if (resources == null) return string.Empty;
    string param = parameter.ToString();

    if (!resources.ContainsKey(param)) return string.Empty;

    return resources[param];
  }
}
```

```
public SmartResourceManager() {
  Instances.Add(this);
  localeClient.GetResourcesCompleted +=
    localeClient_GetResourcesCompleted;
  if (Instances.Count == 0) {
    ChangeCulture(Thread.CurrentThread.CurrentUICulture);
  }
}
```

The callback method should add the resource set to the list of resource sets and make the resource set the active set. The code is shown in **Figure 10**.

The ChangeCulture method needs to be modified to make a call to the WCF operation:

```
public void ChangeCulture(CultureInfo culture) {
  if (!ResourceSets.ContainsKey(culture.Name)) {
    localeClient.GetResourceSetsAsync(culture.Name, culture);
  }
  else {
    ResourceSet = ResourceSets[culture.Name];
    Thread.CurrentThread.CurrentCulture =
      Thread.CurrentThread.CurrentUICulture = culture;
  }
}
```

Loading the Neutral Locale

This application needs a way to recover if the Web services can't be contacted or are timing out. A resource file containing the neutral language resources should be stored outside of the Web services and loaded at startup. This will serve as a fallback and a performance enhancement over the service call.

I'm going to create another SmartResourceManager constructor with two parameters: a URL pointing to the neutral language resources file and a culture code identifying the culture of the resource file (see **Figure 11**).

If there's no neutral resource file, the normal process of calling the WCF call is performed. The LoadNeutralResources method uses a WebClient to retrieve the resource file from the server. It then parses the file and converts the XML string into a Dictionary object. I won't show the code here as it's a bit long and somewhat trivial, but you can find it in the code download for this article if you're interested.

Figure 8 **Culture-Change Buttons**

To call the parameterized SmartResourceManager constructor, I need to move the instantiation of the SmartResourceManager into the code-behind of the App class (because Silverlight doesn't support XAML 2009). I don't want to hardcode the resource file or the culture code, though, so I'll have to create a custom ConfigurationManager class, which you can check out in the code download.

After integrating the ConfigurationManager into the App class, the Startup event callback method looks like this:

```
private void Application_Startup(object sender, StartupEventArgs e) {
  ConfigurationManager.Error += ConfigurationManager_Error;
  ConfigurationManager.Loaded += ConfigurationManager_Loaded;
  ConfigurationManager.LoadSettings();
}
```

The startup callback method now serves to load the application settings and register for callbacks. If you do choose to make the loading of the configuration settings a background call, be careful of the race conditions that you can run into. Here are the callback methods for the ConfigurationManager events:

```
private void ConfigurationManager_Error(object sender, EventArgs e) {
  Resources.Add("SmartRM", new SmartResourceManager());
  this.RootVisual = new MainPage();
}

private void ConfigurationManager_Loaded(object sender, EventArgs e) {
  Resources.Add("SmartRM", new SmartResourceManager(
    ConfigurationManager.GetSetting("neutralLanguageFile"),
    ConfigurationManager.GetSetting("neutralLanguage")));
  this.RootVisual = new MainPage();
}
```

The Error event callback method loads SmartResourceManager without a neutral language, and the Loaded event callback method loads with a neutral language.

I need to put the resource file in a location where I don't have to recompile anything if I change it. I'm going to put it in the ClientBin directory of the Web project, and after creating the resource file, I'm going to change its extension to .xml so that it's publicly accessible and the WebClient class can access it from the

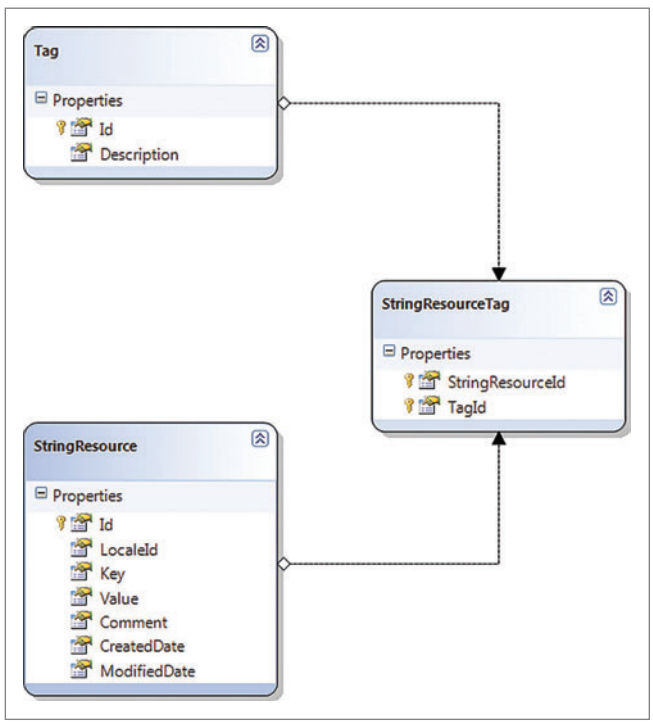


Figure 9 **Schema for Localization Tables in SQL Server 2008 Express**

Figure 10 **Adding Resources**

```
private void localeClient_GetResourcesCompleted(object sender,
  LocaleService.GetResourcesCompletedEventArgs e) {
  if (e.Error != null) {
    var evt = CultureChangeError;

    if (evt != null)
      evt(this, new CultureChangeErrorEventArgs(
        e.UserState as CultureInfo, e.Error));
  }
  else {
    if (e.Result == null) return;

    CultureInfo culture = e.UserState as CultureInfo;

    if (culture == null) return;

    ResourceSets.Add(culture.Name, e.Result);
    ResourceSet = e.Result;
    Thread.CurrentThread.CurrentCulture =
      Thread.CurrentThread.CurrentUICulture = culture;
  }
}
```

Figure 11 **Loading the Neutral Locale**

```
public SmartResourceManager(string neutralLanguageFile,   string neutralLanguage) {
  Instances.Add(this);
  localeClient.GetResourcesCompleted +=
    localeClient_GetResourcesCompleted;

  if (Instances.Count == 1) {
    if (string.IsNullOrWhiteSpace(neutralLanguageFile)) {
      // No neutral resources
      ChangeCulture(Thread.CurrentThread.CurrentUICulture);
    }
    else {
      LoadNeutralResources(neutralLanguageFile, neutralLanguage);
    }
  }
}
```

Silverlight application. Because it's publicly accessible, don't put any sensitive data in the file.

ConfigurationManager also reads from the ClientBin directory. It looks for a file called appSettings.xml, and the file looks like this:

```
<AppSettings>
  <Add Key="neutralLanguageFile" Value="strings.xml"/>
  <Add Key="neutralLanguage" Value="en-US"/>
</AppSettings>
```

Once appSettings.xml and strings.xml are in place, ConfigurationManager and SmartResourceManager can work together to load the neutral language. There's room for improvement in this process, because if the thread's active culture is different than the neutral language and the Web service is down, the thread's active culture will be different than the active resource set. I'll leave that as an exercise for you.

Wrapping Up

What I didn't go over in this article was normalizing the resources on the server side. Let's say that the fr-FR resource is missing two keys that the fr resource has. When requesting the fr-FR resources, the Web service should insert the missing keys from the more general fr resource.

Another aspect that's built into this solution that I didn't cover is loading resources by culture and resource set instead of just culture. This is useful for loading resources per screen or per .xap file.

The solution I walked you through here does allow you to do a few useful things, however, including loading resource sets at run time, using any format for storing resources, changing resources without recompiling and lazy loading resources.

The solution presented here is general-purpose, and you can hook into it in multiple points and drastically change the implementation. I hope this helps reduce your daily programming load.

For further in-depth reading about internationalization, check out the book ".NET Internationalization: The Developer's Guide to Building Global Windows and Web Applications" (Addison-Wesley, 2006), by Guy Smith-Ferrier. Smith-Ferrier also has a great video on internationalization on his Web site; the video is called "Internationalizing Silverlight at SLUGUK" (bit.ly/gJGptU). ■

MATTHEW DELISLE *enjoys studying both the software and hardware aspects of computers. His first daughter was born in 2010 and he thinks she's almost ready to begin her programming career. Keep up with Delisle via his blog at mattdelisle.net.*

THANKS to the following technical expert for reviewing this article: *John Brodeur*

Writing a Debugging Tools for Windows Extension

Andrew Richards

Troubleshooting production issues can be one of the most frustrating jobs that any engineer can do. It can also be one of the most rewarding jobs. Working in Microsoft Support, I'm faced with this every day. Why did the application crash? Why did it hang? Why is it having a performance issue?

Learning how to debug can be a daunting task, and is one of those skills that requires many hours of regular practice to stay proficient. But it's a crucial skill for being an all-around developer. Still, by bottling the skills of a few debugging experts, debugger engineers of all skill levels can execute extremely complex debugging logic as easy-to-run commands.

Myriad troubleshooting techniques can be used to get to the root cause of a crash, but the most valuable—and most fruitful to engineers with debugging skills—is a process dump. Process dumps contain a snapshot of a process memory at the time of capture. Depending on the dumping tool, this could be the entire address space or just a subset.

Windows automatically creates a minidump through Windows Error Reporting (WER) when any application throws an unhandled exception. In addition, you can manually create a dump file via the Userdump.exe tool. The Sysinternals tool ProcDump (technet.microsoft.com/sysinternals/dd996900) is becoming the preferred process-dumping tool of Microsoft Support because it can capture a dump based upon a large variety of triggers and can generate dumps of various sizes. But once you have the dump data, what can you do with it to aid debugging?

Various versions of Visual Studio support opening dump files (.dmp), but the best tool to use is a debugger from Debugging Tools for Windows. These tools are all based on a single debugging engine that supports two debugger extension APIs. In this article, I'm going to cover the basics of building a custom debugger extension so you can analyze these dump files (and also live systems) with ease.

This article discusses:

- Debugging tools basics
- Using the debugger APIs
- Symbol resolution
- Dealing with processor types

Technologies discussed:

Debugging Tools for Windows, C++

Code download available at:

code.msdn.microsoft.com/mag201103Debugger

Setting up the Tools

Debugging Tools for Windows (microsoft.com/whdc/devtools/debugging) is an installable and redistributable component of the Windows SDK and Windows Driver Kit (WDK). As I write this, the current version is 6.12, and it's available in version 7.1 of the Windows SDK or the WDK. I recommend using the most-recent version, as the debugging engine has many valuable additions, including better stack walking.

Figure 1 **Sources**

```
TARGETNAME=MyExt
TARGETTYPE=DYNLINK

_NT_TARGET_VERSION=$(_NT_TARGET_VERSION_WINXP)

DLLENTRY=_DllMainCRTStartup

!if "$(DBGSDK_INC_PATH)" != ""
INCLUDES = $(DBGSDK_INC_PATH);$(INCLUDES)
!endif
!if "$(DBGSDK_LIB_PATH)" == ""
DBGSDK_LIB_PATH = $(SDK_LIB_PATH)
!else
DBGSDK_LIB_PATH = $(DBGSDK_LIB_PATH)\$(TARGET_DIRECTORY)
!endif

TARGETLIBS=$(SDK_LIB_PATH)\kernel32.lib \
           $(DBGSDK_LIB_PATH)\dbgeng.lib

USE_MSVCRT=1

UMTYPE=windows

MSC_WARNING_LEVEL = /W4 /WX

SOURCES= dbgexts.rc      \
         dbgexts.cpp     \
         myext.cpp
```

The Debugging Tools for Windows guidelines say that you should compile debugger extensions using the WDK build environment. I use the latest release of the WDK (version 7.1.0 build 7600.16385.1), but any version of the WDK or its previous incarnation as the Driver Development Kit (DDK) will suffice. When building an extension using the WDK, you use x64 and x86 Free Build environments.

With a little bit of effort you can also adapt my projects to build in the Windows SDK build environment or Visual Studio.

One note of warning: The WDK doesn't like spaces in path names. Make sure you compile from an unbroken path. For example, use something like C:\Projects instead of C:\Users\Andrew Richards\Documents\Projects.

Regardless of how you build the extension, you'll need the header and library files of the Debugging Tools SDK, which is a component of Debugging Tools for Windows. The examples in this article use my x86 path (C:\debuggers_x86\sdk) when referencing the header and library files. If you choose to install the debugger elsewhere, remember to update the path and add quotes when necessary to accommodate spaces in the path names.

Using the Debugging Tools

The Debugging Tools for Windows debuggers are architecture-agnostic. Any edition of the debugger can debug any target architecture. A common example is using the x64 debugger to debug an x86 application. The debugger is released for x86, x64 (amd64) and IA64, but it can debug x86, x64, IA64, ARM, EBC and PowerPC (Xbox) applications. You can install all of the debugger editions side-by-side.

This agility isn't universally understood, though. Not all debugger extensions adapt to the target architecture as well as the debugger engine does. Some debugger extensions assume that the pointer size of the target will be the same as the pointer size of the debugger. Similarly, they use the wrong hardcoded register (esp in place of rsp, for example) instead of a pseudo-register such as $csp.

If you're having an issue with a debugger extension, you should try running the debugger designed for the same architecture as the target environment. This might overcome the assumptions of the poorly written extension.

Each application build type and associated processor architecture comes with its own set of debugging headaches. The assembler generated for a debug build is relatively linear, but the assembler generated for a release build is optimized and can resemble a bowl of spaghetti. On x86 architectures, Frame Pointer Omission (FPO) plays havoc with call stack reconstruction (the latest debugger handles this well). On x64 architectures, function parameters and local variables are stored in registers. At the time of dump capture, they may have been pushed onto the stack, or may no longer exist due to register reuse.

Experience is the key here. To be accurate, one person's experience is the key here. You just need to bottle their know-how in a debugger extension for the rest of us. It only takes a few repetitions of a similar debugging sequence before I automate it as a debugger extension. I've used some of my extensions so much that I forget how I did the same thing using the underlying debugging commands.

Using the Debugger APIs

There are two debugger extension APIs: the deprecated WdbgExts API (wdbgexts.h) and the current DbgEng API (dbgeng.h).

WdbgExts extensions are based on a global call that's configured at initialization (WinDbgExtensionDllInit):

```
WINDBG_EXTENSION_APIS ExtensionApis;
```

Figure 2 **dbgexts.cpp**

```
// dbgexts.cpp

#include "dbgexts.h"

extern "C" HRESULT CALLBACK
DebugExtensionInitialize(PULONG Version, PULONG Flags) {
  *Version = DEBUG_EXTENSION_VERSION(EXT_MAJOR_VER, EXT_MINOR_VER);
  *Flags = 0;  // Reserved for future use.
  return S_OK;
}

extern "C" void CALLBACK
DebugExtensionNotify(ULONG Notify, ULONG64 Argument) {
  UNREFERENCED_PARAMETER(Argument);
  switch (Notify) {
    // A debugging session is active. The session may not necessarily be suspended.
    case DEBUG_NOTIFY_SESSION_ACTIVE:
      break;
    // No debugging session is active.
    case DEBUG_NOTIFY_SESSION_INACTIVE:
      break;
    // The debugging session has suspended and is now accessible.
    case DEBUG_NOTIFY_SESSION_ACCESSIBLE:
      break;
    // The debugging session has started running and is now inaccessible.
    case DEBUG_NOTIFY_SESSION_INACCESSIBLE:
      break;
  }
  return;
}

extern "C" void CALLBACK
DebugExtensionUninitialize(void) {
  return;
}
```

The global provides the functionality required to run functions such as dprintf("\n") and GetExpression("@$csp") without any namespace. This type of extension resembles the code you'd write when doing Win32 programming.

DbgEng extensions are based on debugger interfaces. The IDebugClient interface is passed to you by the debug engine as a parameter of each call. The interfaces support QueryInterface for access to the ever-increasing range of debugger interfaces. This type of extension resembles the code you'd write when doing COM programming.

It's possible to make a hybrid of the two extension types. You expose the extension as DbgEng, but add the functionality of the WdbgExts API at run time via a call to IDebugControl::GetWindbgExtensionApis64. As an example, I've written the classic "Hello World" as a DbgEng extension in C. If you prefer C++, refer to the ExtException class in the Debugging Tools SDK (.\inc\engextcpp.cpp).

Compile the extension as MyExt.dll (TARGETNAME in the sources file shown in **Figure 1**). It exposes a command called !helloworld. The extension dynamically links to the Microsoft Visual C runtime (MSVCRT). If you want to use static, change the USE_MSVCRT=1 statement to USE_LIBCMT=1 in the sources file.

The DebugExtensionInitialize function (see **Figure 2**) is called when the extension is loaded. Setting the Version parameter is a simple matter of using the DEBUG_EXTENSION_VERSION macro with the EXT_MAJOR_VER and EXT_MINOR_VER #defines I've added to the header file:

```
// dbgexts.h

#include <windows.h>
#include <dbgeng.h>

#define EXT_MAJOR_VER  1
#define EXT_MINOR_VER  0
```

The Version value is reported as the API version in the debugger .chain command. To change the File Version, File Description, Copyright and other values you need to edit the dbgexts.rc file:

```
myext.dll: image 6.1.7600.16385, API 1.0.0, built Wed Oct 13 20:25:10 2010
  [path: C:\Debuggers_x86\myext.dll]
```

The Flags parameter is reserved and should be set to zero. The function needs to return S_OK.

The DebugExtensionNotify function is called when the session changes its active or accessible status. The Argument parameter is wrapped with the UNREFERENCED_PARAMETER macro to eliminate the unused parameter compiler warning.

Figure 3 **MyExt.cpp**

```
// MyExt.cpp

#include "dbgexts.h"

HRESULT CALLBACK
helloworld(PDEBUG_CLIENT pDebugClient, PCSTR args) {
  UNREFERENCED_PARAMETER(args);

  IDebugControl* pDebugControl;
  if (SUCCEEDED(pDebugClient->QueryInterface(__uuidof(IDebugControl),
    (void **)&pDebugControl))) {
    pDebugControl->Output(DEBUG_OUTPUT_NORMAL, "Hello World!\n");
    pDebugControl->Release();
  }
  return S_OK;
}
```

I've added the switch statement for the Notify parameter for completeness, but I haven't added any functional code in this area. The switch statement processes four session state changes:

- DEBUG_NOTIFY_SESSION_ACTIVE occurs when you attach to a target.
- DEBUG_NOTIFY_SESSION_INACTIVE occurs when the target becomes detached (via .detach or qd).
- If the target suspends (hits a breakpoint, for example), the function will be passed DEBUG_NOTIFY_SESSION_ACCESSIBLE.
- If the target resumes running, the function will be passed DEBUG_NOTIFY_SESSION_INACCESSIBLE.

The DebugExtensionUninitialize function is called when the extension is unloaded.

Each extension command to be exposed is declared as a function of type PDEBUG_EXTENSION_CALL. The name of the function is the name of the extension command. Because I'm writing "Hello World," I've named the function helloworld (see **Figure 3**).

Note that the convention is to use lower-case function names. Because I'm using the WDK build environment, the myext.def file also needs to be changed. The name of the extension command needs to be added so that it's exported:

```
;-------------
;   MyExt.def
;-------------
EXPORTS
  helloworld
  DebugExtensionNotify
  DebugExtensionInitialize
  DebugExtensionUninitialize
```

The args parameter contains a string of the arguments for the command. The parameter is passed as a null-terminated ANSI string (CP_ACP).

The pDebugClient parameter is the IDebugClient interface pointer that allows the extension to interact with the debugging engine. Even though the interface pointer looks like it's a COM Interface pointer, it can't be marshaled, nor accessed at a later time. It also can't be used from any other thread. To do work on an alternate thread, a new debugger client (a new interface pointer to IDebugClient) must be created on that thread using IDebugClient::CreateClient. This is the only function that can be run on an alternate thread.

The IDebugClient interface (like all interfaces) is derived from IUnknown. You use QueryInterface to access the other DbgEng interfaces, be they later versions of the IDebugClient interface (IDebugClient4) or different interfaces (IDebugControl, IDebugRegisters, IDebugSymbols, IDebugSystemObjects and so on). To output text to the debugger, you need the IDebugControl interface.

I have two non-SDK files in the folder to help with development. The make.cmd script adds the Debugger SDK inc and lib paths to the WDK build environment, then runs the appropriate build command:

```
@echo off
set DBGSDK_INC_PATH=C:\Debuggers_x86\sdk\inc
set DBGSDK_LIB_PATH=C:\Debuggers_x86\sdk\lib
set DBGLIB_LIB_PATH=C:\Debuggers_x86\sdk\lib
build -cZMg %1 %2
```

Note that the WDK build environment itself determines whether an x86 or x64 binary will be built. If you want to build for multiple

architectures, you'll need to open multiple prompts and run make.cmd in each. The building can be done concurrently.

Once built, I use the (x86) test.cmd script to copy the compiled i386 binaries into the x86 debugger folder (c:\Debuggers_x86), then launch an instance of Notepad with the debugger attached and the extension loaded:

```
@echo off
copy objfre_win7_x86\i386\myext.dll c:\Debuggers_x86
copy objfre_win7_x86\i386\myext.pdb c:\Debuggers_x86
\Debuggers_x86\windbg.exe -a myext.dll -x notepad
```

If everything has gone as planned, I can type "!helloworld" in the debugger command prompt and see a "Hello World!" response:

```
0:000> !helloworld
Hello World!
```

Symbol Resolution and Reading

The "Hello World" application may be amazing, but you can do better. I'm now going to use this infrastructure to add a command that actually interacts with the target and would help you do some analysis. The simple test01 application has a global pointer that's assigned a value:

```
// test01.cpp

#include <windows.h>

void* g_ptr;
int main(int argc, char* argv[]) {
  g_ptr = "This is a global string";
  Sleep(10000);
  return 0;
}
```

The new !gptr command in MyExt.cpp (see **Figure 4**) will resolve the test01!g_ptr global, read the pointer and then output the values found in the same format as "x test01!g_ptr":

```
0:000> x test01!g_ptr
012f3370 Test01!g_ptr = 0x012f20e4

0:000> !gptr
012f3370 test01!g_ptr = 0x012f20e4
<string>
```

The first step is to determine the location of the test01!g_ptr pointer. The pointer will be in a different location each time the application runs because Address Space Layout Randomization (ASLR) will change the module load address. To get the location, I use QueryInterface to get the IDebugSymbols interface, and then use GetOffsetByName. The GetOffsetByName function takes a symbol name and returns the address as a 64-bit pointer. The debugger functions always return 64-bit pointers (ULONG64) so that 64-bit targets can be debugged with a 32-bit debugger.

Remember, this is the address of the pointer in the target address space, not your own. You can't just read from it to determine its value. To get the value of the pointer, I use QueryInterface again to get the IDebugDataSpaces interface, and then use ReadPointersVirtual. This reads the pointer from the target address space. ReadPointersVirtual automatically adjusts for pointer size and endian differences. You don't need to manipulate the pointer returned.

IDebugControl::Output takes the same format string as printf, but also has formatters that allow you to reference the target address space. I use the %ma format to print out the ANSI string that the global pointer points to in the target address space. The %p format is pointer-size-aware and should be used for pointer output (you must pass a ULONG64).

I've modified the test script to load a dump file of the x86 version of test01 instead of launching Notepad:

```
@echo off
copy objfre_win7_x86\i386\myext.dll c:\Debuggers_x86
copy objfre_win7_x86\i386\myext.pdb c:\Debuggers_x86
\Debuggers_x86\windbg.exe -a myext.dll -y "..\Test01\x86;SRV*c:\symbols*http://
msdl.microsoft.com/download/symbols" -z ..\Test01\x86\Test01.dmp
```

I've also set the symbol path to the test01 x86 folder and the Microsoft Public Symbol Server so that everything can be resolved. Additionally, I've made an x64 test script that does the same as the x86 test script, but with a dump file of the x64 version of the test application:

```
@echo off
copy objfre_win7_x86\i386\myext.dll c:\Debuggers_x86
copy objfre_win7_x86\i386\myext.pdb c:\Debuggers_x86
\Debuggers_x64\windbg.exe -a myext.dll -y "..\Test01\x64;SRV*c:\symbols*http://
msdl.microsoft.com/download/symbols" -z ..\Test01\x64\Test01.dmp
```

When I run the scripts, the x86 debugger is launched, the appropriate dump file is opened, the x86 version of the extension is loaded and symbols are resolvable.

Once again, if everything has gone to plan, I can type "x test01!g_ptr" and !gptr in the debugger command prompt and see similar responses:

```
// x86 Target
0:000> x test01!g_ptr
012f3370 Test01!g_ptr = 0x012f20e4

0:000> !gptr
012f3370 test01!g_ptr = 0x012f20e4
This is a global string

// x64 Target
0:000> x test01!g_ptr
00000001`3fda35d0 Test01!g_ptr = 0x00000001`3fda21a0

0:000> !gptr
000000013fda35d0 test01!g_ptr = 0x000000013fda21a0
This is a global string
```

Figure 4 **Revised MyExt.cpp**

```
HRESULT CALLBACK
gptr(PDEBUG_CLIENT pDebugClient, PCSTR args) {
  UNREFERENCED_PARAMETER(args);

  IDebugSymbols* pDebugSymbols;
  if (SUCCEEDED(pDebugClient->QueryInterface(__uuidof(IDebugSymbols),
    (void **)&pDebugSymbols))) {
    // Resolve the symbol.
    ULONG64 ulAddress = 0;
    if (SUCCEEDED(pDebugSymbols->GetOffsetByName("test01!g_ptr", &ulAddress))) {
      IDebugDataSpaces* pDebugDataSpaces;
      if (SUCCEEDED(pDebugClient->QueryInterface(__uuidof(IDebugDataSpaces),
        (void **)&pDebugDataSpaces))) {
        // Read the value of the pointer from the target address space.
        ULONG64 ulPtr = 0;
        if (SUCCEEDED(pDebugDataSpaces->ReadPointersVirtual(1, ulAddress,
&ulPtr))) {
          PDEBUG_CONTROL pDebugControl;
          if (SUCCEEDED(pDebugClient->QueryInterface(__uuidof(IDebugControl),
            (void **)&pDebugControl))) {
            // Output the values.
            pDebugControl->Output(DEBUG_OUTPUT_NORMAL,
              "%p test01!g_ptr = 0x%p\n", ulAddress, ulPtr);
            pDebugControl->Output(DEBUG_OUTPUT_NORMAL, "%ma\n", ulPtr);
            pDebugControl->Release();
          }
        }
        pDebugDataSpaces->Release();
      }
      pDebugSymbols->Release();
    }
  }
  return S_OK;
}
```

Figure 5 **Sleepy**

```
HRESULT CALLBACK
sleepy(PDEBUG_CLIENT4 Client, PCSTR args) {
  UNREFERENCED_PARAMETER(args);
  BOOL bFound = FALSE;

  IDebugControl* pDebugControl;

  if (SUCCEEDED(Client->QueryInterface(__uuidof(IDebugControl),
    (void **)&pDebugControl))) {
    IDebugSymbols* pDebugSymbols;

    if (SUCCEEDED(Client->QueryInterface(__uuidof(IDebugSymbols),
      (void **)&pDebugSymbols))) {
      DEBUG_STACK_FRAME* pDebugStackFrame =
        (DEBUG_STACK_FRAME*)malloc(
        sizeof(DEBUG_STACK_FRAME) * MAX_STACK_FRAMES);

      if (pDebugStackFrame != NULL) {
        // Get the Stack Frames.
        memset(pDebugStackFrame, 0, (sizeof(DEBUG_STACK_FRAME) *
          MAX_STACK_FRAMES));
        ULONG Frames = 0;

        if (SUCCEEDED(pDebugControl->GetStackTrace(0, 0, 0,
          pDebugStackFrame, MAX_STACK_FRAMES, &Frames)) &&
          (Frames > 0)) {
          ULONG ProcessorType = 0;
          ULONG SymSize = 0;
          char SymName[4096];
          memset(SymName, 0, 4096);
          ULONG64 Displacement = 0;

          if (SUCCEEDED(pDebugControl->GetEffectiveProcessorType(
            &ProcessorType))) {
            for (ULONG n=0; n<Frames; n++) {

              // Use the Effective Processor Type and the contents
              // of the frame to determine existence
              if (SUCCEEDED(pDebugSymbols->GetNameByOffset(
                pDebugStackFrame[n].InstructionOffset, SymName, 4096,
                &SymSize, &Displacement)) && (SymSize > 0)) {

                if ((ProcessorType == IMAGE_FILE_MACHINE_I386) &&
                  (_stricmp(SymName, "KERNELBASE!Sleep") == 0) &&
                  (Displacement == 0xF)) {
                  // Win7 x86; KERNELBASE!Sleep+0xF is usually in frame 3.
                  IDebugDataSpaces* pDebugDataSpaces;

                  if (SUCCEEDED(Client->QueryInterface(
                    __uuidof(IDebugDataSpaces),
                    (void **)&pDebugDataSpaces))) {
                    // The value is pushed immediately prior to
                    // KERNELBASE!Sleep+0xF
                    DWORD dwMilliseconds = 0;

                    if (SUCCEEDED(pDebugDataSpaces->ReadVirtual(
                      pDebugStackFrame[n].StackOffset, &dwMilliseconds,
                      sizeof(dwMilliseconds), NULL))) {
                      pDebugControl->Output(DEBUG_OUTPUT_NORMAL,
                        "Sleeping for %ld msec\n", dwMilliseconds);
                      bFound = TRUE;
                    }
                    pDebugDataSpaces->Release();
                  }
                  if (bFound) break;
                }

                else if ((ProcessorType == IMAGE_FILE_MACHINE_AMD64) &&
                  (_stricmp(SymName, "KERNELBASE!SleepEx") == 0) &&
                  (Displacement == 0xAB)) {
                  // Win7 x64; KERNELBASE!SleepEx+0xAB is usually in frame 1.
                  IDebugRegisters* pDebugRegisters;

                  if (SUCCEEDED(Client->QueryInterface(
                    __uuidof(IDebugRegisters),
                    (void **)&pDebugRegisters))) {
                    // The value is in the 'rsi' register.
                    ULONG rsiIndex = 0;
                    if (SUCCEEDED(pDebugRegisters->GetIndexByName(
                      "rsi", &rsiIndex)))
                    {
                      DEBUG_VALUE debugValue;
                      if (SUCCEEDED(pDebugRegisters->GetValue(
                        rsiIndex, &debugValue)) &&
                        (debugValue.Type == DEBUG_VALUE_INT64)) {
                        // Truncate to 32bits for display.
                        pDebugControl->Output(DEBUG_OUTPUT_NORMAL,
                          "Sleeping for %ld msec\n", debugValue.I32);
                        bFound = TRUE;
                      }
                    }
                    pDebugRegisters->Release();
                  }

                  if (bFound) break;
                }
              }
            }
          }
          free(pDebugStackFrame);
        }
        pDebugSymbols->Release();
      }
      if (!bFound)
        pDebugControl->Output(DEBUG_OUTPUT_NORMAL,
          "Unable to determine if Sleep is present\n");
      pDebugControl->Release();
    }
    return S_OK;
}
```

If you repeat the test using the x64 debugger, the amd64-compiled version of the debugger extension and the x86 or x64 dump files, you'll get the same results. That is, the extension is architecture-agnostic.

Processor Types and Stacks

I'm now going to extend this infrastructure once again. Let's add a command that finds the duration of a Sleep call on the current thread stack. The !sleepy command (see **Figure 5**) will resolve the call stack symbols, look for the Sleep function and read the DWORD that represents the milliseconds to delay, and then will output the delay value (if found).

To add some complexity to the command, the command will support the x86 and x64 versions of the test01 application. Because the calling convention is different for x86 and x64 applications, the command will have to be aware of the target's architecture as it progresses.

The first step is to get the stack frames. To get the frames, I use QueryInterface to get the IDebugControl interface, and then use GetStackTrace to retrieve information about each stack frame. GetStackTrace takes an array of DEBUG_STACK_FRAME structures. I always allocate the array of DEBUG_STACK_FRAME structures on the heap so that I don't cause a stack overflow. If you're retrieving a stack overflow thread of a target, you'll probably hit your own stack limit if the array is allocated on your stack.

If GetStackTrace succeeds, the array will be populated with information for each frame that was walked. Success here doesn't necessarily mean that the frame information is correct. The debugger does its best to walk the stack frames, but mistakes can be made when the symbols aren't correct (when they're missing or have been forced). If you've used ".reload /f /i" to force the symbol load, poor symbol alignment will probably occur.

To effectively use the contents of each of the DEBUG_STACK_FRAME structures, I need to know the target's effective processor type. As mentioned previously, the target architecture can be completely different from the debugger extension architecture. The effective processor type (.effmach) is the architecture that the target is currently using.

The processor type can also be a different processor type than that used by the target's host. The most common example of this is when the target is an x86 application that's running via Windows 32-bit on Windows 64-bit (WOW64) on an x64 edition of Windows. The effective processor type is IMAGE_FILE_MACHINE_I386. The actual type is IMAGE_FILE_MACHINE_AMD64.

This means you should consider an x86 application to be an x86 application regardless of whether it's running on an x86 edition of Windows or an x64 edition of Windows. (The only exception to this rule is when you're debugging the WOW64 calls that surround the x86 process.)

To get the effective processor type, I use the IDebugControl interface that I already have, and then use GetEffectiveProcessorType.

If the effective processor type is i386, I need to look for the KERNELBASE!Sleep+0xf function. If all the symbols are resolved correctly, the function should be in frame 3:

```
0:000> knL4
 # ChildEBP RetAddr
00 001bf9dc 76fd48b4 ntdll!KiFastSystemCallRet
01 001bf9e0 752c1876 ntdll!NtDelayExecution+0xc
02 001bfa48 752c1818 KERNELBASE!SleepEx+0x65
03 001bfa58 012f1015 KERNELBASE!Sleep+0xf
```

If the effective processor type is AMD64, I look for the KERNELBASE!SleepEx+0xab function. If all the symbols are resolved correctly, the function should be in frame 1:

```
0:000> knL2
 # Child-SP          RetAddr           Call Site
00 00000000`001cfc08 000007fe`fd9b1203 ntdll!NtDelayExecution+0xa
01 00000000`001cfc10 00000001`3fda101d KERNELBASE!SleepEx+0xab
```

However, based on the level of symbol resolution available, the function symbol I'm looking for may or may not be in the expected frame. If you open the test01 x86 dump file and don't specify a symbol path, you can see an example of this. The KERNELBASE!Sleep call will be in frame 1 instead of frame 3:

```
0:000> knL4
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 001bfa48 752c1818 ntdll!KiFastSystemCallRet
01 001bfa58 012f1015 KERNELBASE!Sleep+0xf
02 001bfaa4 75baf4e8 Test01+0x1015
03 001bfab0 76feaf77 kernel32!BaseThreadInitThunk+0x12
```

The debugger warns you of this possible mistake. If you want to have your extension adapt to these types of issues, you should iterate over the frames as I have, instead of just looking at the expected frame.

To determine the existence of the Sleep function, I need to look up the symbol for each frame. If the effective processor type and symbol make a valid pair, the function has been found. Note that this logic is fragile and is being used to simplify the example. The symbol may change between builds and platforms. For example, Windows Server 2008 is kernel32!Sleep+0xf, but Windows 7 is KERNELBASE!Sleep+0xf.

To get the symbol, I use QueryInterface to get the IDebugSymbol interface. I then use GetNameByOffset to get the symbol of the instruction offset address.

There are two parts to the symbol: the symbol name (KERNELBASE!Sleep) and the displacement (0xf). The symbol name is an amalgamation of the module name and the function name (<module>!<function>). The displacement is the byte offset from the start of the function to which program flow will return to after the call has returned.

If there are no symbols, the function will be reported as just the module name with a large displacement (Test01+0x1015).

Once I've found the frame, the next step is to extract the delay. When the target is x86 based, the delay will be in a DWORD that has been pushed onto the stack immediately prior to the function call (note that this is fragile logic):

```
// @$csp is the pseudo-register of @esp
0:000> dps @$csp
<snip>
001bfa4c  752c1818 KERNELBASE!Sleep+0xf
001bfa50  00002710
<snip>
```

The StackOffset member of the DEBUG_STACK_FRAME structure actually points to this address already, so no pointer arithmetic is necessary. To get the value, I use QueryInterface to get the IDebugDataSpaces interface, and then use ReadVirtual to read the DWORD from the target address space.

If the target is x64 based, the delay isn't in the stack—it's in the rsi register (this is also fragile logic due to its frame-context dependency):

```
0:000> r @rsi
rsi=0000000000002710
```

To get the value, I use QueryInterface to get the IDebugRegisters interface. I first need to use GetIndexByName to get the index of the rsi register. I then use GetValue to read the register value from the target registers. Because rsi is a 64-bit register, the value is returned as an INT64. Because the DEBUG_VALUE structure is a union, you can simply reference the I32 member instead of the I64 member to get the truncated version that represents the DWORD passed to Sleep.

Once again, in both cases, I use the IDebugControl::Output function to output the result.

Break!

In this article, I barely scratched the surface of what can be achieved. Stacks, symbols, registers, memory I/O and environmental information are but a few of the many things that you can interrogate and change from within an extension.

In a future article I'll delve deeper into the relationship a debugger extension can have with the debugger. I'll cover debugger clients and debugger callbacks, and I'll use these to encapsulate the SOS debugger extension so that you can write an extension that can debug managed applications without having to have any knowledge of the underlying .NET structures. ■

ANDREW RICHARDS *is a Microsoft senior escalation engineer for Exchange Server. He has a passion for support tools, and is continually creating debugger extensions and applications that simplify the job of support engineers.*

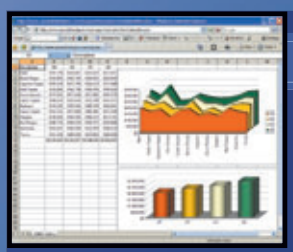
THANKS to the following technical experts for reviewing this article: *Drew Bliss, Jen-Lung Chiu, Mike Hendrickson, Ken Johnson, Brunda Nagalingaiah and Matt Weber*

20 Minutes to 4 Seconds...

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

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

SpreadsheetGear 2010



ASP.NET Excel Reporting

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

Excel Compatible Windows Forms Control

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

Create Dashboards from Excel Charts and Ranges

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

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

www.SpreadsheetGear.com.

SpreadsheetGear

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

Building Data-Centric Web Apps with ASP.NET MVC and Ext JS

Juan Carlos Olamendy

A rich Internet application (RIA) combines the usability of a desktop app with the flexibility of Web-based deployment and revision. There are two key approaches to building RIAs. First, there are browser plug-ins that host execution environments such as Flash, Java and Silverlight. Second, there are JavaScript-based extension libraries such as Dojo, Ext JS, jQuery, MooTools, Prototype and YUI. Each approach has its advantages and disadvantages.

JavaScript libraries are a popular choice for building RIAs because JavaScript is supported by all major browsers and there's no need to install a plug-in or runtime environment. I've been experimenting with another of the libraries mentioned—Ext JS—and I've found that it makes an interesting choice for implementing Web apps. It's easy to implement, well-documented and is compatible with Selenium for testing. Ext JS also provides pre-defined controls that simplify creating the UI of your Web app.

Unfortunately, most examples of Ext JS are illustrated with PHP, Python and Ruby on Rails code on the server side. But that doesn't mean developers using Microsoft technologies can't take advantage of Ext JS. While it's difficult to integrate Ext JS with Web Forms development (due to the abstraction layer that encapsulates the request-response nature of the Web in order to provide a stateful control-based model), you could use the ASP.NET MVC framework, enabling you to leverage both the Microsoft .NET Framework and Ext JS in the same app.

In this article, I'll provide the tutorial I couldn't find, walking through the development of a real-world Web solution using ASP.NET MVC and Ext JS that reads from and writes to a back-end database.

This article discusses:

- Ext JS form basics
- Creating the data store
- Using ASP.NET MVC
- Tying the layers together

Technologies discussed:

ASP.NET MVC, Ext JS, SQL Server 2008

Code download available at:

code.msdn.microsoft.com/mag201103ASPNET

Ext JS Form Basics

To use Ext JS, you first need to download it from sencha.com. (I used version 3.2.1, but you should grab the most recent version.) Note that a free, open source version of Ext JS is available for open source projects, non-profit organizations and educational use. For other uses you may need to purchase a license. See sencha.com/products/license.php for more information.

Uncompress the download into a directory in your file system. It contains everything you need to develop a Web solution using Ext JS, in particular the main file ext-all.js. (There's also a debug version to help find errors more easily.) Dependencies, documentation and example code are all included in the download.

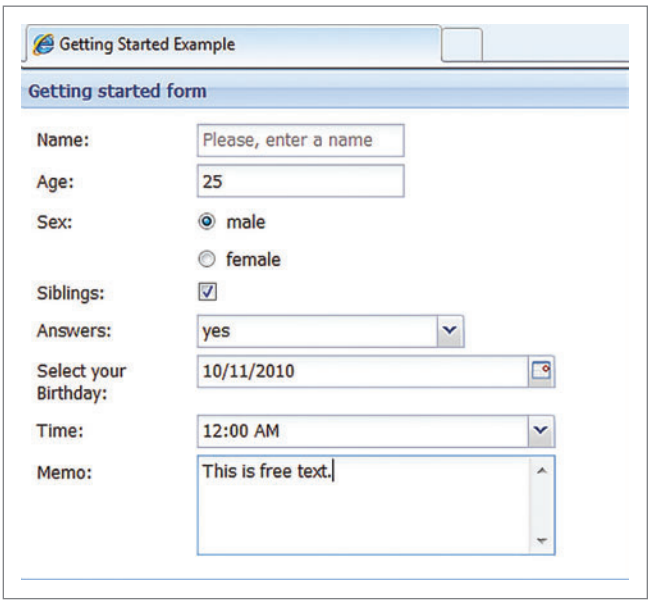


Figure 1 **The Completed Form**

The required folders for a project are \adapters and \resources. The adapters folder enables use of other libraries alongside Ext JS. The resources folder contains dependencies such as CSS and images.

To use Ext JS correctly, you'll also need to include the three key file references in your pages:

```
ext-3.2.1/adapter/ext/ext-base.js
ext-3.2.1/ext-all.js
ext-3.2.1/resources/css/ext-all.css
```

The ext-base.js file contains the core functionality of Ext JS. The widget definitions are contained in ext-all.js, and ext-all.css includes stylesheets for the widgets.

Let's start using Ext JS in a static HTML page to illustrate the basics. The following lines are contained within the head section of the page and link the required files to successfully develop an Ext JS solution (I've also included the JavaScript module with some example widgets from the Ext JS download):

```
<link rel="stylesheet" type="text/css"
  href="ext-3.2.1/resources/css/ext-all.css" />
<script type="text/javascript" language="javascript"
  src="ext-3.2.1/adapter/ext/ext-base.js"></script>
<script type="text/javascript" language="javascript"
  src="ext-3.2.1/ext-all.js"></script>
<script type="text/javascript" language="javascript"
  src="extjs-example.js"></script>
```

Within the body of the file, I insert a div element for rendering the main Ext JS form:

```
<div id="frame">
</div>
```

The extjs-example.js file provides some insight into how Ext JS applications are constructed. The template for any Ext JS application uses the Ext.ns, Ext.BLANK_IMAGE_URL and Ext.onReady statements:

```
Ext.ns('formextjs.tutorial');
Ext.BLANK_IMAGE_URL = 'ext-3.2.1/resources/images/default/s.gif';
formextjs.tutorial.FormTutorial = {
  ...
}
Ext.onReady(formextjs.tutorial.FormTutorial.init,
  formextjs.tutorial.FormTutorial);
```

The Ext.ns statement enables you to logically organize your code in a namespace, in order to avoid naming conflicts and scoping problems.

The Ext.BLANK_IMAGE_URL statement is important for rendering the widgets. It's called the spacer image (a transparent 1x1 pixel image) and mainly used to generate the blank space as well as to place icons and separators.

The Ext.onReady statement is the first method to define with Ext JS code. This method is automatically called once the DOM is fully loaded, guaranteeing that every HTML element that you may reference is available when the script runs. In the case of extjs-example.js, here's the script itself:

```
formextjs.tutorial.FormTutorial = {
  init: function () {
    this.form = new Ext.FormPanel({
      title: 'Getting started form',
      renderTo: 'frame',
      width: 400,
      url: 'remoteurl',
      defaults: { xtype: 'textfield' },
      bodyStyle: 'padding: 10px',
      html: 'This form is empty!'
    });
  }
}
```

An instance of the class Ext.FormPanel is created as a container for the fields. The renderTo property points to the div element where the form will be rendered. The defaults property specifies the default component type on the form. The url property specifies the URI to send the request of the form. Finally, the html property specifies the text (with any HTML formatting) as the default output.

To add fields, you need to replace the html property with the items property:

```
items: [ nameTextField, ageNumberField ]
```

The first two items to add are a text field and a number field:

```
var nameTextField = new Ext.form.TextField({
  fieldLabel: 'Name',
  emptyText: 'Please, enter a name',
  name: 'name'
});
var ageNumberField = new Ext.form.NumberField({
  fieldLabel: 'Age',
  value: '25',
  name: 'age'
});
```

The required properties are: fieldLabel property (to set a descriptive message accompanying the component of the form) and name property (to set name of the request parameter). The emptyText property defines the watermark text that the field will contain when it's empty. The value property is the default value for the control.

Figure 2 **Form Buttons**

```
buttons: [{
  text: 'Save',
  handler: function () {
    form.getForm().submit({
      success: function (form, action) {
        Ext.Msg.alert('Success', 'ok');
      },
      failure: function (form, action) {
        Ext.Msg.alert('Failure', action.result.error);
      }
    });
  }
},
{
  text: 'Reset',
  handler: function () {
    form.getForm().reset();
  }
}]
```

Figure 3 **Creating the Human Resources Database**

```
create table department(
  deptno varchar(20) primary key,
  deptname varchar(50) not null,
  location varchar(50)
);

create unique index undx_department_deptname on department(deptname);

insert into department
  values('HQ-200','Headquarter-NY','New York');
insert into department
  values('HR-200','Human Resources-NY','New York');
insert into department
  values('OP-200','Operations-NY','New York');
insert into department
  values('SL-200','Sales-NY','New York');
insert into department
  values('HR-300','Human Resources-MD','Maryland');
insert into department
  values('OP-300','Operations-MD','Maryland');
insert into department
  values('SL-300','Sales-MD','Maryland');

create table employee(
  empno varchar(20) primary key,
  fullname varchar(50) not null,
  address varchar(120),
  age int,
  salary numeric(8,2) not null,
  deptno varchar(20) not null,
  constraint fk_employee_department_belong_rltn foreign key(deptno)
    references department(deptno)
);
create unique index undx_employee_fullname on employee(fullname);
```

Another way to declare controls is on the fly:

```
items: [
  { fieldLabel: 'Name', emptyText: 'Please, enter a name', name: 'name' },
  { xtype: 'numberfield', fieldLabel: 'Age', value: '25', name: 'age' }
]
```

As you can see, for the name field you don't have to specify the type because it's taken from the default properties of the form.

I'll add some additional elements to the form, which ends up looking like **Figure 1**.

So far, you've built a form using Ext JS to take data from the user. Now, let's send the data to the server. You'll need to add a button to handle the submit process and show the result to the user, as shown in **Figure 2**.

The buttons property enables the form to manage all the possible actions. Each button has name and handler properties. The handler property contains the logic associated with the action executed on the button. In this case, there are two buttons whose names are Save and Reset. The Save button handler executes a submit action on the form and shows a message indicating success or failure. The Reset button handler resets the field values on the form.

Figure 4 **Site.Master**

```
<head runat="server">
  <title><asp:ContentPlaceHolder ID="TitleContent"
    runat="server" /></title>
  <link href="../../Content/Site.css" rel="stylesheet"
    type="text/css" />

  <!-- Include the Ext JS framework -->
  <link href="<%= Url.Content("~/Scripts/ext-3.2.1/resources/css/ext-all.css") %>"
    rel="stylesheet" type="text/css" />
  <script type="text/javascript"
    src="<%= Url.Content("~/Scripts/ext-3.2.1/adapter/ext/ext-base.js") %>">
  </script>
  <script type="text/javascript"
    src="<%= Url.Content("~/Scripts/ext-3.2.1/ext-all.js") %>">
  </script>
  <!-- Placeholder for custom JS and CSS and JS files
    for each page -->
  <asp:ContentPlaceHolder ID="Scripts" runat="server" />
</head>
```

The last—but important—step in form creation is validation. In order to specify required fields, we need to set the allowBlank property to false and the blankText property to an error message that's displayed when the required validation fails. For example, here's the name field of the form:

```
{ fieldLabel: 'Name', emptyText: 'Please, enter a name', name: 'name',
allowBlank: false }
```

When you run the application and click the Save button without entering any data in the Name and Age fields, then you receive an error message and the required fields are underlined in red.

> The view will present the form to get the data related to one employee.

To customize the error messages on the fields, add the following line of code just under the Ext.onReady function:

```
Ext.QuickTips.init();
```

Now, when the user moves the mouse pointer over the field, a balloon with a message displaying the error is displayed.

I set several validation rules for the fields such as specifying the minimum and maximum length allowed, deferring the field validation until form submission, and creating validation functions for URLs, e-mail addresses, and other types of data. You can see the details of this validation in the code download.

Building the Web App

Now, let's develop a Web solution using Ext JS and ASP.NET MVC. I used ASP.NET MVC 2, but this solution should be applicable to ASP.NET MVC 3 as well. The scenario I'm going to address is adding an employee in a human resources management system.

Figure 5 **Adding the Employee Form**

```
<%@ Page Title="" Language="C#"
  MasterPageFile="~/Views/Shared/Site.Master"
  Inherits="System.Web.Mvc.ViewPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent"
  runat="server">
Index
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent"
  runat="server">
  <h2>Add a New Employee</h2>
  <div id="employeeform"></div>
</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="Scripts"
  runat="server">
  <script type="text/javascript"
    src="<%= Url.Content("~/Scripts/employee_form.js") %>">
  </script>
</asp:Content>
```

Does your Team do more than just track bugs?

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

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

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

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

Native Smart Card Login Support including Government and DOD

New in Team 2.11

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

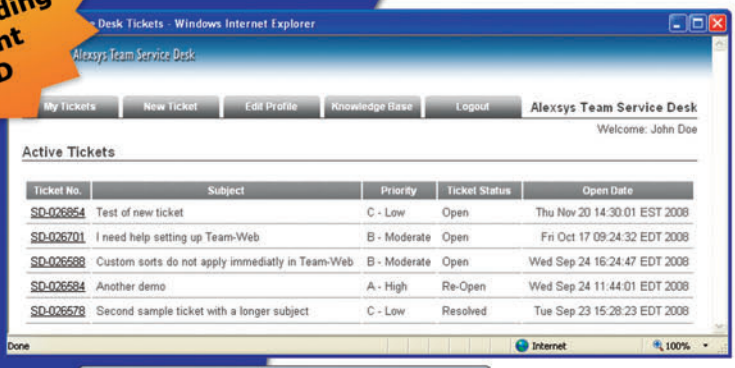
Service Desk Features

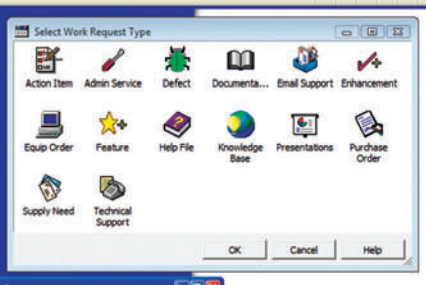
- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

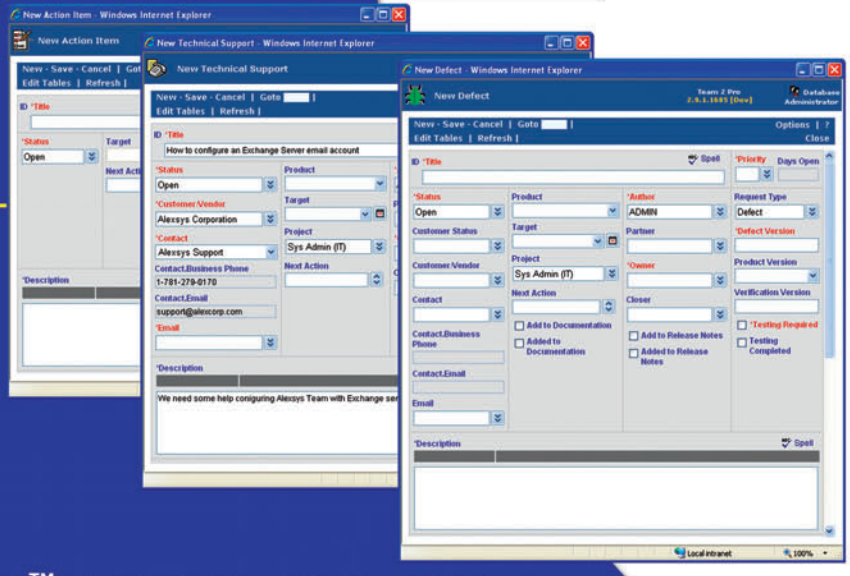
Team 2 Features

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

ZDNet Editors' Pick







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 6 **Form Field Widgets**

```
items: [
  { fieldLabel: 'Employee ID', name: 'empno', allowBlank: false },
  { fieldLabel: 'Fullname', name: 'fullname', allowBlank: false },
  { xtype: 'textarea', fieldLabel: 'Address', name: 'address',
    multiline: true },
  { xtype: 'numberfield', fieldLabel: 'Age', name: 'age' },
  { xtype: 'numberfield', fieldLabel: 'Salary', name: 'salary',
    allowBlank: false },
  { xtype: 'combo', fieldLabel: 'Department', name: 'deptno',
    store: departmentStore, hiddenName: 'deptno',
    displayField: 'deptname', valueField: 'deptno', typeAhead: true,
    mode: 'remote', forceSelection: true, triggerAction: 'all',
    emptyText: 'Please, select a department...', editable: false }
],
```

The Add Employee use case description is as follows: A screen prompts the user to enter valid information for a new employee such as employee identifier, full name, address, age, salary and department. The department field is a list of departments from which to choose.

The main implementation strategy is to create an Ext JS form on the client side—as you've already seen—and then process the data using ASP.NET MVC. The persistence layer will use LINQ to represent business entities and to persist data to the database system. The back-end database is Microsoft SQL Server 2008.

Start by opening Visual Studio 2010 and creating a new project with the ASP.NET MVC 2 Web Application template.

Figure 7 **HumanResourceController**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using HumanResources_ExtJS_ASPNETMVC.Models;

namespace HumanResources_ExtJSASPNETMVC.Models.BusinessObjects {
  public class HumanResourcesController : Controller {
    DepartmentRepository _repoDepartment = new DepartmentRepository();
    EmployeeRepository _repoEmployee = new EmployeeRepository();

    // GET: /HumanResources/
    public ActionResult Index() {
      return View();
    }

    // POST: /HumanResource/Departments
    [HttpPost]
    public ActionResult Departments() {
      var arrDepartment = this._repoDepartment.FindAll();
      var results = (new {
        departments = arrDepartment
      });
      return Json(results);
    }

    // POST: /HumanResource/AddEmployee
    [HttpPost]
    public ActionResult AddEmployee(employee employee) {
      string strResponse = String.Empty;
      try {
        this._repoEmployee.Create(employee);
        strResponse = "{success: true}";
      }
      catch {
        strResponse = "{success: false, error: \"An error occurred\"}";
      }
      return Content(strResponse);
    }
  }
}
```

Next, create the database schema. For this example, the schema will contain two entities: employee and department. **Figure 3** shows how I created the Human Resources database and the underlying tables and constraints.

Now let's use LINQ to SQL to define the structure of the entities and the persistence mechanism. Start by creating an EmployeeRepository class to manage the data-access logic to the employee table. In this case, you only need to implement the create operation:

```
public class EmployeeRepository {
  private HumanResourcesDataContext _ctxHumanResources =
    new HumanResourcesDataContext();

  public void Create(employee employee) {
    this._ctxHumanResources.employees.InsertOnSubmit(employee);
    this._ctxHumanResources.SubmitChanges();
  }
}
```

You also need a DepartmentRepository class to manage the data-access logic to the department table. Again, in this simple case you only need to implement the read operation in order to find a list of departments:

```
public class DepartmentRepository {
  private HumanResourcesDataContext _ctxHumanResources =
    new HumanResourcesDataContext();

  public IQueryable<department> FindAll() {
    return from dept in this._ctxHumanResources.departments
           orderby dept.deptname
           select dept;
  }
}
```

Now let's define another important piece of the architecture: the controller. To define a controller, right-click on the Controllers folder in the Solution Explorer window and select Add | Controller. I used HumanResourcesController as the controller name.

Ext JS Presentation Layer

Now let's go back to Ext JS and use the framework to build the presentation layer of the application. For this solution, you only need to import ext-all.js and the \adapter and \resources folders.

Go to the Site.Master page and add references to the Ext JS files inside the head element, as well as an <asp:ContentPlaceHolder> tag element as a container of the customized JavaScript and CSS code for each page, as shown in **Figure 4**.

Now let's add the other important pieces of the MVC architecture: the view. The view will present the form to get the data related to one employee. Go to HumanResourcesController, right-click on the Index action method and select Add View. Click the Add button in the Add View dialog box.

To implement the Ext JS form created earlier in the article, you need to add a JavaScript file to the Scripts directory and a reference to this JavaScript file in the view. Then include the reference to the employee_form.js file and add a div element into the Index.aspx view (see **Figure 5**).

Go to the employee_form.js file and add some code to configure the ExtJS form and its underlying widgets. The first step is to define an instance of Ext.data.JsonStore class to get a list of departments:

```
var departmentStore = new Ext.data.JsonStore({
  url: 'humanresources/departments',
  root: 'departments',
  fields: ['deptno', 'deptname']
});
```

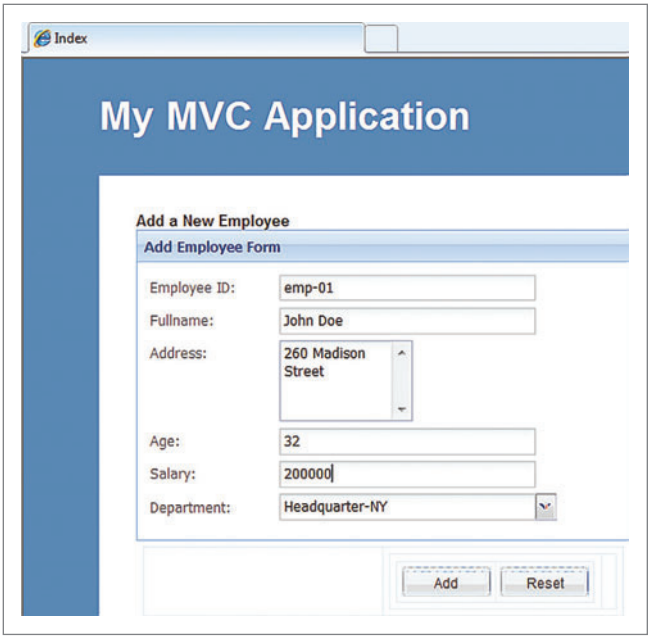


Figure 8 **Running the Application**

The url property points to the departments action method on the HumanResourceController controller. This method is accessed by the HTTP POST verb. The root property is the root element of the list of departments. The fields property specifies the data fields. Now define the form. The properties are self-descriptive:

```
var form = new Ext.FormPanel({
  title: 'Add Employee Form',
  renderTo: 'employeeform',
  width: 400,
  url: 'humanresources/addemployee',
  defaults: { xtype: 'textfield' },
  bodyStyle: 'padding: 10px',
```

In this case, the url property points to the AddEmployee action method on the HumanResourceController controller. This method is also accessed using HTTP POST.

The items property provides the list of widgets representing the fields of the form (**Figure 6**). Here the default widget is a text field (this is specified in the defaults property). The first field is employee number, which is required (specified by the allowBlank property). The second field is the full name, which is also a required text field. The address field is an optional text area. The age field is an optional number field. The salary field is a required number field. And finally, the department number field is an identifier string, which is selected from a list of departments.

Finally, the buttons property is defined to handle the actions over the form. This is configured just like **Figure 2**, but the text property has the value "Add."

Now the employee_form.js file is complete. (I've gone through most of the elements of the file here. See the code download for the complete source code listing for this file.)

Now let's go to HumanResourceController and implement the corresponding action methods, as shown in **Figure 7**.

That's It!

Now run the solution. You'll see the Web page shown in **Figure 8**. Enter some data in the form and then click Add. You'll see a confirmation message box. You'll also see the row inserted in the dbo.employee table on the database.

That's really all there is to creating a simple RIA. Depending on the features you want to leverage, a similar application could be built with any of the other popular JavaScript frameworks while still employing ASP.NET MVC. You could easily substitute Entity Framework for the data layer, and use Windows Azure storage or SQL Azure as the back-end data store. These simple building blocks make building a basic data-centric RIA quick and easy. ■

JUAN CARLOS OLAMENDY *is a senior architect, developer and consultant. He has been recognized as a Microsoft Most Valuable Professional (MVP) and Oracle ACE several times. He is Microsoft Certified Technology Specialist in Windows Communication Foundation. You can contact Olamendy at johnx_olam@fastmail.*

THANKS to the following technical experts for reviewing this article: *Scott Hanselman and Eilon Lipton*

Building and Using Custom OutputCache Providers in ASP.NET

Brandon Satrom

If you're a Web developer, in the past you may have utilized the output-caching facility provided by ASP.NET. Introduced with the first version of the Microsoft .NET Framework, ASP.NET output caching can improve the performance of serving content to site visitors by retrieving that content from a cache, bypassing re-execution of pages or controllers. This saves your application expensive database calls when returning data that doesn't update frequently, or that can be stale for periods of time.

This article discusses a prerelease version of Windows Azure AppFabric SDK 2.0. All information is subject to change.

This article discusses:

- Output caching in ASP.NET
- Extensible output caching in ASP.NET
- NoSQL, document databases and MongoDB
- Building a custom OutputCacheProvider using MongoDB
- Using the MongoDB OutputCacheProvider in ASP.NET MVC
- Using the Windows Azure AppFabric DistrubutedCache provider

Technologies discussed:

ASP.NET 4, Windows Azure, MongoDB

Code download available at:

code.msdn.microsoft.com/mag201103OutputCache

The ASP.NET output cache uses an in-memory storage mechanism and, until the .NET Framework 4, it wasn't possible to override or replace the default cache with your own implementation. With the new OutputCacheProvider type, it's now possible to implement your own mechanism for caching page output in ASP.NET.

In this article, I'll discuss two such custom mechanisms. First, using MongoDB—a popular document-oriented database—I'll create my own provider to facilitate output caching in a simple ASP.NET MVC application. Then, using the same application, I'll quickly swap out my custom provider to leverage features of Windows Azure AppFabric—specifically, the new DistributedCache provider that leverages Windows Azure infrastructure to provide a distributed, in-memory cache in the cloud.

Output Caching in ASP.NET

In ASP.NET Web Forms applications, output caching can be configured by adding an OutputCache Page directive to any ASP.NET page or user control:

```
<%@ OutputCache Duration="60" Location="Any" VaryByParam="name" %>
```

For ASP.NET MVC applications, output caching is available using an action filter that ships with ASP.NET MVC, and which can be leveraged as an attribute on any controller action:

```
[OutputCache(Duration=60, VaryByParam="none")]
```

"Duration" and "VaryByParam" are required in ASP.NET MVC 1 and 2 applications (VaryByParam is optional in ASP.NET MVC 3), and both mechanisms provide several other attributes and parameters

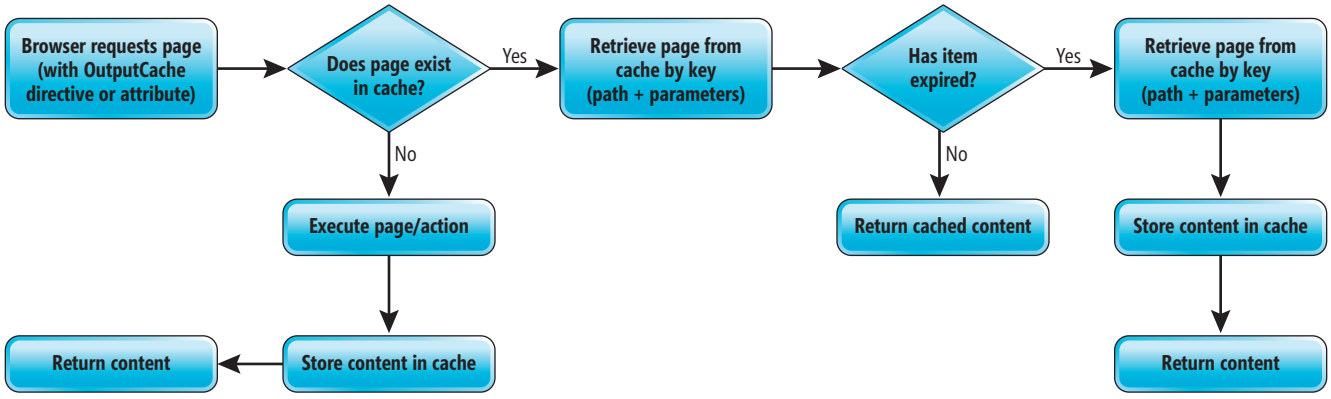


Figure 1 **The ASP.NET Output Caching Process**

that enable developers to control how content is cached (several VaryByX parameters), where it's cached (Location) and capabilities for setting cache invalidation dependencies (SqlDependency).

For traditional output caching, nothing else is needed to implement this functionality in your applications. The OutputCache type is an HttpModule that runs when your application starts and goes to work when a page directive or action filter is encountered. Upon the first request of the page or controller in question, ASP.NET will take the resulting content (HTML, CSS, JavaScript files and so on) and place each item in an in-memory cache, along with an expiration and a key to identify that item. The expiration is determined by the Duration property, and the key is determined by a combination of the path to the page and any necessary VaryBy values—for example, query string or parameter values if the VaryByParam property is provided. So, consider a controller action defined in this manner:

```
[OutputCache(Duration=20, VaryByParam="vendorState")]
Public ActionResult GetVendorList(string vendorState)
{
  // Action logic here.
}
```

In this case, ASP.NET will cache an instance of the resulting HTML view for each occurrence of vendorState (for example, one for Texas, one for Washington and so on) as that state is requested. The key by which each instance is stored, in this case, will be a combination of the path and the vendorState in question.

If, on the other hand, the VaryByParam property is set to "none," ASP.NET will cache the result of the first execution of GetVendorList and will deliver the same cached version to all subsequent requests, regardless of the value of the vendorState parameter passed into that action. The key—by which this instance is stored when no VaryByParam value is provided—would just be the path. A simplified view of this process is depicted in **Figure 1**.

Beyond the Duration parameter—used to control the life of the item in the cache—and a handful of VaryBy parameters (VaryByParam, VaryByHeader, VaryByCustom, VaryByControl and VaryByContentEncoding) used to control the granularity of cached items, the output cache can be configured to control the location of cached content (client, server or downstream proxy). In addition, ASP.NET 2.0 introduced a SqlDependency attribute, which allows developers to specify database tables that a page or control depends upon so that, in addition to time expiration, updates to your underlying source data can also cause cached items to expire.

Although the .NET Framework 2.0 and 3.0 introduced several enhancements to the default cache provider, the provider itself remained the same: an in-memory store, with no extension points or way to provide your own implementation. The in-memory cache is a perfectly acceptable option in most cases, but can, at times, contribute to diminished site performance as server resources are maxed out and memory becomes scarce. What's more, the default caching provider mechanism automatically evicts cached resources—regardless of specified duration—when memory does become scarce, which leaves the developer with little control over how cached resources are managed.

Extensible Output Caching in ASP.NET

The release of the .NET Framework 4 introduced a new facility that enables developers to create their own output cache providers and easily plug those providers into new or existing applications with only minor changes to the application and its configuration. These providers are free to use whatever storage mechanism for cached information that they choose, such as local disks, relational and non-relational databases, the cloud or even distributed caching engines such as that provided in Windows Server AppFabric. It's even possible to use multiple providers for different pages in the same application.

> It's now possible to implement your own mechanism for caching page output in ASP.NET.

Creating your own output cache provider is as simple as creating a new class that derives from the new System.Web.Caching.OutputCacheProvider abstract class and overriding the four methods that ASP.NET requires to work with cached items. The framework definition for the OutputCacheProvider class is listed here (see bit.ly/fozTLc for more information):

```
public abstract class OutputCacheProvider : ProviderBase
{
  public abstract object Get(string key);
  public abstract object Add(string key, object entry, DateTime utcExpiry);
  public abstract void Set(string key, object entry, DateTime utcExpiry);
  public abstract void Remove(string key);
}
```

Figure 2 **A Starter OutputCacheProvider Class**

```
public class DocumentDatabaseOutputCacheProvider : OutputCacheProvider
{
  readonly Mongo _mongo;
  readonly IMongoCollection<CacheItem> _cacheItems;

  public override object Get(string key)
  {
    return null;
  }

  public override object Add(string key, object entry, DateTime utcExpiry)
  {
    return null;
  }

  public override void Set(string key, object entry, DateTime utcExpiry)
  {
    return;
  }

  public override void Remove(string key)
  {
    return;
  }
}
```

Once you've implemented these four methods, all that's left is to add the new provider to your web.config, specify it as the default and add an OutputCache directive or attribute to your application. I'll cover these steps in detail as I walk through the creation of our own output cache provider that uses a document database called MongoDB. But first, it may be helpful to introduce a little context around the tool we'll be using to build our custom provider.

NoSQL, Document Databases and MongoDB

For much of the past few decades, the preferred application storage mechanism has been the relational database management system (RDBMS), which stores data and relationships in tables. SQL Server and Oracle are examples of RDBMSes, as are most of the popular commercial and open source databases currently in use.

However, not all problems requiring storage fit into the same transactional mold. In the late '90s, as the Internet expanded and many sites grew to manage large volumes of data, it became obvious that the relational model provided less-than-ideal performance on certain types of data-intensive applications. Examples include indexing large volumes of documents, delivering Web pages on high-traffic sites or streaming media to consumers.

Many companies addressed their growing storage needs by turning to NoSQL databases, a class of lightweight database that doesn't expose a SQL interface, fixed schemas or pre-defined relationships. NoSQL databases are used heavily by companies such as Google Inc. (BigTable), Amazon.com Inc. (Dynamo) and Facebook (which has a store of more than 50TB for inbox searches) and are experiencing steady growth in popularity and use.

It's important to note that, while some have used the term NoSQL as a rallying cry to call for the abandonment of all RDBMSes, others emphasize the value of utilizing both types of storage. NoSQL databases were conceived to solve a class of problem that RDBMSes couldn't—not to replace these systems outright. The discriminating developer would be wise to understand both systems and utilize each where appropriate, even at times mixing both types of storage in a single application.

One situation well-suited for a NoSQL database is output caching. NoSQL databases are ideal for working with transient or temporary data, and cached pages from an ASP.NET application certainly fit that bill. One popular NoSQL option is MongoDB (mongodb.org), a document-oriented NoSQL database used by Shutterfly, Foursquare, *The New York Times* and many others. MongoDB is a fully open source database written in C++, with drivers for nearly every major programming language, C# included. We'll use MongoDB as the storage mechanism for our custom output cache provider.

Building a Custom OutputCacheProvider Using MongoDB

To get started, you'll want to go to mongodb.org to download and install the tool. The documents at mongodb.org/display/DOCS/Quickstart should tell you everything you need to know to install MongoDB on Windows, Mac OS X and Unix. Once you've downloaded MongoDB and tested things out with the shell, I recommend installing the database as a service using the following command from the installation directory (be sure to run cmd.exe as an administrator):

```
C:\Tools\MongoDB\bin>mongod.exe --logpath C:\Tools\MongoDB\Logs
--directoryperdb --install
```

MongoDB will install itself as a service on your computer and will use C:\Data\db as the default directory for all its databases. The option --diretoryperdb tells MongoDB to create a root directory for every database you create.

After running the previous command, type the following to start the service:

```
net start MongoDB
```

Once you have things up and running, you'll need to install a driver library to work with MongoDB in .NET. There are several options available; I'll be using the mongodb-csharp driver created by Sam Corder (github.com/samus/mongodb-csharp).

We have MongoDB installed, and we have a driver that we can use within a .NET application, so now it's time to create our custom output cache provider. To do this, I created a new class library called DocumentCache and added two classes: DocumentDatabaseOutputCacheProvider and CacheItem.

The first is my provider, a public class that subclasses the abstract OutputCacheProvider. The beginning implementation is depicted in **Figure 2**.

Figure 3 **Implementing the Add Method**

```
public override object Add(string key, object entry, DateTime utcExpiry)
{
  key = MD5(key);
  var item = _cacheItems.FindOne(new { _id = key });
  if (item != null) {
    if (item.Expiration.ToUniversalTime() <= DateTime.UtcNow) {
      _cacheItems.Remove(item);
    } else {
      return Deserialize(item.Item);
    }
  }

  _cacheItems.Insert(new CacheItem
  {
    Id = key,
    Item = Serialize(entry),
    Expiration = utcExpiry
  });

  return entry
}
```

Notice that the second private variable in **Figure 2** references CacheItem, the other class I need to create in my project. CacheItem exists to contain the relevant details that my output cache provider needs to work with both ASP.NET and my database, but it isn't an object needed external to my provider. As such, I define CacheItem as an internal class, as shown here:

```
[Serializable]
internal class CacheItem
{
  public string Id { get; set; }
  public byte[] Item { get; set; }
  public DateTime Expiration { get; set; }
}
```

Id maps to the key provided to me by ASP.NET. You'll recall that the key is a combination of the path and any VaryBy conditions defined in your page directive or action attribute. The Expiration field corresponds to the Duration parameter, and the Item property is the item to be cached.

We'll start implementing our provider by setting things up in the constructor of our DocumentDatabaseOutputCacheProvider class. Because we know that ASP.NET maintains a single instance of our provider for the entire life of the application, we can perform some setup work in our constructor, like this:

```
readonly Mongo _mongo;
readonly IMongoCollection<CacheItem> _cacheItems;

public DocumentDatabaseOutputCacheProvider()
{
  _mongo = new Mongo();
  _mongo.Connect();

  var store = _mongo.GetDatabase("OutputCacheDB");
  _cacheItems = store.GetCollection<CacheItem>();
}
```

The constructor creates a new instance of the Mongo type and connects to the server using the default location (localhost). It then asks MongoDB for the OutputCacheDB database and for an IMongoCollection of our CacheItem type. Because MongoDB is a schema-less database, creating databases on the fly is supported. Your first call to _mongo.GetDatabase("OutputCacheDB") will return an instance of a new database, and that database will be created on disk when the first insert occurs.

Now let's implement the Add method, as shown in **Figure 3**.

The first thing I do in each method is call the MD5 method on the passed-in key. This method—omitted for brevity, but which is available in the online source code download—generates a database-friendly MD5 hash based on the key that ASP.NET provides to me. Then, I call my IMongoCollection<CacheItem> type, _cacheItems, to query the underlying database for the key in question. Notice the anonymous type (new { _id = key}) passed into the FindOne method. Querying MongoDB is primarily done via selector objects or template documents that specify one or more fields in a document to match in the database. _id is the key that MongoDB uses to store documents, and—by convention of the driver I'm using—that property is automatically mapped to the Id property of my CacheItem class. So when I save a new cache item, as you see in the _cacheItems.Insert method shown in **Figure 3**, the key is assigned using the Id property, which MongoDB uses to populate the internal _id field of the record. MongoDB is a key-value store, so each CacheItem object is stored using binary-serialized JSON that looks like the following:

Figure 4 **Implementing the Get Method**

```
public override object Get(string key)
{
  key = MD5(key);
  var cacheItem = _cacheItems.FindOne(new { _id = key });

  if (cacheItem != null) {
    if (cacheItem.Expiration.ToUniversalTime() <= DateTime.UtcNow) {
       _cacheItems.Remove(cacheItem);
     } else {
       return Deserialize(cacheItem.Item);
    }
  }

  return null;
}
```

```
{ "_id" : ObjectId(Id), "CacheItem": new CacheItem { Id = key, Item =
entry, Expiration = utcExpiry } }
```

If I find a CacheItem with the same key as the one passed in, I check the expiration of that item against the current UTC time. If the item hasn't expired, I binary deserialize it using a private method (available in the online source code) and return the existing item. Otherwise, I insert a new item into my store, binary serialize it and return the passed-in entry.

Once I've implemented adding items to the cache, I can add the Get method, which will find and return a cached item by key (or null if a result isn't found) as shown in **Figure 4**.

As with the Add method, the Get method also checks the expiration of the item if it exists in the database and, if it has expired, removes it and returns null. If the item exists and hasn't expired, it's returned.

Now, let's implement the Remove method, which accepts a key and removes the item matching that key from the database, as shown here:

```
public override void Remove(string key)
{
  key = MD5(key);
  _cacheItems.Remove(new { _id = key });
}
```

Just as with the code our driver uses to get a database that doesn't yet exist, MongoDB doesn't complain if we attempt to remove an item that isn't found in our database. It simply does nothing.

According to our abstract base class, there's still one final method we need to implement to have a functional custom output cache provider, the Set method. I've included it in **Figure 5**.

Figure 5 **Implementing the Set Method Public Override Void Set(string key, object entry, DateTime utcExpiry)**

```
{
  key = MD5(key);
  var item = _cacheItems.FindOne(new { _id = key });

  if (item != null)
  {
    item.Item = Serialize(entry);
    item.Expiration = utcExpiry;
    _cacheItems.Save(item);
  }
  else
  {
    _cacheItems.Insert(new CacheItem
    {
      Id = key,
      Item = Serialize(entry),
      Expiration = utcExpiry
    });
  }
}
```

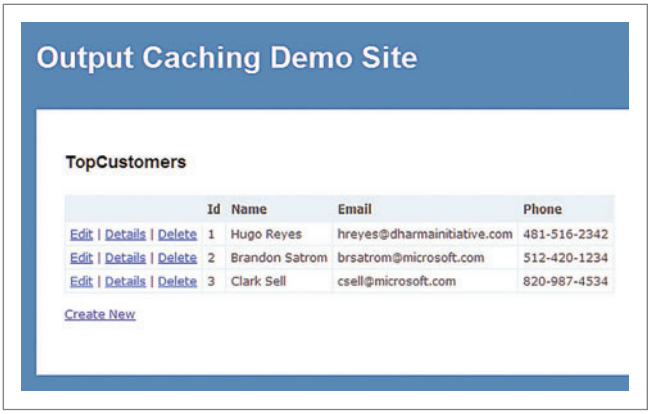


Figure 6 **The Cached TopCustomers View**

At a glance, it may seem that the Add and Set methods are identical, but there's a key difference between their intended implementation. According to the MSDN Library documents on the OutputCacheProvider class (bit.ly/fozTLc), the Add method of a custom provider should look for a value in the cache that matches the specified key and, if it exists, do nothing to the cache and return the saved item. If that item doesn't exist, Add should insert it.

The Set method, on the other hand, should always put its value into the cache, inserting the item if it doesn't exist and overwriting it if it does. You'll notice, in **Figure 3** for Add and **Figure 5** for Set, that these methods behave as specified.

With those four methods implemented, we're now ready to put our provider to work.

Using the MongoDB OutputCacheProvider in ASP.NET MVC

Once we've compiled our custom provider, we can add that provider to any ASP.NET application with a few lines of configuration. After adding a reference to the assembly that contains the provider, add the following text to your web.config file in the <system.web> section:

```
<caching>
  <outputCache defaultProvider="DocumentDBCache">
    <providers>
      <add name="DocumentDBCache"
       type="DocumentCache.DocumentDatabaseOutputCacheProvider, DocumentCache" />
    </providers>
  </outputCache>
</caching>
```

The <providers> element defines all of the custom providers you want to add to your application and defines a name and type for each. Because you can have multiple custom providers in a single application, you'll also want to specify the defaultProvider attribute, as I do in the preceding code snippet.

My sample application is a simple ASP.NET MVC site with a CustomersController. In that controller is an action called TopCustomers, which returns a list of the top customers for my business. This information is the result of complex calculations and several database queries in my SQL Server database and is only updated once an hour. For these reasons, it's an ideal candidate for caching. So I add an OutputCache attribute to my action, like so:

```
[OutputCache(Duration = 3600, VaryByParam = "none")]
public ActionResult TopCustomers()
{
  var topCustomers = _repository.GetTopCustomers();
  return View(topCustomers);
}
```

Now, if I run the site and navigate to my TopCustomers page, my custom provider will roll into action. First, my Get method will be called, but because this page isn't yet cached, nothing will be returned. The controller action will then execute and return the TopCustomers view, as depicted in **Figure 6**.

ASP.NET will then call my custom cache provider, executing the Set method, and the item will be cached. I've set the duration to 3,600 seconds—or 60 minutes—and every subsequent request for that time period will use the cached item returned by my Get method, bypassing re-execution of my Controller Action. If any underlying data is changed, updates will be reflected on the first execution after expiration, and that new information will then be cached for an hour. If you want to see MongoDB in action, you have a couple of options. You can open your browser and navigate to http://localhost:28107/, which displays the log, as well as recent queries and statistics about your database. Or, you can run mongo.exe from the bin directory of your MongoDB installation and query your database via the Mongo Shell. For more information about using these tools, see mongodb.org.

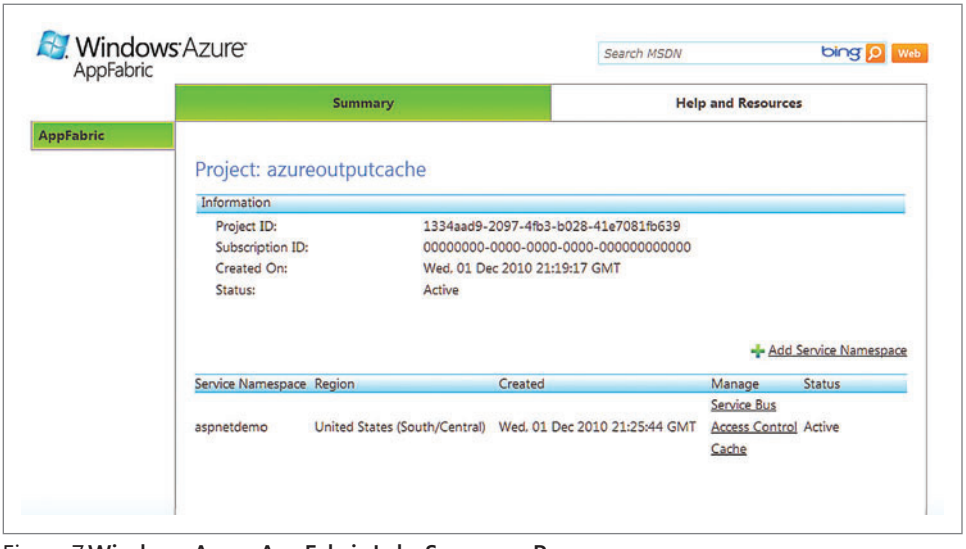


Figure 7 **Windows Azure AppFabric Labs Summary Page**

Using the DistributedCache Provider

So what if everything I've discussed so far is more than you'd care to dive into? Perhaps you want to leverage an alternative caching mechanism, but you have neither the time nor the desire to roll your own? You'll be happy to know that, since the introduction of extensible output caching, many

alternatives—commercial, open source and provided by Microsoft—are either available or in development. One such example is a cloud-based, distributed in-memory cache: the DistributedCache provider currently available as a part of Windows Azure AppFabric. If you're already building cloud-based applications, Windows Azure AppFabric Caching can speed up access to data for those applications and, because caching is delivered as a cloud service, the setup is simple and requires no overhead to maintain.

At the time of this writing, AppFabric Caching is part of the Windows Azure AppFabric Community Technology Preview October Release, so you can access caching features without an active Windows Azure account. However, if you're an MSDN subscriber, I highly recommend activating your Windows Azure benefits at windows.azure.com. Go to portal.appfabriclabs.com and create an account to use the developer preview features.

Once you've created a Labs account, click on the Add Service Namespace link to enable AppFabric services (see **Figure 7**).

After you've set up your service namespace, click on the Cache link, and take note of the service URL and authentication token listed in the cache section (see **Figure 8**). You'll need this information to configure your application to use the DistributedCache provider.

Next, you'll need to download and install the Windows Azure AppFabric SDK (click the download link on the Cache page in the portal). After the installation is complete, you're ready to configure Windows Azure AppFabric Caching for your application.

You'll need to add references to several assemblies that the SDK installation placed on your machine. Using the same ASP.NET MVC application you used for your custom Document database cache, navigate to the SDK install location (default is C:\Program Files*\Windows Azure AppFabric SDK\V2.0\Assemblies\Cache) and add references to each assembly contained within.

Once you've done that, open your web.config and add the following <configSections> entry, keeping any existing configuration sections:

```
<configSections>
  <section name="dataCacheClient"
    type="Microsoft.ApplicationServer.Caching.DataCacheClientSection,
          Microsoft.ApplicationServer.Caching.Core"
    allowLocation="true" allowDefinition="Everywhere"/>
</configSections>
```

Next, create the <dataCacheClient> section, replacing the <host> name, cachePort and <messageSecurity> authorizationInfo properties with details from your portal account, like so:

```
<dataCacheClient deployment="Simple">
  <hosts>
    <host name="yournamespace.cache.appfabriclabs.com" cachePort="your port" />
  </hosts>
  <securityProperties mode="Message">
    <messageSecurity authorizationInfo="your authentication token">
    </messageSecurity>
  </securityProperties>
</dataCacheClient>
```

Then, find the <caching> section under <system.web> and add the following provider entry after the entry you created for your custom provider:

```
<add name="DistributedCache"
    type="Microsoft.Web.DistributedCache.DistributedCacheOutputCacheProvider,
          Microsoft.Web.DistributedCache"
    cacheName="default" />
```

Finally, change the defaultProvider attribute on the <outputCache> element to "DistributedCache." The DistributedCacheOutputCacheProvider is a subclass type of the abstract OutputCacheProvider, just like our MongoDB implementation. Now, build and run your application and navigate to the Top Customers page. Try adding a customer while the list is still cached and notice that, as with our MongoDB implementation, the list will remain cached as long as you specify.

Wrapping Up

In this article, I discussed ASP.NET output caching, classical uses of the default in-memory cache, and new extensible caching facilities provided using the OutputCacheProvider abstract class in the .NET Framework 4. I talked about NoSQL and document databases and how these types of systems are ideal for working with transient data, such as cached output. We used MongoDB to build a sample output cache and used that within an ASP.NET MVC application. Finally, we moved our output cache to the cloud, and with minor setup and configuration and no code changes whatsoever, we were able to swap out caching mechanisms in our application.

Extensible output caching is just one of the many great new features in ASP.NET 4, and I hope this exploration of the feature—and of the technologies that can be leveraged along with it—has been useful. To learn more about MongoDB, go to mongodb.org. To learn more about Windows Azure AppFabric, go to portal.appfabriclabs.com/helpandresources.aspx. ■

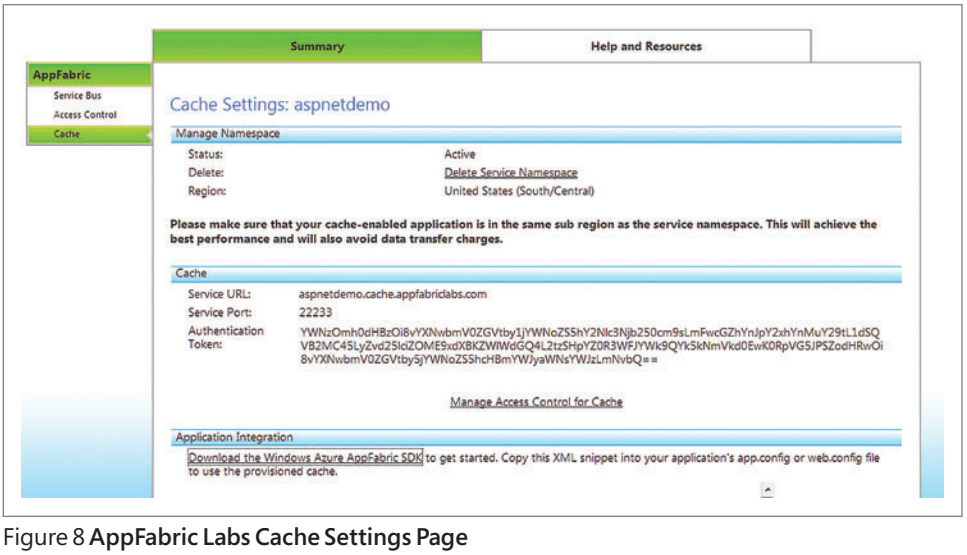


Figure 8 **AppFabric Labs Cache Settings Page**

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

THANKS to the following technical experts for reviewing this article: *Brian H. Prince and Clark Sell*

YOCHAY KIRIATY AND JAIME RODRIGUEZ

Windows Phone Navigation: The Basics

Silverlight Windows Phone applications have a Web-like page model where the end users navigate from one page to another. There's a dedicated hardware Back button to easily navigate back to previous pages (without consuming screen real estate), and the journaling (or history) of your navigation is integrated with the platform to ease navigating or transitioning across different applications. This two-part article will:

- Introduce you to the page navigation model on Windows Phone.
- Provide the best practices you'll need to get the most out of the current APIs–such as integration with the hardware Back button, optimized loading and unloading of pages, and ensuring your navigation model meets Windows Phone certification guidelines.
- Introduce actionable, easy-to-follow recipes to create the most complex navigations not implemented with the current APIs, including transient content and page transitions.

Windows Phone Nav Model

The Windows Phone navigation model consists of a frame (PhoneApplicationFrame) and one or more pages (PhoneApplicationPage) that hold the content loaded into the frame.

The PhoneApplicationFrame exposes most of the navigation events and exposes the Navigate method you'll use to go across pages. It also determines the client area for the application and reserves the space for the application bar and the system tray.

PhoneApplicationPage has page-specific notifications for when a page is navigated to and when a user navigates away from a page. It also handles the events related to the hardware Back button.

Both PhoneApplicationFrame and PhoneApplicationPage share a NavigationService; this service is actually doing the navigation. Windows Phone supports journaling (tracking the history of the pages you've loaded so you can go back to a previous page) and exposes APIs so you can go back. Forward navigation isn't supported on the phone.

Windows Phone has three dedicated hardware buttons: Back, Start and Search. There are specific application-certification requirements around handling of the hardware Back button:

- An application shouldn't prevent the user from going back to a previous page. The only possible exception is a prompt when data loss is involved–you can then prompt to confirm and let the user through if he chooses to navigate back.
- If a popup, the Software Input Panel (SIP) or other transient dialog is open, pressing the hardware Back button should dismiss this dialog but not leave the current page (effectively canceling the Back button navigation).

Figure 1 **Instantiation of RootFrame in 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 PhoneApplicationFrame();
  RootFrame.Navigated += CompleteInitializePhoneApplication;


  // Handle navigation failures
  RootFrame.NavigationFailed += RootFrame_NavigationFailed;

  // Ensure we don't initialize again
  phoneApplicationInitialized = true;
}
```

- Pressing the Back button while in the first screen of an application must exit the application. This functionality comes for free. If you don't prevent the navigation, the framework exits for you—in fact, this is the only way to exit a Silverlight application. There's no Exit method in the exposed APIs.
- To maintain a consistent user experience (UX) across apps, the Back button should only be used for backward navigation.

In addition to the Back button's critical role in navigation, the Start button also participates in navigation. When the user presses the Start button, the running application is deactivated and a context switch is performed as you navigate forward to the Start menu. From here a user can launch another application and navigate within the new application, or he can choose to navigate back (using the hardware Back button) to the previously running application. This effectively creates a navigation model where the Back button navigates through the pages of a running application or through the stack of previously running application.

Windows Phone APIs

As noted earlier, the core players for navigation are PhoneApplicationFrame and PhoneApplicationPage.

PhoneApplicationFrame acts as the RootVisual for the application. At startup, a PhoneApplicationFrame is instantiated in the App class in App.xaml.cs (see **Figure 1**).

The runtime automatically navigates to the instance of PhoneApplicationPage, which is specified by the NavigationPage

Code download available at code.msdn.microsoft.com/mag201103Mobile.

Figure 2 **PhoneApplicationFrame Methods, Properties and Events**

Name	Type	Description
Navigate	Method	Navigates to a new PhoneApplicationPage specified by the URI parameter. The parameter is a Uri, so a Navigate call effectively instantiates the new page and navigates to it (you don't pass it an already instantiated page).
CanGoBack	Read-Only Property	Returns true if the application's back stack (the journaling history) isn't empty. This means users have navigated forward at least once within the app. If the application is in the first page loaded in the app, CanGoBack will return false and you won't be able to programmatically call GoBack, but the end user can still press the hardware Back button and the application will exit because it's going back to the previously running application.
CanGoForward	Read-Only Property	Not applicable to Windows Phone. It's always false because forward navigation isn't supported.
UriMapper	Property	Gets/Sets a UriMapper. Beyond our scope for this article, but worth mentioning that Uri mapping is supported.
GoBack	Method	Navigates to the most recent entry in the back stack. This method will throw an exception if there's no entry in the back stack; always check CanGoForward before calling this method.
GoForward	Method	Not supported on Windows Phone; will throw InvalidOperationException
Navigating	Event	Occurs when a new navigation is requested. At this point, it can still be canceled by setting the Cancel property in the NavigatingCancelEventArgs parameter to true. Please see notes later on about why you shouldn't cancel back navigations in this event.
Navigated	Event	Occurs when a navigation has been executed. This doesn't mean that the page content that was navigated to has been loaded. It simply occurs when the content has been found and navigated to.
NavigationFailed	Event	Occurs when an error has been encountered.
NavigationStopped	Event	Occurs when navigation is stopped by either calling the StopLoading method or more commonly when a new navigation is requested and a navigation was in progress.

attribute in the DefaultTask on the WMAppManifest.xml application manifest, as shown here:

```
<Tasks>
  <DefaultTask  Name ="_default" NavigationPage="MainPage.xaml"/>
</Tasks>
```

Getting a bit closer to responsibilities and APIs, PhoneApplicationFrame exposes most of the navigation methods and events we'll need for this article. **Figure 2** lists the most relevant methods, properties and events in PhoneApplicationFrame.

Most are inherited from Frame, so for those familiar with the Silverlight Frame class, these methods should look familiar. The list in **Figure 2** isn't inclusive of all PhoneApplicationFrame features, just the relevant ones for navigation.

Code Walk-Through: To see all these events and properties in action, explore AllNavigationsEvents.xaml.cs in the sample code accompanying this article. You can also see the order in which events fire in **Figure 3**.

PhoneApplicationFrame also determines the client area that the application will get and reserves the space for the application bar and the system tray. This detail will be relevant as we navigate across pages that have an application bar because it's specified at the page level, and there's a system-wide animation to show and hide the application bar as a page gets loaded.

The second participant in the navigation is PhoneApplicationPage. It plays two critical roles in navigation:

- Handling of the hardware Back button presses.
- Providing events for page lifecycle to know when a page is activated/deactivated.

For integration with the hardware Back button, PhoneApplicationPage exposes a BackKeyPress event. The page also has a virtual OnBackKeyPress method you can override in your instance of a page to handle and even cancel a Back button press event.

PhoneApplicationFrame has a Navigating event and an OnNavigatingFrom notification/callback. Within both of these, you can cancel navigations to other pages within the app by setting e.Cancel = true in the NavigationCancelEventArgs parameter passed to these methods; due to a known bug on the platform, you shouldn't cancel Back button navigations from these events/methods. If you do

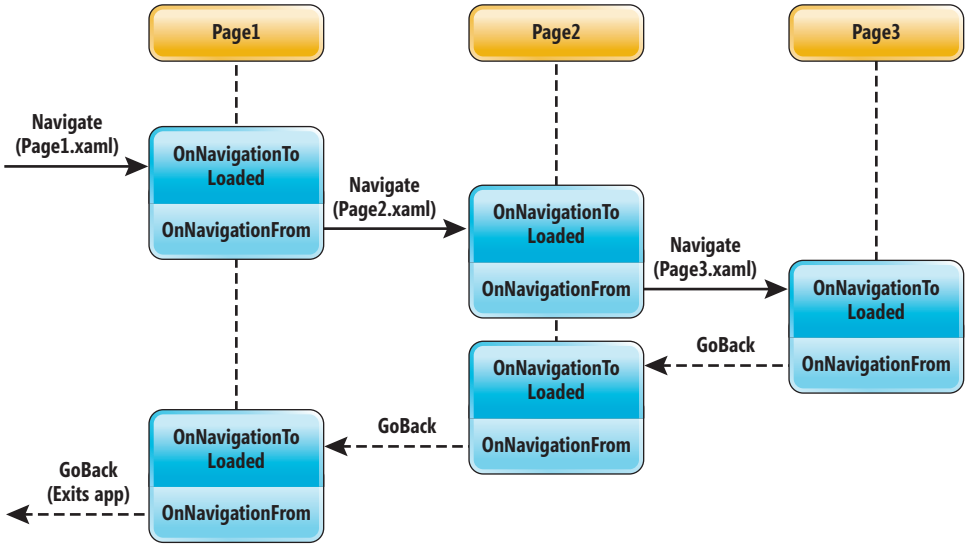


Figure 3 **The Sequence of Events as You Navigate Across Pages**

Figure 4 **Events and Methods Where Navigations Can Be Canceled**

Owner	Event/Notification	Can Cancel a New Navigation	Can Cancel Back Navigations	Check for Back Navigations
PhoneApplicationFrame	Navigating	Yes	No	Yes; check e.NavigationMode != NavigationMode.Back
PhoneApplicationPage	OnNavigatingFrom	Yes	No	Yes; check e.NavigationMode != NavigationMode.Back
PhoneApplicationPage	OnBackKeyPress	No (called only when Back key press is called)	Yes	Not needed; only called on hardware Back key press
PhoneApplicationPage	BackKeyPress (event)	No (called only when Back key press is called)	Yes	Not needed; only called on hardware Back key press

cancel a hardware Back button press in this event, your navigation will break and the application will need to be restarted. The only two recommended methods for canceling a hardware Back button press are the PhoneApplicationPage BackKeyPress event and the OnBackKeyPress callback.

See **Figure 4** for a list of events and methods where navigations can be canceled, with recommendations on whether a Back press can be canceled in that method, and advice on how to check if the event was a back navigation.

PhoneApplicationPage complements these events to complete the navigation lifecycle with the more useful OnNavigatedTo and OnNavigatedFrom method callbacks for the page. To better understand and easily remember when these callbacks are called, it's best to complete their method names with a "this page." One method is called when the user has "navigated to this page," and later the other method gets called when the user is "navigated from this page" onto another page.

Figure 3 shows the sequence of events as you navigate across pages. The symmetry between NavigatedTo/NavigatedFrom makes these two methods ideal for starting and stopping work that's required when the page is visible, but not required when the page is in the back stack. Also notice that NavigatedTo always fires before a page is loaded, so don't assume the contents of the page are loaded at this time.

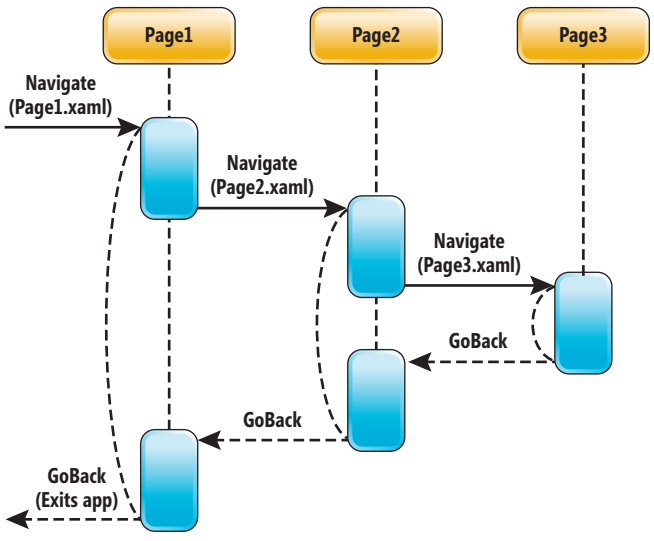


Figure 5 **The PhoneApplicationPage Lifecycle**

The reason OnNavigatedTo and OnNavigatedFrom are critical to Windows Phone is because of the back stack. The OS maintains the back stack for pages you can go back to, so pages aren't immediately unloaded, destroyed or garbage collected when a navigation happens from one page to another. Instead, the pages are moved to the back stack and kept alive (in memory), and when the user clicks back to get to that page, the page is simply added back into the visual tree. The page isn't recreated (unless the application has been deactivated and tombstoned between when the user left the page and clicked back). Because forward journaling isn't supported, pages are eligible for garbage collection when you navigate from a page back to the previous page—assuming there are no other references to this page.

Figure 5 shows a diagram that illustrates a PhoneApplicationPage lifecycle.

As you navigate from Page1 to Page2 and then Page3, pages aren't garbage collected until you call the GoBack method from the page. Inactive pages are in the back stack, but still in memory. If these pages are listening to global events, the event listeners are still active.

Despite a page not getting garbage collected when you navigate from it, the page is no longer visible or active until you navigate back to it, so you should make sure you do any cleanup and release any expensive resources when the user has navigated away from a page. For example, if you're listening to location changes using GeoCoordinateWatcher, you should stop the listener on the OnNavigatedFrom and restart it when the user navigates back to your page–and your page OnNavigatedTo is called.

Code Walk-Through: To see how pages are retained in memory while they're in the back stack, explore the GarbageCollectedSample page included in the accompanying code download. It keeps a running tally of pages in memory, and you can see it increase as you navigate forward and decrease as you navigate back from a page.

That wraps up the first part of our series. Next month, we'll focus on advanced navigation topics. ■

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. You can reach him on Twitter: @jaimerodriguez or on blogs.msdn.com/jaimer.*

THANKS to the following technical expert for reviewing this article: *Peter Torr*

JAMES MCCAFFREY

Diffusion Testing

In this month's column, I introduce you to a software-testing technique I call diffusion testing. The key idea of diffusion testing is that it's sometimes possible to automatically generate new test case data from existing test cases that yield a pass result.

Although diffusion testing is a technique that isn't applicable in most software-testing scenarios, when diffusion testing *is* applicable, it can greatly improve the efficiency of your test effort. Perhaps in part because it's a niche technique, diffusion testing is one of the least known of all major testing techniques, based on my experience.

Before I present examples of diffusion testing, let me explain the motivation behind the method. A test case typically consists of a test case ID, a set of one or more inputs and an expected result. For example, the vector {001, 2, 3, 5} could represent a test case for a Sum function, with ID = 001, inputs = 2 and 3 and an expected result = 5. The test case inputs are sent to the system under test, an actual result is produced and the actual result is compared to the expected result to determine a test case pass/fail result.

In many software-testing situations, it's difficult and time-consuming to determine the expected result part of a test case. For example, suppose you're testing a basic math function that computes the harmonic mean of two inputs that are rates. The average of 30.0 kilometers per hour (kph) and 60.0 kph is *not* (30.0 + 60.0) / 2 = 45.0 kph, but rather the harmonic mean of 30.0 and 60.0, which is 1 / ((1/30.0 + 1/60.0) / 2) = 40.0 kph. Generating hundreds of expected results for this function would be tedious, take a lot of time and be prone to error.

> When diffusion testing *is* applicable, it can greatly improve the efficiency of your test effort.

The difficulty of determining test case expected results is a fundamental concept in software testing and is sometimes referred to as the test oracle problem. In fact, one of the holy grails of software testing is the search for techniques that can automatically determine test cases. So the motivation behind diffusion testing is that, if you can somehow automatically generate new test case data, you'll sidestep a time-consuming part of the testing process and be able to test your system more thoroughly.

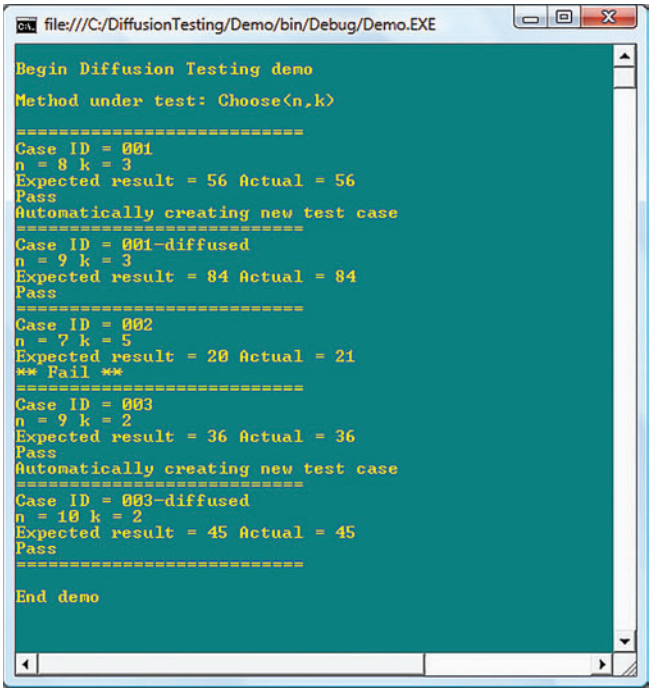


Figure 1 **Diffusion Testing Demo**

Automatically generating test case data with diffusion testing is great in principle, but how does it work? The best way to explain diffusion testing is by way of an example. Take a look at **Figure 1**.

Here, I'm testing a function, Choose(n,k), which returns the number of ways to select k items from n items where order doesn't matter. In my simplified example, I have three existing test cases. The first test case has inputs n = 8 and k = 3 and an expected result of 56. After my test harness executed the first test case, which yielded a pass result, I used diffusion testing to automatically generate a new test case with inputs n = 9, k = 3 and an expected result of 84. Neat! Notice that because test case 002 yielded a fail result, I didn't generate a new diffused test case.

But how are new test cases generated from an existing test case? For the Choose(n,k) function, it turns out that, mathematically, Choose(n+1,k) = Choose(n,k) * (n+1) / (n-k+1). In other words, there's a known relationship between new inputs and old return values. The function I used to generate a diffused test case from an existing test case

Code download available at code.msdn.microsoft.com/mag201103TestRun.

Figure 2 **Generating a New Test Case**

```
static string CreateDiffusedTestCase(string existingTestCase)
{
  // Assumes input format is CaseID:N:K:Expected
  string[] tokens = existingTestCase.Split(':');

  string oldTestCase = tokens[0];
  int oldN = int.Parse(tokens[1]);
  int oldK = int.Parse(tokens[2]);
  long oldExpected = long.Parse(tokens[3]);

  string newTestCase = oldTestCase + "-diffused";
  int newN = oldN + 1;
  int newK = oldK;
  long newExpected = (oldExpected * (oldN + 1)) / (oldN - oldK + 1);

  return newTestCase + ":" + newN + ":" + newK + ":" + newExpected;
}
```

is shown in **Figure 2**. The entire program that generated the output shown in **Figure 1** is available at code.msdn.microsoft.com/mag201103TestRun.

A couple of additional examples may help to make this idea clearer. Suppose you're testing functions that compute the trigonometric sine and cosine. You may recall that sin 2t = 2 * sin t * cos t. So if you have test cases that yield pass results for the sine and cosine of some input, you could use diffusion testing to derive a new test case for the sine function.

Diffusion testing isn't magic. Suppose you're testing a function that accepts a product ID of some sort, searches a SQL database and returns true if the product is in stock and false if the product isn't in stock. Because there's no relationship between different inputs and results, you couldn't use diffusion testing in this scenario. In this respect, diffusion testing is similar to other forms of testing such as boundary condition testing and pairwise testing: It's a technique that's applicable only in certain situations.

Let me present another example of diffusion testing. Suppose you've written a function, Gauss(z), which accepts a standard normal z value and which returns the area under the standard normal (bell-shaped curve) distribution from negative infinity to z. For example, Gauss(-1.645) = 0.0500, Gauss(1.645) = 0.9500 and Gauss(0) = 0.5000. One way to use diffusion testing is to note the monotonic property of Gauss and that for any z value in the range negative infinity to 2.5, the result of Gauss(z + 0.1) must be greater than Gauss(z). Another way to use diffusion testing is to note the symmetric property of Gauss and that for any z value that's less than 0.0, Gauss(-z) = 1.0 - Gauss(z).

> Diffusion testing is one of the least known of all major testing techniques.

The examples I've presented illustrate the three most common—but by no means the only—scenarios where diffusion testing is applicable. The first scenario is where you're testing a mathematical function that can be defined as a recurrence relationship. The second scenario is where you're testing a function that has some monotonic relationship. And the third scenario is where you're testing a function that has some symmetric relationship. A related form of testing, but one that isn't diffusion testing, is when you're testing a function where switching the order of input values doesn't change the return value, such as with Sum(x,y).

Mathematical functions are the most common type of component under test that can benefit from diffusion testing, because such functions most often are recurrent, monotonic or symmetric—but you should be alert to other situations, too. Mathematical functions that involve recurrence relations are especially well-suited for diffusion testing because you can often generate multiple new test cases from an existing test case. In the demo in **Figure 1**, test case 001 with n = 8, k = 3 and expected = 56 generated a new diffused test case with n = 9, k = 3 and expected = 84. This new test case could be used to generate another test case with n = 10, k = 3 and expected = 120, and if that test case passed, it could be used to generate yet another new test case, and so on.

> If you can somehow automatically generate new test case data, you'll sidestep a time-consuming part of the testing process and be able to test your system more thoroughly.

Before I wrap up, let me jump on my soapbox and address a pet peeve of mine related to naming different software-testing techniques and principles. I've labeled the technique described in this column as diffusion testing because existing test cases diffuse, or scatter, to create new cases. I could just as well have called the technique adaptive testing or auto-generation testing or any of a number of other things. It's not the label that's important, but rather the technique represented by the label that counts.

In many fields of study, including software testing, self-proclaimed experts apply some label to a common-sense technique and implicitly attempt to convince people new to the field that the label itself somehow carries some importance. This is typically motivated by the desire to directly sell training or indirectly sell consulting services by delivering a talk at a conference on the marvelous new label. Notable offenders are the terms "exploratory testing" and "context school of testing," but there are many others. So take the term "diffusion testing" for what it is—simply a label to describe a software-testing technique, but one that can be a useful addition to your technical toolkit. ■

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: *Bj Rollison and Alan Page*

TED NEWARD

Multiparadigmatic .NET, Part 6: Reflective Metaprogramming

The marathon continues. In my previous installment we discussed automatic metaprogramming, and by this point the ideas of commonality and variability should be taking on a familiar feel. I'm now full-on talking about metaprogramming—the idea of programs writing programs.

Last month I examined one of the more familiar approaches to metaprogramming: automatic metaprogramming, more commonly known as code generation. In an automatic metaprogramming scenario, developers write programs that describe the "things" to be generated. Usually this is done with the aid of command-line parameters or other inputs such as relational database schemas, XSD files or even Web Services Description Language (WSDL) documents.

Because code generation is essentially "just as if" the code were written by human hands, variability can come at any point inside the code. Data types, methods, inheritance ... all of it can be varied according to need. The drawback, of course, is twofold. First, too much variability can render the generated code (and, more often than not, the templates by which the code is generated) difficult to understand. Second, the generated artifacts are essentially uneditable unless the code generation is somehow partitioned away through the use of partial classes or code generation is no longer necessary.

Fortunately, C# and Visual Basic offer more metaprogrammatic tactics than just automatic metaprogramming, and have done so since the earliest days of the Microsoft .NET Framework.

Persistent Problems

A recurring problem in the object-oriented environment is that of object-to-relational persistence—also known as object-to-XML conversion or, in the more modern Web 2.0 world, object-to-JSON transformation. Despite developers' best efforts, it seems inevitable that object models need to escape the CLR somehow and either move across the network or move onto disk and back again. And herein lies the problem: the previous modes of design—procedural and object-oriented—don't offer good solutions for this dilemma.

Consider a canonical, simplified representation of a human being modeled in code:

```
class Person {
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public int Age { get; set; }

  public Person(string fn, string ln, int a) {
    FirstName = fn; LastName = ln; Age = a;
  }
}
```

Persisting instances of this object, as it turns out, is not difficult, particularly because the properties (in their simplest form) correspond on a one-to-one basis with the columns of a relational table and the properties are publicly accessible. You could write a procedure to take instances of Person, extract the bits of data, inject those bits into SQL statements, and send the resulting statement to the database:

```
class DB {
  public static bool Insert(Person p) {
    // Obtain connection (not shown)
    // Construct SQL
    string SQL = "INSERT INTO person VALUES (" +
      "'" + p.FirstName + "', " +
      "'" + p.LastName + "', " +
      p.Age + ")";
    // Send resulting SQL to database (not shown)
    // Return success or fail
    return true;
  }
}
```

The drawback to a procedural approach rears its ugly head fairly quickly: new types (Pet, Instructor, Student and so on) that also want to be inserted will require new methods of mostly similar code. Worse, if properties exposed to the public API don't correspond one-to-one with columns or internal fields, things can get complicated quickly—developers writing the SQL routines will need to know which fields need persisting and which don't, a pretty clear violation of encapsulation.

From a design perspective, the object-relational problem wants to capture the SQL-esque parts of persisting the data into commonality, so that managing database connections and transactions is handled in one place, but still allows for variability in the actual structure of the things being persisted (or retrieved).

Recall, from our earlier investigations, that procedural approaches capture algorithmic commonality, and that inheritance captures structural commonality while allowing for (positive) variability—but neither of these exactly do what's needed. The inheritance approach—putting the commonality into a base class—will require that developers working in derived classes specify the SQL string and handle much of the in/out bookkeeping (essentially pushing commonality back down into derived classes). The procedural approach will need some kind of variability inside the procedure (extracting and building the SQL to execute) specified from outside the procedure, which turns out to be relatively difficult to achieve.

Enter Metaprogramming

One solution to the object-relational persistence problem frequently cited is that of automatic metaprogramming: using the database schema, create classes that know how to persist themselves to and from the database.

Unfortunately, this has all the traditional problems of code generation, particularly when classes want to change the object representation to be something easier to work with than what the physical database schema implies. For example, a VARCHAR(2000) column would be much easier to work with if it were a .NET Framework System.String and not a char[2000].

Other code-generation techniques started from the class definitions and created a database schema alongside the persistent class definitions … but that meant that somehow now the object hierarchy was duplicated into two different models, one solely for persistence and one for everything else. (Note that as soon as it becomes necessary to transform the object into XML, another hierarchy springs into being that you have to handle, and yet another one to handle JSON. Quickly this approach grows intractable.)

Fortunately, *reflective metaprogramming* offers potential relief. A part of the .NET Framework since 1.0, System.Reflection allows developers to examine the structure of objects at run time, which in this case permits the persistence-minded infrastructure the opportunity to examine the structure of the object being persisted and generate the SQL required from there. A basic introduction to System.Reflection is well-documented both within the MSDN documentation at msdn.microsoft.com/library/f7ykdhsy(v=VS.400) and in the *MSDN Magazine* articles "Use Reflection to Discover and Assess the Most Common Types in the .NET Framework" (msdn.microsoft.com/magazine/cc188926) and "CLR Inside Out: Reflections on Reflection" (msdn.microsoft.com/magazine/cc163408). I won't discuss it any further here.

Reflection permits commonality of algorithm, while allowing for variability of structure manipulated by that algorithm, all while continuing to preserve the appearance of encapsulation. Because reflection (in an appropriately configured security context) has access to private members of objects, internal data can still be manipulated without forcing those data members to be made public.

Positive variability—the ability to vary by adding things—is, as always, easy to work with, as the number of fields is largely irrelevant to most reflection-based code. Negative variability—the ability to vary by removing things—doesn't seem to fit at all, however. After all, a class without fields doesn't really need to be persisted, does it? And a reflection-based infrastructure looping through private fields won't have much of a problem not looping at all, as nonsensical as that may seem.

However, negative variability here is slightly different than just not having fields. In certain scenarios, the Person class will have internal fields that don't want to be persisted at all. Or, more strikingly, the Person class will have fields it wants persisted in a different data format than its CLR-hosted representation. Person.Birthdate wants to be stored as a String, perhaps, or even across three columns (day, month, year) rather than in a single column. In other words, negative variability in a reflective metaprogrammatic sense is not about the lack of fields, but about doing something different to certain instances of types that would otherwise be handled in a standard way (persisting a string to a VARCHAR column being the standard, for example, but for one or more particular fields, persisting a string to a BLOB column).

The .NET Framework makes use of custom attributes to convey this negative variability. Developers use attributes to tag elements within the class to convey the desire for that custom handling, such as @NotSerialized in the case of object serialization. It's important to note, however, that the attribute itself does nothing—it's merely a flag to the code looking for that attribute. Of itself, then, the attribute provides no negative variability, but merely makes it easier to denote when that negative variability should kick in.

Attributes can also be used to convey positive variability. One example is how the .NET Framework uses attributes to convey transactional handling, assuming that the lack of an attribute on a method indicates no transactional affinity whatsoever.

Now *names* can be used to refer to elements within the program.

Mirror, Mirror, on the Wall

Without attributes, reflective metaprogramming establishes an entirely new kind of variability. Now *names* can be used to refer to elements within the program (rather than through compiler symbols)—and at a much later time (runtime) than the compiler traditionally permits. For example, early drops of the NUnit unit-testing framework, like its cousin JUnit in the Java space, used reflection to discover methods that began with "test" as part of the name, and assumed that they were test methods to execute as part of a test suite.

The name-based approach requires developers to take elements traditionally reserved for human eyes—the names of things—and require them to follow strict conventions, such as the "test" prefix for NUnit methods. The use of custom attributes relaxes that naming-based convention (at the expense of now requiring additional code constructs in the classes in question), essentially creating an opt-in mechanism that developers must accept in order to receive the benefits of the metaprogram.

Attributes also provide the ability to tag arbitrary data along with the attribute, providing a much more fine-grained parameterization to the metaprogrammatic behavior. This is something not typically possible with automatic metaprogramming, particularly not when the client wants different behavior for structurally similar constructs (such as the strings-to-BLOBs-instead-of-VARCHAR-columns example from earlier).

Owing to its runtime-bound nature, however, reflection frequently enforces a performance hit on code that uses it extensively. In addition, reflection doesn't offer solutions to the problems cited in the automatic metaprogramming scenario from last month—the proliferation of classes, for example. Another metaprogrammatic solution is available, but that will have to wait for next month.

Happy coding! ■

TED NEWARD *is a Principal with Neward & Associates, an independent firm specializing in enterprise Microsoft .NET Framework and Java platform systems. He's written more than 100 articles, is a C# MVP and INETA speaker and has authored and coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He also consults and mentors regularly. Reach him at ted@tedneward.com with questions or consulting requests, and read his blog at blogs.tedneward.com.*

THANKS to the following technical expert for reviewing this article: *Anthony Green*

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!

Touch Gestures on Windows Phone 7

As someone who spends much of his professional life observing the evolution of APIs, I've been quite entertained by that little corner of the API universe occupied by multi-touch. I'm not sure I'd even want to count the number of different multi-touch APIs spread out over Windows Presentation Foundation (WPF), Microsoft Surface, Silverlight, XNA and Windows Phone, but what's most evident is that a "unified theory" of multi-touch is still elusive.

Of course, this plethora of touch APIs shouldn't be surprising for a technology that's still comparatively young. Moreover, multi-touch is more complex than the mouse. That's partially due to the potential interaction of multiple fingers, but it also reflects the difference between a purely artificial device such as the mouse and all-natural fingers. We humans have a lifetime of experience using our fingers, and we expect them to interact with the world in well-known ways, even if we're touching the glossy surface of a video display.

For the application programmer, Windows Phone 7 defines four—yes, four—different touch interfaces.

Silverlight applications written for Windows Phone 7 have the option of obtaining low-level touch input through the static Touch.FrameReported event, or higher-level input through the various Manipulation routed events. These Manipulation events are mostly a subset of similar events in WPF, but they're different enough to cause major headaches.

XNA applications for Windows Phone 7 use the static TouchPanel class to obtain touch input, but that single class actually incorporates two touch interfaces: The GetState method obtains low-level finger activity, and the ReadGesture method obtains higher-level gestures. The gestures supported by the ReadGesture method are not stylus-like gestures such as checkmarks and circles. They're much simpler gestures described by names such as Tap, Drag and Pinch. In keeping with XNA architecture, touch input is polled by the application rather than being delivered through events.

Gestures Come to Silverlight

I naturally assumed that Silverlight for Windows Phone 7 already had a sufficient number of multi-touch APIs, so I was quite surprised to see a third one added to the mix—albeit in a toolkit that came out a little too late for me to describe in my book, "Programming Windows Phone 7" (Microsoft Press, 2010).

As you probably know, various releases of WPF and Silverlight over the past several years have been supplemented by toolkits released through CodePlex. These toolkits allow Microsoft to get new classes to developers outside of the usual ship cycle and often give us a "sneak peek" at enhancements to the frameworks that might be incorporated in future releases. Full source code is an extra bonus.

Windows Phone 7 now also benefits from this custom. The Silverlight for Windows Phone Toolkit (available at silverlight.codeplex.com) contains DatePicker, TimePicker and ToggleSwitch controls already familiar to users of Windows Phone 7; a WrapPanel (handy for dealing with phone orientation changes); and multi-touch gesture support.

> I've been quite entertained by that little corner of the API universe occupied by multi-touch.

This new Silverlight gesture support in the toolkit is intended to be similar to the XNA TouchPanel.ReadGesture method, except it's delivered through routed events rather than polling.

How similar is it? Much more so than I expected! Looking at the source code, I was quite surprised to discover that these new Silverlight gesture events were entirely derived from a call to the XNA TouchPanel.ReadGesture method. I wouldn't have thought that a Silverlight application on Windows Phone was allowed to call this XNA method, but there it is.

Although the Silverlight and XNA gestures are fairly similar, the properties associated with the gestures are not. The XNA properties use vectors, for example, and because Silverlight doesn't include a Vector structure (an omission I feel is ridiculous), the properties had to be redefined for Silverlight in certain simple ways.

As I've been working with these gesture events, they've come to be my favorite multi-touch API for Silverlight for Windows Phone. I've found them to be comprehensive for much of what I need to do and also fairly easy to use.

Let me demonstrate by giving these gestures actual work to do.

Gesture Service and Listener

All the source code for this column is in a downloadable Visual Studio solution named GestureDemos that contains three projects. You'll

Code download available at code.msdn.microsoft.com/mag201103UIFrontiers.

need to have the Windows Phone 7 development tools installed, of course, and also the Silverlight for Windows Phone Toolkit.

After installing the toolkit, you can use it in your own Windows Phone projects by adding a reference to the Microsoft.Phone.Controls.Toolkit assembly. In the Add Reference dialog box, it should be listed under the .NET tab.

In a XAML file, you'll then need an XML namespace declaration like this one (but all on one line):

```
xmlns:toolkit=
"clr-namespace:Microsoft.Phone.Controls;
assembly=Microsoft.Phone.Controls.Toolkit"
```

Here are the 12 available gesture events, roughly in the order that I'll discuss them (the events that I've grouped on a single line are related and occur in a sequence):

```
GestureBegin, GestureCompleted
Tap
DoubleTap
Hold
DragStarted, DragDelta, DragCompleted
Flick
PinchStarted, PinchDelta, PinchCompleted
```

Suppose you want to handle Tap and Hold events that occur on a Grid or any child of the Grid. You can specify that in the XAML file like so:

```
<Grid ... >
  <toolkit:GestureService.GestureListener>
    <toolkit:GestureListener
      Tap="OnGestureListenerTap"
      Hold="OnGestureListenerHold" />
  </toolkit:GestureService.GestureListener>
    ...
</Grid>
```

You indicate the events and handlers in a GestureListener tag that's a child of the GestureListener attached property of the GestureService class.

Alternatively in code, you'll need a namespace directive for the Microsoft.Phone.Controls namespace and the following code:

```
GestureListener gestureListener =
  GestureService.GetGestureListener(element);

gestureListener.Tap += OnGestureListenerTap;
gestureListener.Hold += OnGestureListenerHold;
```

In either case, if you're setting this gesture listener on a panel, make sure that the Background property is at least set to Transparent! Events will simply fall through a panel with a default background of null.

Tap and Hold

All gesture events are accompanied by event arguments of type GestureEventArgs or a type that derives from GestureEventArgs. The OriginalSource property indicates the top-most element touched by the first finger that meets the screen; the GetPosition method provides the current coordinates of that finger relative to any element.

The gesture events are routed, which means that they can travel up the visual tree and be handled for any element that has a GestureListener installed. As usual, an event handler can set the Handled property of GestureEventArgs to true to prevent an event from travelling further up the visual tree. However, this only affects other elements using these gesture events. Setting Handled to true does not prevent elements higher in the visual tree from obtaining touch input through other interfaces.

The GestureBegin event indicates that a finger has touched a previously fingerless screen; GestureCompleted signals when all fingers have left the screen. These events may be handy for initialization or cleanup, but you'll generally be more focused on gesture events that occur between these two events.

Figure 1 **The Image Element in ScaleAndRotate**

```
<Image Name="image"
  Source="PetzoldTattoo.jpg"
  Stretch="None"
  HorizontalAlignment="Left"
  VerticalAlignment="Top">
  <Image.RenderTransform>
    <TransformGroup>
      <MatrixTransform x:Name="previousTransform" />

        <TransformGroup x:Name="currentTransform">
          <ScaleTransform x:Name="scaleTransform" />
          <RotateTransform x:Name="rotateTransform" />
          <TranslateTransform x:Name="translateTransform" />
        </TransformGroup>
    </TransformGroup>
  </Image.RenderTransform>
</Image>
```

I'm not going to spend much time on the simpler gestures. A Tap occurs when a finger touches the screen and then lifts up within about 1.1 seconds, without moving too far from the original position. If two taps are close in succession, the second one comes through as a DoubleTap. A Hold occurs when a finger is pressed on the screen and remains in roughly the same spot for about 1.1 seconds. The Hold event is generated at the end of this time without waiting for the finger to lift.

Drag and Flick

A Drag sequence—consisting of a DragStarted event, zero or more DragDelta events and a DragCompleted event—occurs when a finger touches the screen, moves and lifts. Because it isn't known that dragging will occur when a finger first touches the screen, the DragStarted event is delayed until the finger actually starts moving beyond the Tap threshold. The DragStarted event might be preceded by a Hold event if the finger has been on the screen without moving for about a second.

> We humans have a lifetime of experience using our fingers, and we expect them to interact with the world in well-known ways.

Because the finger has already begun moving when the DragStarted event is fired, the DragStartedEventArgs object can include a Direction property of type Orientation (Horizontal or Vertical). The DragDeltaEventArgs object accompanying the DragDelta event includes more information: HorizontalChange and VerticalChange properties that are convenient for adding to the X and Y properties of a TranslateTransform, or the Canvas.Left and Canvas.Top attached properties.

The Flick event occurs when a finger leaves the screen as it's still moving, suggesting that the user wants inertia to occur. The event arguments include an Angle (measured clockwise from the positive X axis) and HorizontalVelocity and VerticalVelocity values, both in pixels per second.

The Flick event can occur in isolation; or it can occur between DragStarted and DragCompleted events without any DragDelta events; or it might follow a series of DragDelta events before DragCompleted. Generally you'll want to handle Drag events and Flick events in conjunction, almost as if the Flick is a continuation of the Drag. However, you'll need to add your own inertia logic.

Although the Silverlight and XNA gestures are fairly similar, the properties associated with the gestures are not.

This is demonstrated in the DragAndFlick project. The display contains an ellipse that the user simply drags around with a finger. If the finger leaves the screen with a flicking motion, then a Flick event occurs and the Flick handler saves some information and installs a handler for the CompositionTarget.Rendering event. This event—which occurs in synchronization with the video display refresh—keeps the ellipse moving while applying a deceleration to the velocity.

Bouncing off the sides is handled a bit unusually: The program maintains a position as if the ellipse simply keeps moving in the same direction until it stops; that position is folded into the area in which it can bounce.

Pinch Me, I Must Be Dreaming

The Pinch sequence occurs when two fingers are touching the screen; it's generally interpreted to expand or contract an on-screen object, possibly rotating it as well.

There's no question that the pinching operation constitutes one of the most treacherous areas of multi-touch processing, and it's not unusual to see higher-level interfaces fail at providing adequate information. Most notoriously, the Windows Phone 7 ManipulationDelta event is particularly tricky to use.

When handling gestures, Drag sequences and Pinch sequences are mutually exclusive. They don't overlap but they can occur back to back. For example, press a finger to the screen and drag it. That generates a DragStarted and multiple DragDelta events. Now press a second finger to the screen. You'll get a DragCompleted to complete the Drag sequence followed by a PinchStarted and multiple PinchDelta events. Now lift the second finger while the first finger keeps moving. That's a PinchCompleted to complete the Pinch sequence, followed by DragStarted and DragDelta. Depending on the number of fingers touching the screen, you're basically alternating between Drag sequences and Pinch sequences.

One helpful characteristic of this Pinch gesture is that it doesn't discard information. You can use properties of the event arguments to entirely reconstruct the positions of the two fingers, so you can always go back to first principles if you need to.

During a Pinch sequence, the current location of one finger—let's call it the primary finger—is always available with the GetPosition method. For this discussion, call that return value pt1. For the PinchStarted event, the PinchStartedGestureEventArgs class has two additional properties named Distance and Angle indicating the location of the second finger relative to the first. You can easily calculate that actual location using the following statement:

```
Point pt2 = new Point(pt1.X + args.Distance * Cos(args.Angle),
                      pt1.Y + args.Distance * Sin(args.Angle));
```

The Angle property is in degrees, so you'll need Cos and Sin methods to convert to radians before calling Math.Cos and Math.Sin. Before the PinchStarted handler has completed, you'll also want to save the Distance and Angle properties in fields, perhaps named pinchStartDistance and pinchStartAngle.

The PinchDelta event is accompanied by a PinchGestureEventArgs object. Once again, the GetPosition method gives you the location of the primary finger, which has perhaps moved from its original location. For the second finger, the event arguments provide DistanceRatio and TotalAngleDelta properties.

The DistanceRatio is the ratio of the current distance between the fingers to the original distance, which means you can calculate the current distance like so:

```
double distance = args.DistanceRatio * pinchStartDistance;
```

The TotalAngleDelta is a difference between the current angle between the fingers and the original angle. You can calculate the current angle like this:

```
double angle = args.TotalAngleDelta + pinchStartAngle;
```

Now you can calculate the location of the second finger as before:

```
Point pt2 = new Point(pt1.X + distance * Cos(angle),
                      pt1.Y + distance * Sin(angle));
```

You don't need to save any additional information to fields during PinchDelta handling to process further PinchDelta events.

For the application programmer, Windows Phone 7 defines four—yes, four—different touch interfaces.

The TwoFingerTracking project demonstrates this logic by displaying blue and green ellipses that track one or two fingers around the screen.

Scale and Rotate

The PinchDelta event also provides sufficient information to perform scaling and rotation on objects. I had to supply my own matrix multiplication method, but that was about the extent of the hassles.

To demonstrate, the ScaleAndRotate project implements what is now a "traditional" type of demonstration that lets you drag, scale and optionally rotate a photograph. To perform these transforms, I defined the Image element with a double-barreled RenderTransform as shown in **Figure 1**.

Figure 2 **The ScaleAndRotate Code**

```
public partial class MainPage : PhoneApplicationPage
{
    bool isDragging;
    bool isPinching;
    Point ptPinchPositionStart;

    public MainPage()
    {
        InitializeComponent();
    }

    void OnGestureListenerDragStarted(object sender, DragStartedGestureEventArgs args)
    {
        isDragging = args.OriginalSource == image;
    }

    void OnGestureListenerDragDelta(object sender, DragDeltaGestureEventArgs args)
    {
        if (isDragging)
        {
            translateTransform.X += args.HorizontalChange;
            translateTransform.Y += args.VerticalChange;
        }
    }

    void OnGestureListenerDragCompleted(object sender,
      DragCompletedGestureEventArgs args)
    {
        if (isDragging)
        {
            TransferTransforms();
            isDragging = false;
        }
    }

    void OnGestureListenerPinchStarted(object sender,
      PinchStartedGestureEventArgs args)
    {
        isPinching = args.OriginalSource == image;

        if (isPinching)
        {
            // Set transform centers
            Point ptPinchCenter = args.GetPosition(image);
            ptPinchCenter = previousTransform.Transform(ptPinchCenter);

            scaleTransform.CenterX = ptPinchCenter.X;
            scaleTransform.CenterY = ptPinchCenter.Y;

            rotateTransform.CenterX = ptPinchCenter.X;
            rotateTransform.CenterY = ptPinchCenter.Y;

            ptPinchPositionStart = args.GetPosition(this);
        }
    }

    void OnGestureListenerPinchDelta(object sender, PinchGestureEventArgs args)
    {
        if (isPinching)
        {
            // Set scaling
            scaleTransform.ScaleX = args.DistanceRatio;
            scaleTransform.ScaleY = args.DistanceRatio;

            // Optionally set rotation
            if (allowRotateCheckBox.IsChecked.Value)
                rotateTransform.Angle = args.TotalAngleDelta;

            // Set translation
            Point ptPinchPosition = args.GetPosition(this);
            translateTransform.X = ptPinchPosition.X - ptPinchPositionStart.X;
            translateTransform.Y = ptPinchPosition.Y - ptPinchPositionStart.Y;
        }
    }

    void OnGestureListenerPinchCompleted(object sender, PinchGestureEventArgs args)
    {
        if (isPinching)
        {
            TransferTransforms();
            isPinching = false;
        }
    }

    void TransferTransforms()
    {
        previousTransform.Matrix = Multiply(previousTransform.Matrix,
          currentTransform.Value);

        // Set current transforms to default values
        scaleTransform.ScaleX = scaleTransform.ScaleY = 1;
        scaleTransform.CenterX = scaleTransform.CenterY = 0;

        rotateTransform.Angle = 0;
        rotateTransform.CenterX = rotateTransform.CenterY = 0;

        translateTransform.X = translateTransform.Y = 0;
    }

    Matrix Multiply(Matrix A, Matrix B)
    {
        return new Matrix(A.M11 * B.M11 + A.M12 * B.M21,
                          A.M11 * B.M12 + A.M12 * B.M22,
                          A.M21 * B.M11 + A.M22 * B.M21,
                          A.M21 * B.M12 + A.M22 * B.M22,
                          A.OffsetX * B.M11 + A.OffsetY * B.M21 + B.OffsetX,
                          A.OffsetX * B.M12 + A.OffsetY * B.M22 + B.OffsetY);
    }
}
```

When a Drag or Pinch operation is in progress, the three transforms in the nested TransformGroup are manipulated to move the picture around the screen, scale it and rotate it. When a DragCompleted or PinchCompleted event occurs, the Matrix in the MatrixTransform named previousTransform is multiplied by the composite transform available as the Value property of the TransformGroup. The three transforms in this TransformGroup are then set back to their default values.

Scaling and rotation are always relative to a center point, which is the point that remains in the same location when the transform occurs. A photograph scaled or rotated relative to its upper-left corner ends up in a different location than a photograph scaled or rotated relative to its lower-right corner.

The ScaleAndRotate code is shown in **Figure 2**. I use the primary finger as the scaling and rotation center; these center points are set on the transforms during PinchStarted handling and they don't change for the duration of the Pinch sequence. During PinchDelta events, the DistanceRatio and TotalAngleDelta properties provide scaling and rotation information relative to that center. Any change in movement of the primary finger (which must be detected with a saved field) then becomes an overall translation factor.

That's certainly the simplest pinch code I've ever written, and that fact is perhaps the best endorsement I can provide for this new gesture interface.

Perhaps a unified theory of multi-touch isn't far off after all. ■

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: *Richard Bailey*

DAVID PLATT

Missing the (Power) Point

I just sat through the gazzilionth bad PowerPoint presentation of my working life. Has anyone ever seen a good PPT presentation? I can't remember one. They're rarer than Siberian tigers.

This speaker had apparently just discovered slide transition effects, and used a different one for every slide. Bounces. Dissolves. Wipe downs. A clockwise wheel with eight spokes. It reminded of the late '80s when we first got laser printers. We switched fonts every few words just *because* **we** could, so our documents resembled kidnap ransom notes.

Worse, this presenter did nothing but read out his PPT bullets one by one. I pay $2,000 to get groped by the TSA, crammed into a metal tube with crying infants and flung across many time zones, eat bad hotel food and sleep on lumpy mattresses to hear some art major read a list of bullet points? I could've stayed home, slept in my own Tempur-Pedic bed, played with my kids, and read the darn bullet points myself.

A speaker should have to earn a license before presenting a PPT talk. Until you enter the authorization key from a certified instructor, PPT will refuse to run on any computer connected to a projector. You'd learn to stroll through the audience as they gather for your talk, greeting as many as you could—addressing them by the names on their badges, shaking hands if you could. You'd ask where they're from, what they're doing in their jobs, what they came here to learn from you. You'd start getting onto their wavelength, and getting them onto yours.

And during the talk, you'd get out from behind the podium and use a remote clicker to switch slides. You'd make eye contact with individuals, placing a monitor facing you at the front of the audience so you wouldn't have to turn your head away from them to see the screen that's being projected. If the venue wouldn't give you one, you could have a colleague hold up a laptop with its screen facing you. If a colleague wouldn't do it, you could bribe an attendee. If none of those work, then you'd be alone in the world, my friend, and have problems way bigger than PPT. But you shouldn't inflict your problems on your audience.

Likewise, anyone who composed a PPT deck would have to earn a new, different certification before PPT would project it. The current PPT certification teaches you how to do things, but it doesn't teach you what to do and what not to, or where or when or why. It teaches you how to start the chain saw, but not which end to hold.

For example, it teaches you how to do color gradients, but doesn't tell you when they improve the audience's experience and when they degrade it. The slides in the Microsoft November 2010 Windows Azure training kit all contain a horizontal color gradient, dark blue on the left fading to light blue on the right (see **Figure 1**). "That's our branding," said one Microsoft employee. "It looks nice. What's wrong with it?"

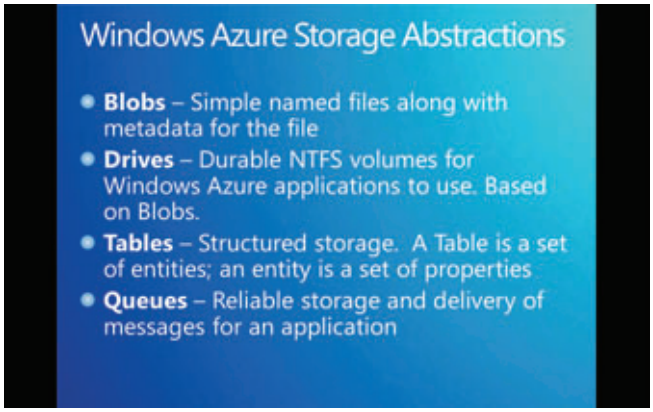


Figure 1 **The Background Isn't Easy on the Eyes**

Here's what's wrong: The constantly changing contrast between the white text and the background color requires constant fine muscle adjustment as your eye scans the line. That quickly causes pain in your eye muscles, your body saying, "Hey, knock it off." Try it now: Concentrate on the text in the figure. You'll feel the first twinges of genuine pain in under a minute. And class attendees look at these slides for three days. Ouch.

This branding says, "We're Microsoft. We know we make your head hurt, and we do it anyway. Remember us." I doubt that was the intention, but it is the result.

Both the bad presenter and the bad slide builder had been carefully trained in the use of a software package. They had not been trained in communication with human beings, the ultimate goal of their efforts, which the software package was supposed to enhance. We need to start teaching them what they're really doing. ■

DAVID S. PLATT *teaches Programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

Can Your GRID Do This?

SPREAD 5 ... DOES IT ALL!

Start building smarter applications
GCPowerTools.com

WE ARE
SPREADSHEETS

ActiveReports **Spread** **DataDynamicsReports** **ActiveAnalysis**

GrapeCity PowerTools
Report Analyze Excel

Dundas Dashboard

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™

Dundas Data Visualization

www.dundas.com

(416) 467-5100 • (800) 463-1492