THE MICROSOFT JOURNAL FOR DEVELOPERS

JULY 2011 VOL 26 NO 7

msdn magazine

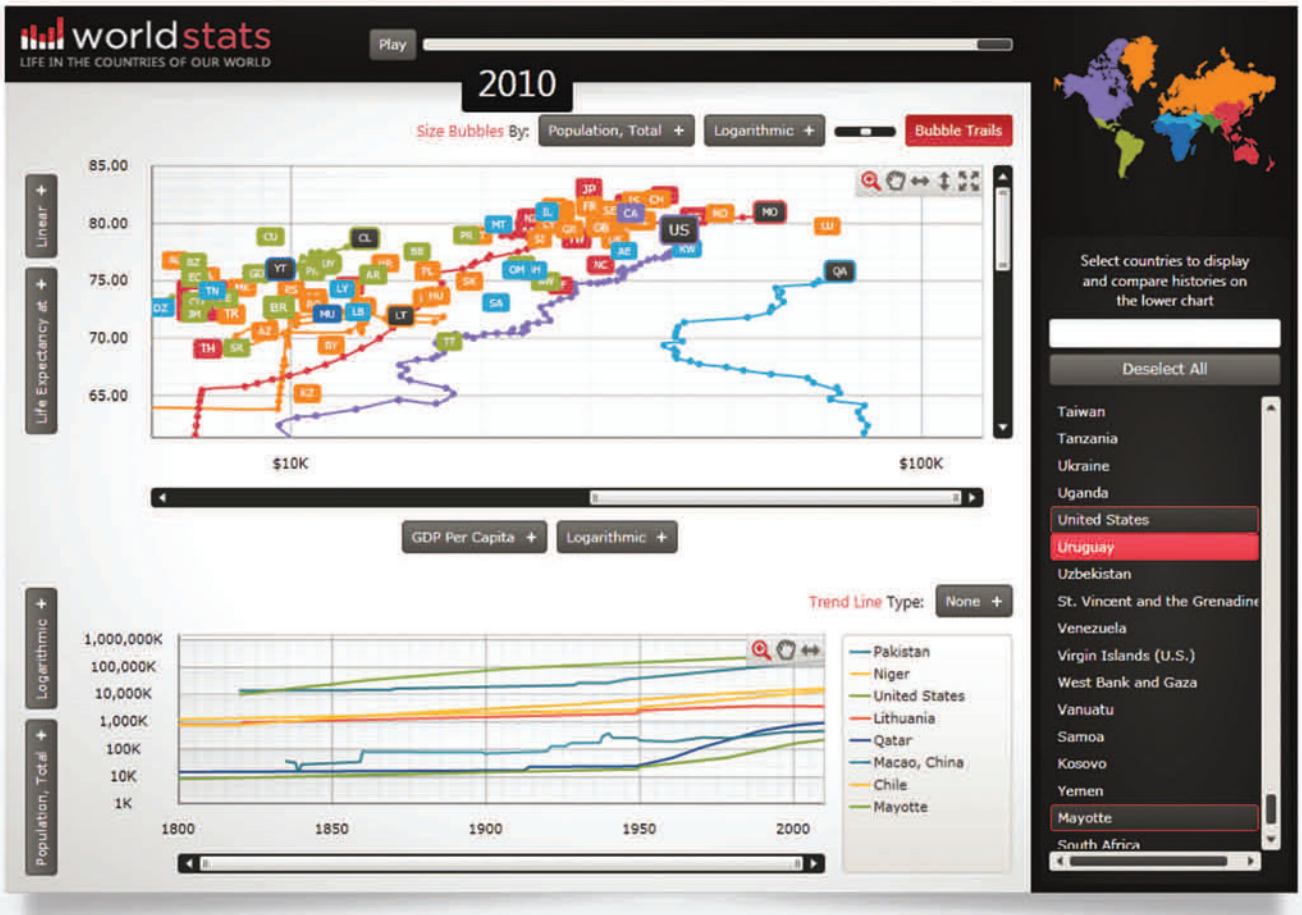
WEB PLATFORM

COLUMNS

Microsoft

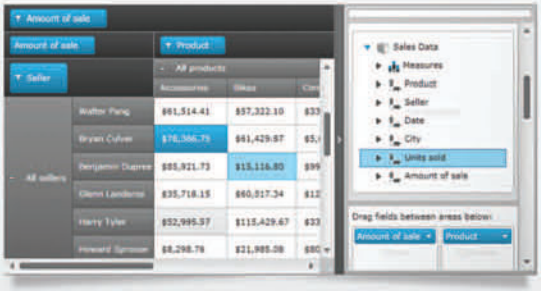
NetAdvantage®
ULTIMATE

REPORTING, DATA VISUALIZATION AND LOB UI CONTROLS FOR ASP.NET, WINDOWS FORMS, JQUERY/HTML5, WPF, SILVERLIGHT AND WINDOWS PHONE 7



INFRAGISTICS MOTION FRAMEWORK™

Delivering a great user experience in Windows Presentation Foundation (WPF) and Microsoft Silverlight business intelligence applications requires more than styling, it requires giving your application's end users greater insight into the story of their data.

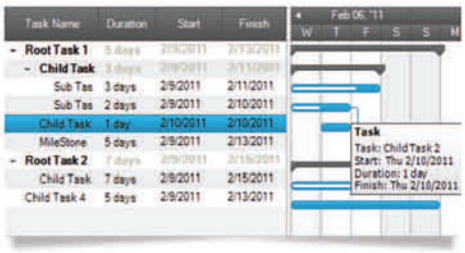


OLAP PIVOT GRID DATA VISUALIZATION

Work with multidimensional data from your OLAP cubes, data warehouses and Microsoft® SQL Server® Analysis Services.

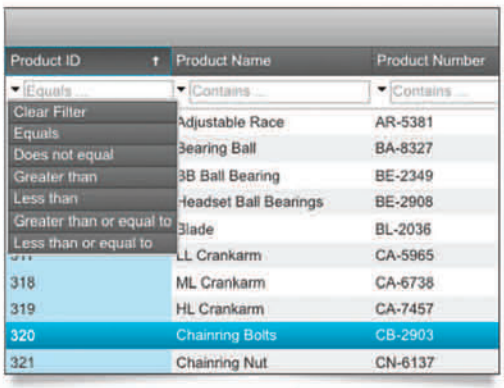
ASP.NET GAUGE

Whether it's for a sidebar gadget or an internal portal such as SharePoint®, gauges play a crucial role on any dashboard.



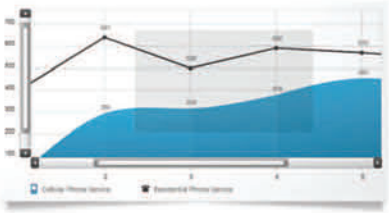
WINDOWS FORMS GANTT CHARTS

Deliver a Microsoft Project-style user experience to your users, with the composite tabular/timeline view of your scheduling data.



jQUERY

The most robust and forward-thinking product we have based on emerging Web technologies including HTML5 and CSS 3.



CHARTING

Make business charting quick and easy with fast, interactive, and vivid visuals across every .NET platform.

EXPORT TO MICROSOFT® WORD

New class library can create Word documents and stream xamGrid™ contents from Silverlight to a Word document.

SILVERLIGHT DATA VISUALIZATION

Use the Silverlight Data Visualization treemap control to communicate differences in data points with different pattern identifications.

SCAN HERE
for an exclusive
look at Ultimate!
www.infragistics.com/ult

TAKE YOUR APPLICATIONS TO THE NEXT LEVEL WITH OUR TOOLS

INFRAGISTICS.COM/ULTIMATE

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

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

msdn magazine

JULY 2011 VOLUME 26 NUMBER 7

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

techxtend.com
866-719-1528

TechXtend | programmer's paradise

LEADTOOLS Document Imaging Suite SDK v17.0

by LEAD Technologies

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

Paradise # L05 03301A02
$4,018.99
programmers.com/LEAD

UltraEdit

The world's #1 text editing solution is also the world's most affordable!

by IDM Computer Solutions

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 Delphi XE

The Fastest Way to Build Native Windows Applications

by Embarcadero

Embarcadero® Delphi® XE is the fastest way to deliver ultra-rich, ultra-fast Windows applications. Dramatically reduce coding time and create applications 5x faster with component-based development and a fully visual two-way RAD IDE. Speed development across multiple Windows and database platforms — for GUI desktop applications, interactive touch-screen, kiosk, and database-driven multi-tier, cloud, and Web applications.

SPECIAL PRICE!

Paradise # CGI 32401A01
$1,977.99
programmers.com/embarcadero

Apple Mac Pro MC560LL/A Workstation

by Apple

The easy-access interior of the Mac Pro feels like the well-organized workstation it is. No rat's nest of components here. You don't need to turn the system on its side or struggle to reach into awkward spaces to make changes. Just remove the side panel for instant access to everything. Slide out the processor tray to add memory. Slide out drive bays to add storage. Slide a simple bar to change up to four expansion cards at once. And with plenty of I/O ports both front and back, you'll have room for all your external devices.

Paradise # ZHI BG5665
$2,498.99
programmers.com/apple

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

Spread 5 for Windows Forms

by GrapeCity PowerTools

- 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

Upgrade
Paradise # F02 01101A01
$936.99
programmers.com/grapecity

HP ProLiant DL380 G7 Servers

by Hewlett Packard

The HP ProLiant DL380 G7 Server continues to deliver on its heritage of engineering excellence with increased flexibility and performance, enterprise-class uptime and HP Insight Control manageability, 2 socket Intel® Xeon® performance, and 2U density for a variety of applications.

Paradise # ZHI GA5596
$4,479.00
programmers.com/hp

Lenovo ThinkPad X220 4290

by Lenovo

Engineered for mobile professionals, the ThinkPad X220 features the quality and performance Lenovo users have come to expect and increased audio, video and communications features that respond to the increased use of laptops as a multimedia and communications tools in business. The ultra-portable ThinkPad X220 comes equipped with a full-powered Intel processor with outstanding graphic performance.

Paradise # ZHI GF0801
$2,169.00
programmers.com/lenovo

StorageCraft ShadowProtect Desktop Edition 4.0

by StorageCraft Technology Corp

ShadowProtect Desktop provides fast and reliable disaster recovery, data protection and system migration to get desktops and laptops online as quickly as possible. ShadowProtect Desktop is an automated backup that works in the background. In the event of a disaster, its flexible recovery options allow you to complete granular recovery of files and folders or full bare metal recovery in a matter of minutes. Hardware Independent Restore (HIR) technology makes it quick and easy to recover to the same system, to dissimilar hardware or to and from virtual environments.

Minimum Quantity: 1
Paradise # SC5 02201E01
$83.99
programmers.com/storagecraft

Programmer's Paradise is getting a new name: TechXtend!

For nearly 30 years, Programmer's Paradise has served the software development and IT communities with the best selection of software, terrific values and a commitment to service excellence.

30 years... much has changed in that time!

However, one thing that won't ever change is that we will still be your "go to" source for software developer tools — AND you'll also be able to depend on TechXtend for all of your other IT needs.

Learn more about our new name: **www.programmers.com/techxtend**

Win an iPad!

NO PURCHASE NECESSARY. Use offer code WEBTRW07 when you place your order online or with your TechXtend/Programmer's Paradise representative and you'll automatically be entered into a drawing to win an iPad Wi-Fi 32GB.

For official rules, or to complete the online entry form: **programmers.com/tradewinds**

Software, Systems, & Solutions – Since 1982

Prices subject to change. Not responsible for typographical errors.

EDITOR'S NOTE

KEITH WARD

Why C++ Still Matters

One of our goals after taking the wheel at *MSDN Magazine* was to bring it closer to readers. And one of the things you've consistently told us is that we need to not treat C++ like the crazy uncle in the attic.

We heard you, and this is the result. This month sees the return of one of our most popular columns, Kenny Kerr's Windows with C++. Kerr wrote this column for years, and we're delighted to welcome back the Yoda of C++ development. To kick it off right, I asked Kerr some questions that put C++ in the context of today's development environment. Here's what he had to say.

What advantages does C++ hold over C#? Kenny Kerr: There are cases where C++ makes more sense. Why did Microsoft develop the native Windows Web Services [WWS] API long after Windows Communication Foundation [WCF] was well established as the premiere Web services stack? Surely, the .NET Framework is ideally suited to crunching XML and handling HTTP requests. Well, it turns out that C++ can still produce dramatically better throughput while using far less memory (bit.ly/czhaLd).

Does this really matter? If you're an investment bank with enough capital to build large computing grids, perhaps not. However, if your customer is developing netbooks and tablets or smartphones, or is concerned about the startup time or battery usage for laptops, or needs to scale their Web server to handle a hundred million video requests without requiring a hundred Web servers, then it begins to matter very much.

How does C++ fit in with the Web and associated technologies? All of the major Web browsers and Web servers are written in C++. The operating systems these applications run on are also written in C++. Whether many developers think about it or not, C++ clearly plays a critical role in making the Web a fast and rich environment for users and developers alike. Of course, when some developers think about programming languages, they're actually thinking about how they can use them to develop applications on top of these Web browsers and servers. That's where domain-specific languages can play a critical role.

JavaScript is the obvious choice for developing applications for the browser. Many domain-specific languages play key roles in building server applications, from SQL for manipulating data, to Razor for producing markup. There is, of course, nothing stopping you from writing Web server applications in C++. Many domain-specific languages require various runtimes that may not be appropriate for some scenarios—shrink-wrapped server applications and embedded devices come to mind. IIS also has a great C++ API for handling performance-critical aspects of a Web application, such as handling I/O or rendering graphical charts.

What are some of the biggest changes in C++ with the latest version? Rather than thinking in terms of changes, you should think in terms of what C++ does well and how those things can be made better. C++ didn't change much, but it has been improved in many ways to make it easier to do the things that C++ has traditionally been good at doing.

For example, C++ has a lot to say about power and performance, but is not always as simple as it could be. C++0x goes a long way toward making it both simpler for the application developer and more expressive for the library developer to build more powerful abstractions that are even more efficient, yet in a simpler and more intuitive way. Move semantics, for example, takes something that was quite difficult to do before and makes it not only efficient, but also simple for both library and application developers. Lambdas are another great example of C++0x making C++ more expressive while reducing the need to create function objects that fragment the structure of an application unnecessarily.

What are your main goals with your new column? I believe that C++ and Windows together make a powerful combination for producing great applications. Many developers have lost sight of this, driven to a large degree by Microsoft's emphasis on the .NET Framework. I hope to show developers that it's feasible—and even enjoyable—to write applications for Windows with C++ using the Windows API. Much like the .NET Framework, the Windows API continues to expand to offer more capabilities. You only have to look at applications like Internet Explorer 9 to realize the potential of C++ on Windows 7. I also think that Windows 8 and beyond is going to usher in even more capabilities specifically for C++ developers.

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

The Project Management Software...

NEW!

...that left its parents on Krypton.

Project Management • Bug Tracking • Agile/Scrum

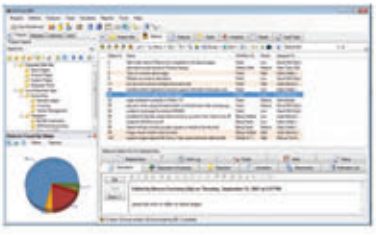
Watch the Video & Sign Up for a Trial at axosoft.com/11

Hosted

OnTime Now!

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

Windows



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

Web

OnTime Web provides you with an anywhere-anytime solution, whether you go with OnTime hosted or installed. This is what a web app is supposed to be - an intuitive UI, blazing fast reaction, and a powerful back end.

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

axosoft.com
800.653.0024

TERRENCE DORSEY

Tools and Techniques for .NET Code Profiling

The pithy epigram "premature optimization is the root of all evil" has been variously attributed over the years to Donald Knuth, William Wulf and C.A.R. Hoare. (It appears that Hoare originated the phrase, but Knuth gets credit for popularizing it. See bit.ly/fW1jWE for a summary of Knuth's thoughts on the subject.)

So what does this mean to you, the intrepid .NET developer? Should you ignore performance and code away to your heart's content? Is it best to follow the whims of IntelliSense and rely on ever-faster hardware to scale your apps?

Well, no.

An important part of testing is making sure your application not only executes without errors, but also executes efficiently and responsively. That's where code-profiling tools and techniques come into play. These let you, as part of the build and testing process, evaluate your code for constructions—and outright errors—that are likely to cause problems. You get an automated heads-up that points you directly at the places in your app that need refactoring.

Profiling Basics

A good place to start learning about .NET profiling is the Code Project article by Paul Glavich, "Profiling the Performance of a .NET Application" (bit.ly/fpua6i). You'll not only get some basics on .NET profiling, but Glavich also introduces you to the ANTS Profiler (which we'll get to a bit later).

Another great starting point is Amirthalingam Prasanna's blog post, "Profiling Your .NET Code" (bit.ly/dDXWsF). He starts out by listing 10 reasons why you should profile your code, then walks you through the process using the CLR Profiler (bit.ly/eSJyWd).

In Brian Long's extensive walkthrough of the Microsoft .NET Framework profiling APIs, ".NET Internals: The Profiling API" (bit.ly/hNEDYP), you'll learn about performance counters, the Performance Monitor and the relevant APIs. You'll also see how to create your own simple profiling tools to illustrate how the APIs work.

The CLR now provides extensive profiling support, and David Broman shows you how to use it in his blog post, "Profilers, in-process side-by-side CLR instances, and a free test harness" (bit.ly/dYeRnQ). Make sure to read through the archives of Broman's site—there's a lot of great information there.

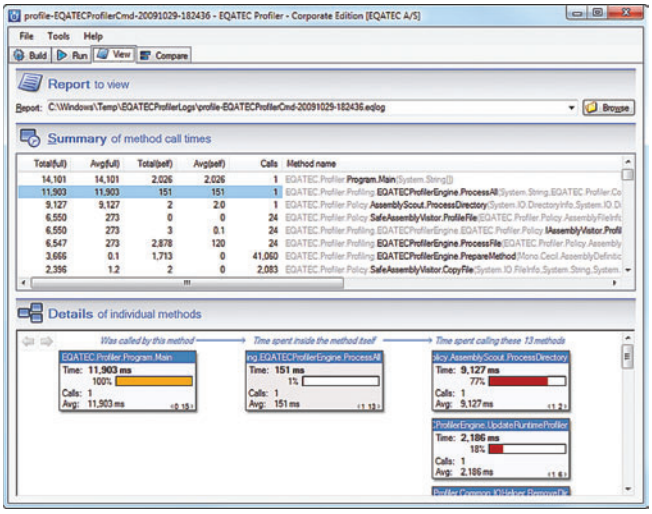
Profiling gets really important when you're doing high-performance, highly scaled programming. If that's up your alley, check out the "SC08: Windows HPC: Multi-Core Parallel Code Profiling in VS2010" webcast on Channel 9 (bit.ly/gyeKPi).

Don't miss the Visual Studio Profiler Team Blog (blogs.msdn.com/b/profiler) for profiling tips, tricks and late-breaking announcements.

Writing More Efficient Code

Of course, one way to avoid the pain of testing, profiling and refactoring your code repeatedly is to write more efficient code in the first place. Here are three articles that discuss best practices for writing apps that will, hopefully, need a few less runs through the profiler to achieve the speed you hoped for:

- "Guide to Improving Code Performance in .NET: Part I" by Satesh Arveti on C# Corner (bit.ly/gylmk9)
- "Writing Efficient C and C Code Optimization" by Koushik Ghosh on Code Project (bit.ly/icnYEi)
- "Writing High Performance .NET Code" by Juan A Rodriguez and Simonijt Dutta from Intel (intel.ly/fvweaP)



EQATEC Profiler

Profiling Tools

This is the Toolbox column, so let's talk tools. Software-profiling utilities perform dynamic analysis of your application code while it's running. Here are a few free and commercial offerings you might want to take for a spin.

AMD **CodeAnalyst Performance Analyzer** (bit.ly/gAqPeu) is a free profiling tool from—you guessed it—Advanced Micro Devices Inc. that lets you profile C/C++, Fortran, Java and .NET code. It's particularly designed for optimizing your code for multithreaded executing on AMD chipsets. CodeAnalyst integrates with Visual Studio 2003 through 2010 and runs on systems from Windows XP through Windows 7 in both x86 and AMD64 architectures.

EQATEC Profiler (bit.ly/h2hDCF) is a code profiler designed specifically for making your .NET apps run faster. It supports the .NET Framework 2.0 and later, Windows Presentation Foundation (WPF), Silverlight and the .NET Compact Framework. It's also the first profiler to offer Windows Phone 7 profiling features both in the emulator and on devices. Free and paid licenses are available. See the site for details.

JetBrains **dotTrace** (jetbrains.com/profiler) is actually a collection of two lightweight .NET profiling apps: dotTrace4 4 Performance and

Blazing-Fast **GRID CONTROLS**

Optimized for .NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/GRIDS**

***or Call Us* (818) 844-3383**

DevExpress™

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

dotTrace 3.5 Memory. dotTrace lets you target apps using the .NET Framework 1.0 through 4, .NET Compact Framework 3.5 and Silverlight 4. You can run tracing, sampling or line-by-line analysis on both local and remote systems. dotTrace integrates with Visual Studio 2005 through 2010. JetBrains currently offers a time-limited free trial and a number of licensing options. See the site for details.

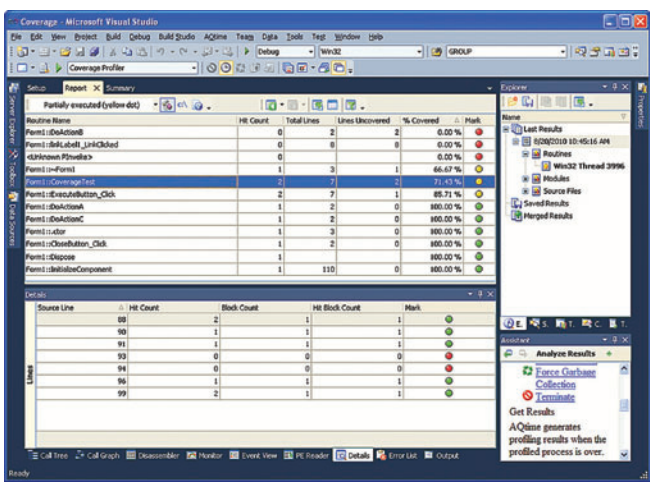
Red Gate **ANTS Performance Profiler** (bit.ly/g1yVEt) and **ANTS Memory Profiler** (bit.ly/h3JzAX) are another set of commercial profiling tools that analyze Windows Forms apps, ASP.NET Web apps, SharePoint and Silverlight 4. With the .NET Framework 4, ANTS Performance Profiler supports CLR profiling so you can attach to running processes. You can also get performance data for SQL queries triggered from your code. Both products support the .NET Framework 1.0 through 4 and Windows XP through Windows 7, and can be run directly from Visual Studio. See the Red Gate site for pricing.

SmartBear **AQtime Pro** (bit.ly/ePmENJ) is a performance profiling and memory and resource debugging program for targeting the .NET Framework, Java, JScript and VBScript code. You can use AQtime Pro to profile both 32- and 64-bit applications, and the profiler can be directly integrated into Visual Studio 2002 through 2010 and Embarcadero RAD Studio development environments. AQtime Pro also provides a programmable debugger using a COM-based architecture so you can create custom profilers. Download a time-limited free trial, or contact Smart Bear for current pricing.

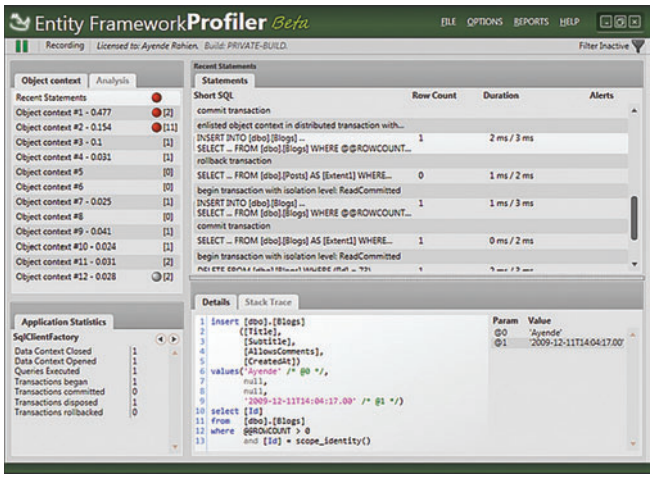
> Premature optimization is the root of all evil.

SlimTune (code.google.com/p/slimtune) is a free, open source profiling and performance-analysis tool for .NET development. Development is currently in beta, and though you can target both x86 and x64 applications, only sample-based profiling is available. Source code is available if you want to hack in it yourself.

Electric Software **GlowCode** (glowcode.com) is another commercial performance and memory profiler targeted at 32- and 64-bit managed, native and mixed code written in C, C++ or any .NET Framework-compliant language. GlowCode integrates directly with Visual Studio



SmartBear AQtime Pro



Entity Framework Profiler

2010 so you can profile your app without leaving the IDE. A time-limited evaluation license is available along with single-user and floating licenses. Check the Web site for details.

Data Access Profiling

The data layer of your application needs love, too. Here are a few tools and resources for profiling data access.

If you're using SQL Server 2008 R2, **SQL Server Profiler** is included as part of the installation; it can analyze queries, Transact-SQL statements and expressions, and so on. Other great resources for getting started with SQL Server Profiler include:

- "Using SQL Server Profiler" in the MSDN Library (msdn.microsoft.com/library/ms187929)
- "Introduction to SQL Server 2008 Profiler" by Pinal Dave on dotnetslackers.com (bit.ly/g5IDA0)
- "Step-By-Step: An introduction to SQL Server Profiler" by Susan Harkins at TechRepublic (tek.io/hTV6Kh)

If you're using SQL Server 2005 or 2008 Express Edition, however, you won't have access to the profiling tools included in the full version of SQL Server. In that case, you'll want to take a look at the free AnjLab open source **SQL Server Express Edition Profiler** (bit.ly/eNg5oi).

Entity Framework Profiler (efprof.com) is designed for analyzing and real-time debugging data access in your code using the Entity Framework. It's a great tool for uncovering what's really happening behind the scenes. Entity Framework Profiler is a commercial product, but you can request a 30-day trial license to give it a whirl on your own projects.

SQL Load Test (sqlloadtest.codeplex.com) is a CodePlex project created by the Visual Studio Team System (VSTS) Ranger team to generate unit tests from SQL Profiler traces and replay the database calls from the trace using Visual Studio Load Test. It's a handy tool for testing data-centric apps that aren't directly load-testable.

Finally, don't forget **LINQPad** (linqpad.net) as a handy tool for evaluating LINQ queries as well as your C# and Visual Basic code. While not really a profiler, it's a handy way to work through iterations of your code outside the context of your project. ■

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

Rock-Solid **REPORTING CONTROLS**
Optimized for.NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today
Visit **DEVEXPRESS.COM/REPORTING**
***or Call Us* (818) 844-3383**

DevExpress™

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

DINO ESPOSITO

Code Contracts: Inheritance and the Liskov Principle

Just like real-life contracts, software contracts bind you to additional constraints and cost you something in terms of time. When a contract is standing, you might want to make sure that you don't break its terms. When it comes to software contracts—including Code Contracts in the Microsoft .NET Framework—nearly every developer will eventually manifest doubts about the computational costs of having contracts around classes. Are contracts appropriate for your software, regardless of the type of build? Or are contracts instead mostly a debugging aid that should be stripped off retail code?

Eiffel, the first language to introduce software contracts, has native language keywords to define preconditions, postconditions and invariants. In Eiffel, therefore, contracts are part of the language. If used in the source code of a class, then contracts become an integral part of the code.

> Nearly every developer will eventually manifest doubts about the computational costs of having contracts around classes.

In .NET, though, contracts are part of the framework and don't belong to supported languages. This means that runtime checking can be enabled or disabled at will. In particular, in .NET you're allowed to decide about contracts on a per-build configuration basis. In Java, things are nearly the same. You use an external framework and either add contract code to the sources to be compiled or ask tools around the framework to modify the bytecode accordingly.

In this article, I'll discuss a few scenarios where Code Contracts prove particularly helpful in driving you toward a higher quality of overall software design.

What Code Contracts Are For

An evergreen best practice of software developers is writing methods that carefully check any input parameters they receive. If the input parameter doesn't match the expectations of the method, then an exception is thrown. This is known as the *if-then-throw* pattern. With precondition contracts, this same code looks nicer and more compact. More interestingly, it also reads better, because a precondition lets you clearly state just what's required instead of testing against what isn't desirable. So, at first sight, software contracts simply look like a nicer-to-write approach to prevent exceptions in class methods. Well, there's a lot more to it than just that.

The simple fact that you think of contracts for each and every method indicates that you're now thinking more about the role of those methods. In the end, design gets terser and terser. And contracts also represent a valuable form of documentation, especially for refactoring purposes.

Code Contracts, however, aren't limited to preconditions, even though preconditions are the easiest part of software contracts to pick up. The combination of preconditions, postconditions and invariants—and the extensive application of them throughout your code—gives you a decisive advantage and really leads to some higher-quality code.

Assertions vs. Code Contracts vs. Tests

Code Contracts aren't entirely like assertions and other debug instruments. While contracts can help you track down bugs, they don't replace either a good debugger or a well-done set of unit tests.

Figure 1 **Inheriting Invariants**

```
public class Rectangle
{
  public virtual Int32 Width { get; set; }
  public virtual Int32 Height { get; set; }

  [ContractInvariantMethod]
  private void ObjectInvariant()
  {
    Contract.Invariant(Width > 0);
    Contract.Invariant(Height > 0);
  }
}
public class Square : Rectangle
{
  public Square()
  {
  }

  public Square(Int32 size)
  {
    Width = size;
    Height = size;
  }

  [ContractInvariantMethod]
  private void ObjectInvariant()
  {
    Contract.Invariant(Width == Height);
  }
  ...
}
```

Powerhouse **ANALYTICS**

Optimized for.NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/ANALYTICS**

***or Call Us* (818) 844-3383**

DevExpress™

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

Like assertions, Code Contracts indicate a condition that must be verified at a certain point during the execution of a program.

An assertion that fails is a symptom that something is wrong somewhere. An assertion, however, can't tell you why it failed and where the problem originated. A Code Contract that fails, on the other hand, tells you a lot more. It shares details about the kind of failure. So, for example, you can learn whether the exception was raised because a given method received an unacceptable value, failed in the calculation of the expected return value or holds an invalid state. Whereas an assertion tells you only about a detected bad symptom, a Code Contract can show invaluable information on how the method should be used. This information may ultimately help you understand what has to be fixed in order to stop violating a given assertion.

> Code Contracts aren't entirely like assertions and other debug instruments.

How do software contracts relate to unit testing? Obviously, one doesn't exclude the other and the two features are sort of orthogonal. A test harness is an external program that works by applying a fixed input to selected classes and methods to see how they behave. Contracts are a way for the classes to yell out when something is wrong. In order to test contracts, however, you must be running the code.

Unit tests are a great tool to catch regression after a deep refactoring process. Contracts are perhaps more informative than tests to document the expected behavior of methods. To get design value out of testing, you must be practicing test-driven development (TDD). Contracts are probably a simpler tool than TDD to document and design methods.

Contracts add extra information to your code and leave it up to you to decide whether this information should make it to the deployed binaries. Unit testing involves an external project that can estimate how the code is doing. Whether you compile contract information or not, having clear contract information in advance helps immensely as a documentation and design aid.

Code Contracts and Input Data

Contracts refer to conditions that always hold true in the normal execution flow of a program. This seems to suggest that the ideal place where you might want to use contracts is in internal libraries that are only subject to input strictly controlled by the developer. Classes that are directly exposed to user input aren't necessarily a good place for contracts. If you set preconditions on unfiltered input data, the contract may fail and throw an exception. But is this really what you want? Most of the time, you want to degrade gracefully or return a polite message to the user. You don't want an exception and you don't want to throw and then trap an exception just to recover gracefully.

In .NET, Code Contracts belong to libraries and may be a good complement to (and, in some cases, a replacement for) data annotations. Data annotations do a great job in relation to the UI because in Silverlight and ASP.NET you have components that understand those annotations and adjust the code or the HTML output. On the domain layer, though, you often need more than just attributes, and Code Contracts are an ideal replacement. I'm not necessarily stating that you can't get the same capabilities with attributes that you can with Code Contracts. I find, however, that in terms of readability and expressivity, the results are better with Code Contracts than attributes. (By the way, this is precisely why the Code Contracts team prefers plain code over attributes.)

Inherited Contracts

Software contracts are inheritable in nearly any platform that supports them, and the .NET Framework is no exception. When you derive a new class from an existing one, the derived class picks up the behavior, context and contracts of the parent. That seems to be the natural course of things. Inheritance of contracts doesn't pose any issue for invariants and postconditions. It's a bit problematic for preconditions, though. Let's tackle invariants and consider the code in **Figure 1**.

The base class Rectangle has two invariants: width and height are greater than zero. The derived class Square adds another invariant condition: width and height must match. Even from a logical perspective, this makes sense. A square is like a rectangle except that it has an additional constraint: width and height must always be the same.

For postconditions, things mostly work in the same manner. A derived class that overrides a method and adds more postconditions just augments the capabilities of the base class and acts like a special case of the parent class that does all the parent does and more.

What about preconditions, then? That's exactly why summing up contracts across a class hierarchy is a delicate operation. Logically speaking, a class method is the same as a math function. Both get some input values and produce some output. In mathematics, the range of values produced by a function is known as the codomain; the domain is the range of possible input values. By adding invariants and postconditions to a derived class method, you just increase the size of the method's codomain. But by adding preconditions, you restrict the method's domain. Is this something you should really be worried about? Read on.

Figure 2 **Illustrating the Liskov Principle**

```
public class Rectangle
{
  public Int32 Width { get; private set; }
  public Int32 Height { get; private set; }

  public virtual void SetSize(Int32 width, Int32 height)
  {
    Width = width;
    Height = height;
  }
}
public class Square : Rectangle
{
  public override void SetSize(Int32 width, Int32 height)
  {
    Contract.Requires<ArgumentException>(width == height);
    base.SetSize(width, width);
  }
}
```

Elegant **CHARTING CONTROLS**

Optimized for .NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/CHARTING**

***or Call Us* (818) 844-3383**

DevExpress™

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

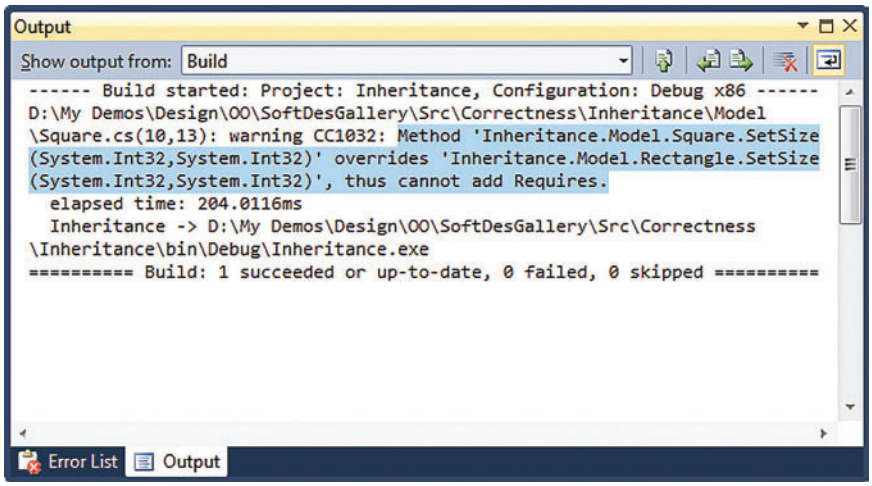


Figure 3 **The Warning You Get When You're Violating the Liskov Principle**

The Liskov Principle

SOLID is a popular acronym that results from the initials of five key principles of software design including Single responsibility, Open/Closed, Interface segregation and Dependency inversion. The L in SOLID stands for the Liskov substitution principle. You can learn a lot more about the Liskov principle at bit.ly/IKXCxF.

In a nutshell, the Liskov principle states that it should always be safe to use a subclass in any place where the parent class is expected. As emphatic as it may sound, this is *not* something that we get out of the box with plain object orientation. No compiler of any object-oriented language can do the magic of ensuring that the principle always holds.

> Contracts are a way for the classes to yell out when something is wrong.

It's a precise developer's responsibility to ensure that it's safe to use any derived class in places where the parent class is expected. Notice I said "safe." Plain object orientation makes it possible to use any derived classes in places where the parent class is expected. "Possible" isn't the same as "safe." To fulfill the Liskov principle, you need to adhere to a simple rule: The domain of a method can't be shrunk in a subclass.

Code Contracts and the Liskov Principle

Aside from the formal and abstract definition, the Liskov principle has a lot to do with software contracts and can be easily rephrased in terms of a specific technology such as .NET Code Contracts. The key point is that a derived class can't just add preconditions. In doing so, it will restrict the range of possible values being accepted for a method, possibly creating runtime failures.

It's important to note that violating the principle doesn't necessarily result in a runtime exception or misbehavior. However, it's a sign that a possible counterexample breaks your code. In other words, effects of violation may ripple across the entire codebase and show nefarious symptoms in apparently unrelated areas. It makes the entire codebase harder to maintain and evolve—a deadly sin these days. Imagine you have the code in **Figure 2**.

The class Square inherits from Rectangle and just adds one precondition. At this point, the following code (which represents a possible counterexample) will fail:

```
private static void Transform(Rectangle rect)
{
  // Height becomes twice the width
  rect.SetSize(rect.Width, 2*rect.Width);
}
```

The method Transform was originally written to deal with instances of the Rectangle class, and it does its job quite well. Suppose that one day you extend the system and start passing instances of Square to the same (untouched) code, as shown here:

```
var square = new Square();
square.SetSize(20, 20);
Transform(square);
```

Depending on the relationship between Square and Rectangle, the Transform method may start failing without an apparent explanation.

Worse yet, you may easily spot how to fix the issue, but because of the hierarchy of classes, it may not be something you want to take lightly. So you end up fixing the bug with a workaround, as shown here:

```
private static void Transform(Rectangle rect)
{
  // Height becomes twice the width
  if (rect is Square)
  {
    // ...
    return;
  }
  rect.SetSize(rect.Width, 2*rect.Width);
}
```

But regardless of your effort, the notorious ball of mud has just started to grow bigger. The nice thing about .NET and the C# compiler is that if you use Code Contracts to express preconditions, you get a warning from the compiler if you're violating the Liskov principle (see **Figure 3**).

The Least-Understood and Least-Applied SOLID Principle

Having taught a .NET design class for a couple of years now, I think I can safely say that of the SOLID principles, the Liskov principle is by far the least understood and applied. Quite often, a weird behavior detected in a software system can be tracked down to a violation of the Liskov principle. Nicely enough, Code Contracts can help significantly in this area, if only you take a careful look at compiler warnings. ■

Dino Esposito *is the author of "Programming Microsoft ASP.NET 4" (Microsoft Press, 2011) and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. You can follow him on Twitter at twitter.com/despos.*

Thanks to the following technical expert for reviewing this article: *Manuel Fahndrich*

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

OCR Module available with royalty-free licensing!

The new OCR Module from Amyuni enables developers to:

- Convert non-searchable PDF files into searchable PDFs
- Create searchable PDF documents out of various image formats such as multi-page TIFF, JPEG or PNG while applying text recognition
- Compress image based PDF documents using high compression JBIG2 or more standard CCITT, JPEG and PNG compression formats

The Amyuni OCR module is based on the Tesseract Library with the Amyuni PDF technology being used to process and create the PDF documents.

Learn more at www.amyuni.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

KENNY KERR

C++ and the Windows API

The Windows API presents a challenge to the C++ developer. The various libraries that make up the API are, for the most part, exposed either as C-style functions and handles or COM-style interfaces. Neither of these is very convenient to work with and requires some level of encapsulation or indirection.

The challenge for the C++ developer is to determine the level of encapsulation. Developers who grew up with libraries like MFC and ATL may be inclined to wrap everything up as classes and member functions, because that's the pattern exhibited by the C++ libraries they've relied on for so long. Other developers may scoff at any sort of encapsulation and just use the raw functions, handles and interfaces directly. Arguably these other developers aren't really C++ developers, but simply C developers with identity issues. I believe there's a more natural middle ground for the contemporary C++ developer.

As I restart my column here at *MSDN Magazine*, I'll show you how you can use C++0x, or C++ 2011 as it will likely be named, along with the Windows API to lift the art of native Windows software development out of the dark ages. For the next few months I'm going to take you through an extended tour of the Windows Thread Pool API. Follow along and you'll discover how to write amazingly scalable applications without the need for fancy new languages and complicated or costly runtimes. All you'll need is the excellent Visual C++ compiler, the Windows API and a desire to master your craft.

> The challenge for the C++ developer is to determine the level of encapsulation.

As with all good projects, some groundwork is needed to get off to a good start. How, then, am I going to "wrap" the Windows API? Rather than bog down every subsequent column with these details, I'm going to spell out my recommended approach in this column and simply build on this going forward. I'll leave the issue of COM-style interfaces for the time being, as that won't be needed for the next few columns.

The Windows API consists of many libraries that expose a set of C-style functions and one or more opaque pointers called handles. These handles usually represent a library or system resource. Functions are provided to create, manipulate and release the resources using handles. As an example, the CreateEvent function creates an event object, returning a handle to the event object. To release the handle and tell the system you're done using the event object, simply pass the handle to the CloseHandle function. If there are no other outstanding handles to the same event object, the system will destroy it:

```
auto h = CreateEvent( ... );
CloseHandle(h);
```

New to C++

If you're new to C++ 2011, I should point out that the auto keyword tells the compiler to deduce the type of variable from the initialization expression. This is useful when you don't know the type of an expression, as is often the case in metaprogramming, or when you just want to save some keystrokes.

But you should almost never write code like this. Undoubtedly, the single most valuable feature C++ offers is that of the class. Templates are cool, the Standard Template Library (STL) is magical, but without the class nothing else in C++ makes sense. The class is what makes C++ programs succinct and reliable. I'm not talking about virtual functions and inheritance and other fancy features. I'm just talking about a constructor and a destructor. Often that's all you need, and guess what? It doesn't cost you anything. In practice, you need to be aware of the overhead imposed by exception handling, and I'll address that at the end of this column.

To tame the Windows API and make it accessible to modern C++ developers, a class that encapsulates a handle is needed. Yes, your favorite C++ library may already have a handle wrapper, but was it designed from the ground up for C++ 2011? Can you reliably store these handles in an STL container and pass them around your program without losing track of who owns them?

The C++ class is the perfect abstraction for handles. Note I didn't say "objects." Remember that the handle is the object's representative within your program, and is most often not the object itself. The handle is what needs shepherding—not the object. It may sometimes be convenient to have a one-to-one relationship between a Windows API object and a C++ class, but that's a separate issue.

Even though handles are typically opaque, there are still different types of handles and, often, subtle semantic differences that neces-

LEADTOOLS 17.5

LEADTOOLS provides developers easy access to decades of expertise in color, grayscale, document, medical, vector and multimedia imaging.

- OCR/OMR
- BARCODE
- PDF & PDF/A
- MEDICAL 3D
- DICOM
- PACS
- MPEG-2 TRANSPORT STREAM
- ANNOTATIONS
- VIEWER CONTROLS
- FORMS RECOGNITION & PROCESSING
- FILE FORMATS
- VIRTUAL PRINTER
- DOCUMENT CLEANUP & PREPROCESSING
- SCANNING
- DVD & DVR
- DIRECTSHOW CODECS
- MULTIMEDIA PLAYBACK & CAPTURE
- IMAGE PROCESSING
- MEDICAL WORKSTATION FRAMEWORK
- MEDICAL WEB VIEWER FRAMEWORK

MAJOR ADDITIONS INCLUDE

PDF READER AND VIEWER WITH TEXT EXTRACTION, BOOKMARKS AND ANNOTATIONS. A FASTER OCR ENGINE WITH MORE ACCURACY AND LANGUAGES. A NEW HIGH LEVEL BARCODE INTERFACE WITH SUPPORT FOR .NET, SILVERLIGHT AND WINDOWS PHONE. A NEW FRAMEWORK FOR CREATING CLOUD APPLICATIONS.

.NET | **C/C++**

SILVERLIGHT/WINDOWS PHONE

ASP.NET | WPF | WF | WCF

FREE 60 DAY EVALUATION - TRY IT TODAY!!!

LEAD Technologies' newly released **LEADTOOLS Version 17.5** is packed with new features and enhancements delivering more speed, power and extensibility into the hands of application developers than ever before.

800 637-1840

WWW.LEADTOOLS.COM

sitate a class template to adequately wrap handles in a general way. Template parameters are needed to specify the handle type and the specific characteristics or traits of the handle.

In C++, a traits class is commonly used to provide information about a given type. In this way I can write a single class template for handles and provide different traits classes for the different types of handles in the Windows API. A handle's traits class also needs to define how a handle is released so that the handle class template can automatically release it if needed. As such, here's a traits class for event handles:

```
struct handle_traits
{
  static HANDLE invalid() throw()
  {
    return nullptr;
  }

  static void close(HANDLE value) throw()
  {
    CloseHandle(value);
  }
};
```

Because many libraries in the Windows API share these semantics, they can be used for more than just event objects. As you can see, the traits class consists only of static member functions. The result is that the compiler can easily inline the code and no overhead is introduced, while providing a great deal of flexibility for metaprogramming.

The invalid function returns the value of an invalid handle. This is usually a *nullptr*, a new keyword in C++ 2011 representing a null pointer value. Unlike traditional alternatives, nullptr is strongly typed so that it works well with templates and function overloading. There are cases where an invalid handle is defined as something other than nullptr, so the inclusion of the invalid function in the traits class exists for that. The close function encapsulates the mechanism by which the handle is closed or released.

Given the outline of the traits class, I can go ahead and start defining the handle class template, as shown in **Figure 1**.

Figure 1 **The Handle Class Template**

```
template <typename Type, typename Traits>
class unique_handle
{
  unique_handle(unique_handle const &);
  unique_handle & operator=(unique_handle const &);

  void close() throw()
  {
    if (*this)
    {
      Traits::close(m_value);
    }
  }

  Type m_value;

public:

  explicit unique_handle(Type value = Traits::invalid()) throw() :
    m_value(value)
  {
  }

  ~unique_handle() throw()
  {
    close();
  }
```

I've named it unique_handle because it's similar in spirit to the standard unique_ptr class template. Many libraries also use identical handle types and semantics, so it makes sense to provide a typedef for the most commonly used case, simply called handle:

```
typedef unique_handle<HANDLE, handle_traits> handle;
```

I can now create an event object and "handle" it as follows:

```
handle h(CreateEvent( ... ));
```

I've declared the copy constructor and copy assignment operator as private and left them unimplemented. This prevents the compiler from automatically generating them, as they're rarely appropriate for handles. The Windows API allows certain types of handles to be copied, but this is a very different concept from C++ copy semantics.

Unlike traditional alternatives, nullptr is strongly typed so that it works well with templates and function overloading.

The constructor's value parameter relies on the traits class to provide a default value. The destructor calls the private close member function, which in turn relies on the traits class to close the handle if needed. In this way I have a stack-friendly and exception-safe handle.

But I'm not done yet. The close member function relies on the presence of a Boolean conversion to determine whether the handle needs to be closed. Although C++ 2011 introduces explicit conversion functions, this is not yet available in Visual C++, so I use a common approach to Boolean conversion to avoid the dreaded implicit conversions that the compiler otherwise permits:

```
private:

  struct boolean_struct { int member; };
  typedef int boolean_struct::* boolean_type;

  bool operator==(unique_handle const &);
  bool operator!=(unique_handle const &);

public:

  operator boolean_type() const throw()
  {
    return Traits::invalid() != m_value ? &boolean_struct::member : nullptr;
  }
```

This means I can now simply test whether I have a valid handle, but without allowing dangerous conversions to go unnoticed:

```
unique_handle<SOCKET, socket_traits> socket;
unique_handle<HANDLE, handle_traits> event;

if (socket && event) {} // Are both valid?

if (!event) {} // Is event invalid?

int i = socket; // Compiler error!

if (socket == event) {} // Compiler error!
```

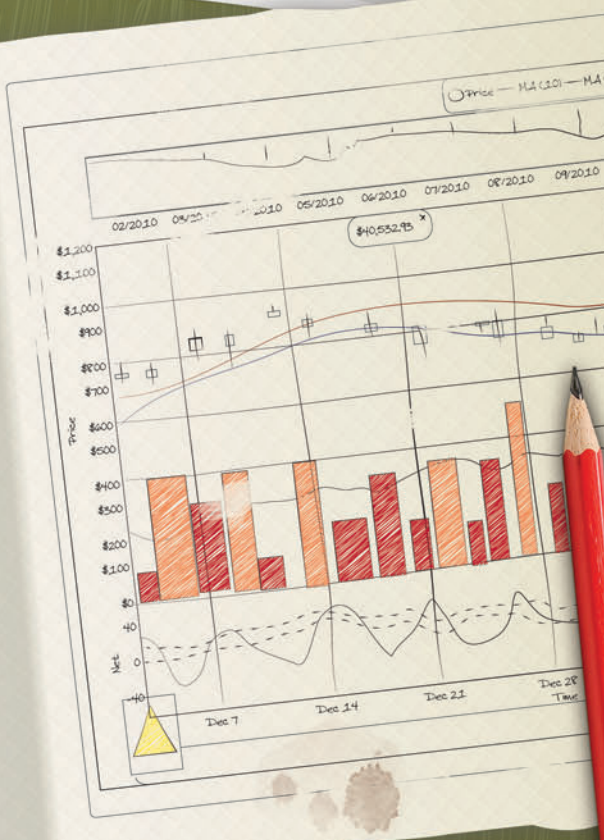
Your Guide to .NET Charts

Ahead of Schedule and Under Budget

With the right tools you can turn your long development list of requirements into a finished solution, from conception to completion. Developers, just like you, continue to turn to ComponentOne Charts™ for their enterprise solutions. Get rock solid, advanced charts for all .NET platforms; download ComponentOne Studio® Enterprise today.

Requirements:

- ✓ Financial & scientific charts
- ✓ Stacked charts
- ✓ Interactivity - zoom, animation, drag-and-drop
- ✓ Tooltips and markers
- ✓ Trend lines
- ✓ ~~Fast~~ Really fast
- ✓ Live updates
- ✓ 2D & 3D
- ✓ Multiple platforms - Silverlight, WPF, ASP.NET and WinForms



ComponentOne® Studio® Enterprise

Download your FREE Trial @ c1.ms/fastcharts

ComponentOne

Using the more obvious operator bool would've allowed those last two errors to go unnoticed. This does, however, allow one socket to be compared with another—hence the need to either explicitly implement the equality operators or declare them as private and leave them unimplemented.

The way a unique_handle owns a handle is analogous to the way the standard unique_ptr class template owns an object and manages that object through a pointer. It then makes sense to provide the familiar get, reset and release member functions to manage the underlying handle. The get function is easy:

```
Type get() const throw()
{
  return m_value;
}
```

The reset function is a bit more work, but builds on what I've already discussed:

```
bool reset(Type value = Traits::invalid()) throw()
{
  if (m_value != value)
  {
    close();
    m_value = value;
  }

  return *this;
}
```

I've taken the liberty of changing the reset function slightly from the pattern provided by unique_ptr by returning a bool value indicating whether or not the object has been reset with a valid handle. This comes in handy with error handling, to which I'll return in a moment. The release function should now be obvious:

```
Type release() throw()
{
  auto value = m_value;
  m_value = Traits::invalid();
  return value;
}
```

Figure 2 **Checking Return Value**

```
inline void check_bool(BOOL result)
{
  if (!result)
  {
    throw check_failed(GetLastError());
  }
}

inline void check_bool(bool result)
{
  if (!result)
  {
    throw check_failed(GetLastError());
  }
}

inline void check_hr(HRESULT result)
{
  if (S_OK != result)
  {
    throw check_failed(result);
  }
}

template <typename T>
void check(T expected, T actual)
{
  if (expected != actual)
  {
    throw check_failed(0);
  }
}
```

Copy vs. Move

The final touch is to consider copy versus move semantics. Because I've already banned copy semantics for handles, it makes sense to allow move semantics. This becomes essential if you want to store handles in STL containers. These containers have traditionally relied on copy semantics, but with the introduction of C++ 2011, move semantics are supported.

> Templates are cool, the Standard Template Library is magical, but without the class nothing else in C++ makes sense. The class is what makes C++ programs succinct and reliable.

Without going into a lengthy description of move semantics and rvalue references, the idea is to allow the value of an object to pass from one object to another in a way that's predictable for the developer and coherent for library authors and compilers.

Prior to C++ 2011, developers had to resort to all kinds of complicated tricks to avoid the excessive fondness that the language—and by extension the STL—has for copying objects. The compiler would often create a copy of an object, then immediately destroy the original. With move semantics the developer can declare that an object will no longer be used and its value moved elsewhere, often with as little as a pointer swap.

In some cases the developer needs to be explicit and indicate this; but more often than not the compiler can take advantage of move-aware objects and perform insanely efficient optimizations that were never possible before. The good news is that enabling move semantics for your own classes is straightforward. Just as copying relies on a copy constructor and a copy assignment operator, move semantics relies on a move constructor and a move assignment operator:

```
unique_handle(unique_handle && other) throw() :
  m_value(other.release())
{
}

unique_handle & operator=(unique_handle && other) throw()
{
  reset(other.release());
  return *this;
}
```

The rvalue Reference

C++ 2011 introduces a new kind of reference, called an rvalue reference. It's declared using &&; this is what's being used in the unique_handle members in the preceding code. Although similar

to references of old, now called lvalue references, the new rvalue references exhibit somewhat different rules when it comes to initialization and overload resolution. For now, I'll leave it at that (I'll return to this topic later). The main benefit at this stage of a handle with move semantics is that you can correctly and efficiently store handles in STL containers.

Error Handling

That's it for the unique_handle class template. The final topic this month—and to prepare for the columns ahead—is error handling. We could debate endlessly about the pros and cons of exceptions versus error codes, but if you want to embrace the standard C++ libraries you'll just have to get used to exceptions. Of course, the Windows API uses error codes, so a compromise is needed.

My approach to error handling is to do as little as possible, and write exception-safe code but avoid catching exceptions. If there are no exception handlers, Windows will automatically generate an error report that includes a minidump of the crash that you can debug postmortem. Throw exceptions only when unexpected runtime errors occur and handle everything else with error codes. When an exception is thrown, you know it's either a bug in your code or some catastrophe that's befallen the computer.

The example I like to give is that of accessing the Windows Registry. Failing to write a value to the Registry is usually a symptom of a bigger problem that will be hard to handle sensibly in your program. This should result in an exception. Failing to read a value from the Registry, however, should be anticipated and handled gracefully. This shouldn't result in an exception, but return a bool or enum value indicating whether or why the value couldn't be read.

The Windows API is not particularly consistent with its error handling; that's the result of an API that's evolved over many years. For the most part, the errors are returned either as BOOL or HRESULT values. There are some others, which I tend to handle explicitly by comparing the return value against documented values.

> My approach to error handling is to do as little as possible.

If I decide a given function call must succeed for my program to continue functioning reliably, I use one of the functions listed in **Figure 2** to check the return value.

There are two things worth mentioning about these functions. The first is that the check_bool function is overloaded so that you can also check the validity of a handle object, which rightly does not allow implicit conversion to BOOL. The second is the check_hr function, which explicitly compares against S_OK rather than using the more common SUCCEEDED macro. This avoids silently accepting other dubious success codes such as S_FALSE, which is almost never what the developer expects.

My first attempt at writing these check functions was a set of overloads. But as I used them in various projects, I realized that the Windows API simply defines far too many result types and macros, so that creating a set of overloads that would work for all of them is simply not possible. Hence the decorated function names. I found a few cases where errors were not being caught due to unexpected overload resolution. The check_failed type being thrown is quite simple:

```
struct check_failed
{
  explicit check_failed(long result) :
    error(result)
  {
  }

  long error;
};
```

I could decorate it with all kinds of fancy features, like adding support for error messages, but what's the point? I include the error value so that I can easily pick it out when performing an autopsy on a crashed application. Beyond that, it's just going to get in the way.

Given these check functions, I can create an event object and signal it, throwing an exception if something goes wrong:

```
handle h(CreateEvent( ... ));

check_bool(h);

check_bool(SetEvent(h.get()));
```

Exception Handling

The other issue with exception handling concerns efficiency. Again, developers are divided, but more often than not because they hold some presupposition not based in reality.

The cost of exception handling arises in two areas. The first is throwing exceptions. This tends to be slower than using error codes, and is one of the reasons you should only throw exceptions when a fatal error occurs. If all goes well, you'll never pay this price.

The second, and more common, cause of performance problems has to do with the runtime overhead of ensuring that the appropriate destructors are called, in the unlikely event an exception is thrown. Code is needed to keep track of which destructors need to be executed; of course, this also increases the size of the stack, which in large code bases can significantly affect performance. Note that you pay this cost whether or not an exception is actually thrown, so minimizing this is essential to ensure good performance.

That means ensuring that the compiler has a good idea of what functions can potentially throw exceptions. If the compiler can prove that there won't be any exceptions from certain functions, it can optimize the code it generates to define and manage the stack. This is why I decorated the entire handle class template and traits class member functions with the exception specification. Although deprecated in C++ 2011, it's an important platform-specific optimization.

That's it for this month. You now have one of the key ingredients for writing reliable programs using the Windows API. Join me next month as I begin to explore the Windows Thread Pool API. ■

KENNY KERR *is a software craftsman with a passion for native Windows development. Reach him at kennykerr.ca.*

Experience the Devexpress Difference

"As a training, mentoring and consulting company, we are often put on the spot as to which vendors we like for .Net tools. There is only one answer from my company and that is Developer Express. We have used Developer Express tools in our projects for the past five years and the company continues to impress me with the quality of their tools.

They simply work. You get what you pay for. Mileage will vary with other vendors but I can assure you Developer Express is a sure bet. "

–Mark Dunn, MCT, MCAD, MCDBA, MCSD .Net

"Our flagship product needed extensive visualizations including charts and graphs. We were looking for a single charting system that could address all of these needs, and handle large volumes of data. It needed to look attractive yet blend into our application.

With *XtraCharts* we were able to create high performance, real-time graphs of performance data through to detailed analytical bar charts. *XtraCharts* was the only option that was fast enough to handle the tens of thousands of data points our customers routinely throw at it. "

–Kendall Miller

Read More User Comments at:
DevExpress.com/Comments

JULIE LERMAN

Demystifying Entity Framework Strategies, Part 3: Classes, Queries and Contexts

This is the third in a series of Data Points columns aimed at helping you make some important decisions when using the Entity Framework as your data access layer in your applications. The first, about model creation workflow, in the May 2011 issue (bit.ly/l0cjPz), discussed choosing among the Code First, Model First and Database First workflows. Code First doesn't use a visual model, but Database First and Model First do. One of the targeted choices in this column will focus on the code generation options when you have a visual model from which you'll create your domain classes. While on the topic of code generation, I'll take a quick look at choosing between using the ObjectContext and DbContext and choosing between LINQ to Entities and Entity SQL.

Generated Classes: EntityObjects or POCOs?

The first version of the Entity Framework (EF hereafter for brevity) relied on the EntityObject class to enable entities to interact with the ObjectContext as it managed relationships and tracked changes to entity instances. The code generation ensured that the classes generated from your model inherited from EntityObject. This is still the default with Visual Studio 2010, but now you have another option. In the Microsoft .NET Framework 4, the EF and its ObjectContext gained the ability to track changes and manage relationships among entities without depending on the EntityObject to send notifications to the ObjectContext. That means your classes no longer have to inherit from EntityObject, which makes a huge difference for developers who are interested in persistence ignorance, separation of concerns, unit testing and other software practices that fall under the generalized umbrella of Agile development.

Classes that don't rely on other APIs are referred to as Plain Old CLR Objects, or POCOs. The EF ability to use these cleaner classes but still perform its change tracking and other entity management tasks is referred to as its "POCO support." This support provides the backbone for Code First. Because the EF is able to work with POCO entities, classes you create in the Code First scenario can also be managed by the EF context.

So if the EF relies on the EntityObject notifying the context of changes to the entities, how is it possible to have POCOs that not only don't inherit from EntityObject, but have no knowledge of the EF? The EF uses two paths to let developers have their proverbial cake and eat it too. One thing that doesn't change is that the ObjectContext still needs to be aware of what classes it's responsible for.

The first path to POCO support results from the ObjectContext getting even smarter as of the .NET Framework 4. Now the context is able to inspect the classes it's managing. It has new methods such as DetectChanges that will read the objects it's managing and then update the state information that it tracks for those objects.

The second way the EF lets you use POCOs while still benefiting from the framework uses a bit of sleight of hand in the form of proxy objects. If every one of the POCO class properties is marked virtual, the EF runtime will create a proxy (wrapper) around the object and that proxy does the same job as the EntityObject. The proxy class will notify the context of property and relationship changes. You can also leverage the proxies without affecting the entire class. The EF will be able to lazy load navigation properties that are marked virtual, even when the other properties are not.

Figure 1 shows a visual representation of the three ways now available for the ObjectContext to keep track of entity changes, using EntityObjects or one of the two POCO mechanisms.

ObjectContext or DbContext?

The EF 4.1 introduced a lightweight version of the ObjectContext called DbContext. It provides all of the same POCO support as the ObjectContext. DbContext also wraps some of the more complex logic required for coding against the ObjectContext into simpler methods and properties, making it easier to execute the most common coding tasks in the EF.

The DbContext is the default context to use with Code First classes, but the ObjectContext is the default with Database First and Model First. Microsoft provides alternate code generation templates for the latter two models. The first is the ADO.NET POCO Entity Generator. This creates POCO classes along with an ObjectContext class to manage them. The second, part of the EF 4.1 installation, is the ADO.NET DbContext Generator. This also creates POCO classes. But the context class that's generated to manage the classes inherits from DbContext. So whichever workflow you begin with—Code First, Model First or Database First—you have the option to use the DbContext if that's your preference. The DbContext has a window into the ObjectContext, so you can get there if needed.

Querying Options: LINQ to Entities or Entity SQL

Another big question developers have about the EF is if they should use LINQ to Entities or Entity SQL to write and execute queries. They also ask why the two exist. Entity SQL was built alongside the Entity Data Model as its native query syntax. LINQ is an extension of C# and Visual Basic and was created by the languages teams. When the

Data Platform group learned about the work being done on LINQ, they knew it would be a natural extension to the querying needs in the EF, so they created the LINQ to Entities implementation.

LINQ to Entities as Your Default Query Strategy

LINQ to Entities is an implementation of LINQ to Objects. LINQ allows you to write queries against strongly typed objects, and in the case of the the EF, you can write queries against your entity classes. LINQ is expressed in two ways. The first is with operators. The query statements look a little like a SQL statement. The query requires an instance of an EntityContainer, which inherits from an ObjectContext, here called context:

```
IQueryable<Family> query =
  from f in context.Families where f.Pets.Any() select f;
```

As you type this expression, IntelliSense helps you with not only the strongly typed classes, but also with the LINQ syntax. The query returns an IQueryable of Family types. The query still needs to be executed, for example with ToList:

```
List<Family> reptileFamilies = query.ToList();
```

This causes the EF to execute a query on the database, grab the results from the database and create objects using those results.

The second way to express a LINQ query is with LINQ methods. These require lambda expressions as their parameters. I'll compress the two previous statements into a single LINQ query:

```
List<Family> reptileFamiles =
  context.Families.Where(f=>f.Pets.Any()).ToList();
```

Because LINQ is so easy to use to express queries—thanks to the strong typing and IntelliSense—I generally recommend that developers plan to use LINQ as their default query strategy. You can express a broad variety of queries with LINQ. Additionally, because LINQ has many implementations, you may already have a good handle on using it. If not, you'll most likely benefit from learning LINQ to Entities and using LINQ to Objects or one of the many other flavors of LINQ to solve other coding problems. Also, you can find a great many resources for learning how to query with LINQ.

Entity SQL for Edge Case Queries and Other Languages

Entity SQL is a string-based query syntax. To execute a query using Entity SQL, you need to use an EF ObjectQuery and pass in the Entity SQL expression. What does that look like? Here's an example:

```
string eSql = "SELECT VALUE f FROM PetsModelContainer.Families AS f";
ObjectQuery<Family> query = context.CreateQuery<Family>(eSql);
List<Family> families = query.ToList();
```

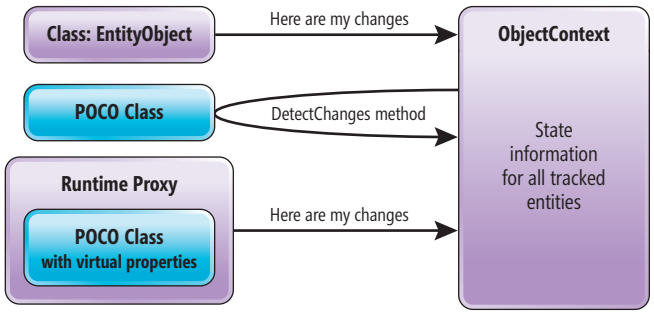


Figure 1 **How Entity Framework Tracks Changes to Entities**

Like the IQueryable you create with LINQ, this ObjectQuery still needs to execute to retrieve the results from the database. ObjectQuery does provide an alternate way to create queries using methods that take snippets of Entity SQL as their parameters.

But with a string expression, there's no strong typing and that query expression doesn't get resolved until run time, which means that you won't discover problems until then. (Note that you can use the indispensable LINQPad—found at linqpad.net—to test Entity SQL as well as LINQ to Entities, outside of Visual Studio.) So why would anyone want to use Entity SQL?

There are a number of reasons.

Let's start with the language you code in. LINQ is part of C# and Visual Basic. There's a power pack for F# that provides LINQ. If you code in any other .NET language, you can't use LINQ. But you can still use Entity SQL to express queries. I've also found Entity SQL useful for building complex search utilities in applications. LINQ is composable, but at a certain point, it just becomes easier to simply build a string.

You can also execute queries at a lower level in the EF using connections and commands along with an Entity SQL expression. This path returns streamed data and is great for reporting or moving data around.

Query, Code Generation and Context Options

While I recommend using LINQ to Entities as your default query strategy, you've seen reasons why you might want to leverage Entity SQL in edge cases. This is what I do and what I recommend to clients, and I'm always happy for an excuse to exercise my infrequently used Entity SQL chops. The documentation on MSDN is pretty thorough for learning how to build Entity SQL expressions. But other than that, some old blog posts from the EF team and a chapter in my book, "Programming Entity Framework" (O'Reilly Media, 2010), I'm not aware of many resources for learning the syntax.

On the code-generation front, using EntityObjects is a fine strategy if you're writing simple applications and want things to just work. However, if you're architecting applications where you want to have persistence-ignorant classes, use unit testing and follow the path of separation of concerns, that's exactly what the EF POCO support was created for. Even then, you have some flexibility in using pure objects that will require a little extra attention when coding or leveraging the proxy generation, which will let the EF do its job with little interference.

The EF 4.1 adds one more option to the code generation, which is choosing between ObjectContext or DbContext to manage your POCO classes. Many developers are jumping to the DbContext because it's a simpler API to work with. If you need a more granular level of control over interacting with the change tracker, or prefer to leverage existing code or knowledge of the ObjectContext, you can continue to use the ObjectContext as the base for your context. ■

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

THANKS to the following technical expert for reviewing this article: *Tim Laverty*

Get the Most out of WebGrid in ASP.NET MVC

Stuart Leeks

Earlier this year Microsoft released ASP.NET MVC version 3 (asp.net/mvc), as well as a new product called WebMatrix (asp.net/webmatrix). The WebMatrix release included a number of productivity helpers to simplify tasks such as rendering charts and tabular data. One of these helpers, WebGrid, lets you render tabular data in a very simple manner with support for custom formatting of columns, paging, sorting and asynchronous updates via AJAX.

In this article, I'll introduce WebGrid and show how it can be used in ASP.NET MVC 3, then take a look at how to really get the most out of it in an ASP.NET MVC solution. (For an overview of WebMatrix—and the Razor syntax that will be used in this article—see Clark Sell's article, "Introduction to WebMatrix," in the April 2011 issue at msdn.microsoft.com/magazine/gg983489).

This article discusses:

- Rendering tabular data using WebGrid
- Adding strong typing and server-side paging
- Updating grid content using AJAX
- Using WebGrid with the WebForms view engine

Technologies discussed:

ASP.NET MVC, WebMatrix, WebGrid

Code Download:

code.msdn.microsoft.com/mag201107WebGrid

This article looks at how to fit the WebGrid component into an ASP.NET MVC environment to enable you to be productive when rendering tabular data. I'll be focusing on WebGrid from an ASP.NET MVC aspect: creating a strongly typed version of WebGrid with full IntelliSense, hooking into the WebGrid support for server-side paging and adding AJAX functionality that degrades gracefully when scripting is disabled. The working samples build on top of a service that provides access to the AdventureWorksLT database via the Entity Framework. If you're interested in the data-access code, it's available in the code download, and you might also want to check out Julie Lerman's article, "Server-Side Paging with the Entity Framework and ASP.NET MVC 3," in the March 2011 issue (msdn.microsoft.com/magazine/gg650669).

> Often when you render a list of items, you want to let users click on an item to navigate to the Details view.

Getting Started with WebGrid

To show a simple example of WebGrid, I've set up an ASP.NET MVC action that simply passes an IEnumerable<Product> to the view. I'm using the Razor view engine for most of this

article, but later I'll also discuss how the WebForms view engine can be used. My ProductController class has the following action:

```
public ActionResult List()
  {
    IEnumerable<Product> model =
      _productService.GetProducts();

    return View(model);
  }
```

The List view includes the following Razor code, which renders the grid shown in **Figure 1**:

```
@model IEnumerable<MsdnMvcWebGrid.Domain.Product>
@{
  ViewBag.Title = "Basic Web Grid";
}
<h2>Basic Web Grid</h2>
<div>
@{
  var grid = new WebGrid(Model, defaultSort:"Name");
}
@grid.GetHtml()
</div>
```

The first line of the view specifies the model type (for example, the type of the Model property that we access in the view) to be IEnumerable<Product>. Inside the div element I then instantiate a WebGrid, passing in the model data; I do this inside an @{...} code block so that Razor knows not to try to render the result. In the constructor I also set the defaultSort parameter to "Name" so theWebGrid knows that the data passed to it is already sorted by Name. Finally, I use @grid.GetHtml() to generate the HTML for the grid and render it into the response.

This small amount of code provides rich grid functionality. The grid limits the amount of data displayed and includes pager links to move through the data; column headings are rendered as links to enable paging. You can specify a number of options in the WebGrid constructor and the GetHtml method in order to customize this behavior. The options let you disable paging and sorting, change the number of rows per page, change the text in the pager links and much more. **Figure 2** shows the WebGrid constructor parameters and **Figure 3** the GetHtml parameters.

> The format parameter of the Column method allows you to customize the rendering of a data item.

The previous Razor code will render all of the properties for each row, but you may want to limit which columns are displayed. There are a number of ways to achieve this. The first (and simplest) is to pass the set of columns to the WebGrid constructor. For example, this code renders just the Name and ListPrice properties:

```
var grid = new WebGrid(Model, columnNames: new[] {"Name", "ListPrice"});
```

You could also specify the columns in the call to GetHtml instead of in the constructor. While this is slightly longer, it has the advantage that you can specify additional information about how to render the columns. In the following example, I specified the header property to make the ListPrice column more user-friendly:

```
@grid.GetHtml(columns: grid.Columns(
 grid.Column("Name"),
 grid.Column("ListPrice", header:"List Price")
 )
)
```

Basic Web Grid

CategoryId	Color	DiscontinuedDate	ListPrice	ModifiedDate	Name	ProductId	ProductModelId	ProductNumber	rowguid	SellEndDate
31			159.0000	11/03/2004 10:01:36	All-Purpose Bike Stand	879	122	ST-1401	c7bb564b-a637-40f5-b21b-cbf7e4f713be	
23	Multi		8.9900	11/03/2004 10:01:36	AWC Logo Cap	712	2	CA-1098	b9ede243-a6f4-4629-b1d4-ffe1aedc6de7	
33			7.9500	11/03/2004 10:01:36	Bike Wash - Dissolver	877	119	CL-9009	3c40b5ad-e328-4715-88a7-ec3220f02acf	
38			25.0000	11/03/2004 10:01:36	Cable Lock	843	115	LO-C100	56ffd7b9-1014-4640-b1bd-b2649589b4d7	30/06/2003 00:00:00
11	Silver		20.2400	11/03/2004 10:01:36	Chain	952	98	CH-0234	5d27e2a5-27ec-4ccb-ba2c-fc980ffe6708	
29	Blue		63.5000	11/03/2004 10:01:36	Classic Vest, L	866	1	VE-C304-L	3211f5a8-b6c4-48bd-9aa4-d69cb40d97dd	
29	Blue		63.5000	11/03/2004 10:01:36	Classic Vest, M	865	1	VE-C304-M	2e52f96e-64a1-4069-911c-e3fd6e094a1e	
29	Blue		63.5000	11/03/2004 10:01:36	Classic Vest, S	864	1	VE-C304-S	eb423ef3-409d-46fe-b35b-d69970820314	
34			21.9800	11/03/2004 10:01:36	Fender Set - Mountain	878	121	FE-6654	e6e76c7f-c145-4cad-a9e8-b1e4e845a2c0	
10	Silver		106.5000	11/03/2004 10:01:36	Front Brakes	948	102	FB-9873	c1813164-1b4b-42d1-9007-4e5f9aee0e19	

1 2 3 4 5 >

Figure 1 **A Basic Rendered Web Grid**

Figure 2 **WebGrid Constructor Parameters**

Name	Type	Notes
source	IEnumerable<dynamic>	The data to render.
columnNames	IEnumerable<string>	Filters the columns that are rendered.
defaultSort	string	Specifies the default column to sort by.
rowsPerPage	int	Controls how many rows are rendered per page (default is 10).
canPage	bool	Enables or disables paging of data.
canSort	bool	Enables or disables sorting of data.
ajaxUpdateContainerId	string	The ID of the grid's containing element, which enables AJAX support.
ajaxUpdateCallback	string	The client-side function to call when the AJAX update is complete.
fieldNamePrefix	string	Prefix for query string fields to support multiple grids.
pageFieldName	string	Query string field name for page number.
selectionFieldName	string	Query string field name for selected row number.
sortFieldName	string	Query string field name for sort column.
sortDirectionFieldName	string	Query string field name for sort direction.

Often when you render a list of items, you want to let users click on an item to navigate to the Details view. The format parameter of the Column method allows you to customize the rendering of a data item. The following code shows how to change the rendering of names to output a link to the Details view for an item, and outputs the list price with two decimal places as typically expected for currency values; the resulting output is shown in **Figure 4**.

```
@grid.GetHtml(columns: grid.Columns(
 grid.Column("Name", format: @<text>@Html.ActionLink((string)item.Name,
             "Details", "Product", new {id=item.ProductId}, null)</text>),
 grid.Column("ListPrice", header:"List Price",
             format: @<text>@item.ListPrice.ToString("0.00")</text>)
 )
)
```

Although it looks like there's some magic going on when I specify the format, the format parameter is actually a Func<dynamic,object>—a delegate that takes a dynamic parameter and returns an object. The Razor engine takes the snippet specified for the format parameter and turns it into a delegate. That delegate takes a dynamic parameter named item, and this is the item variable that's used in the format snippet. For more information on the way these delegates work, see Phil Haack's blog post at bit.ly/h0Q00z.

> Invoking extension methods with dynamic parameters isn't supported.

Figure 3 **WebGrid.GetHtml Parameters**

Name	Type	Notes
tableStyle	string	Table class for styling.
headerStyle	string	Header row class for styling.
footerStyle	string	Footer row class for styling.
rowStyle	string	Row class for styling (odd rows only).
alternatingRowStyle	string	Row class for styling (even rows only).
selectedRowStyle	string	Selected row class for styling.
caption	string	The string displayed as the table caption.
displayHeader	bool	Indicates whether the header row should be displayed.
fillEmptyRows	bool	Indicates whether the table can add empty rows to ensure the rowsPerPage row count.
emptyRowCellValue	string	Value used to populate empty rows; only used when fillEmptyRows is set.
columns	IEnumerable<WebGridColumn>	Column model for customizing column rendering.
exclusions	IEnumerable<string>	Columns to exclude when auto-populating columns.
mode	WebGridPagerModes	Modes for pager rendering (default is NextPrevious and Numeric).
firstText	string	Text for a link to the first page.
previousText	string	Text for a link to the previous page.
nextText	string	Text for a link to the next page.
lastText	string	Text for a link to the last page.
numericLinksCount	int	Number of numeric links to display (default is 5).
htmlAttributes	object	Contains the HTML attributes to set for the element.

Because the item parameter is a dynamic type, you don't get any IntelliSense or compiler checking when writing your code (see Alexandra Rusina's article on dynamic types in the February 2011 issue at msdn.microsoft.com/magazine/gg598922). Moreover, invoking extension methods with dynamic parameters isn't supported. This means that, when calling extension methods, you have to ensure that you're using static types—this is the reason that item.Name is cast to a string when I call the Html.ActionLink extension method in the previous code. With the range of extension methods used in ASP.NET MVC, this clash between dynamic and extension methods can become tedious (even more so if you use something like T4MVC: bit.ly/9GMoup).

Figure 4 **A Basic Grid with Custom Columns**

Adding Strong Typing

While dynamic typing is probably a good fit for WebMatrix, there are benefits to strongly typed views. One way to achieve this is to create a derived type WebGrid<T>, as shown in **Figure 5**. As you can see, it's a pretty lightweight wrapper!

So what does this give us? With the new WebGrid<T> implementation, I've added a new Column method that takes a Func<T, object> for the format parameter, which means that the cast isn't required when calling extension methods. Also, you now get IntelliSense and compiler checking (assuming that MvcBuildViews is turned on in the project file; it's turned off by default).

The Grid extension method allows you to take advantage of the compiler's type inference for generic parameters. So, in this example, you can write Html.Grid(Model) rather than new WebGrid<Product>(Model). In each case, the returned type is WebGrid<Product>.

> With the range of extension methods used in ASP.NET MVC, the clash between dynamic and extension methods can become tedious.

Adding Paging and Sorting

You've already seen that WebGrid gives you paging and sorting functionality without any effort on your part. You've also seen how to configure the page size via the rowsPerPage parameter (in the constructor or via the Html.Grid helper) so that the grid will automatically show a single page of data and render the paging controls to allow navigation between pages. However, the default behavior may not be quite what you want. To illustrate this, I've added code to render the number of items in the data source after the grid is rendered, as shown in **Figure 6**.

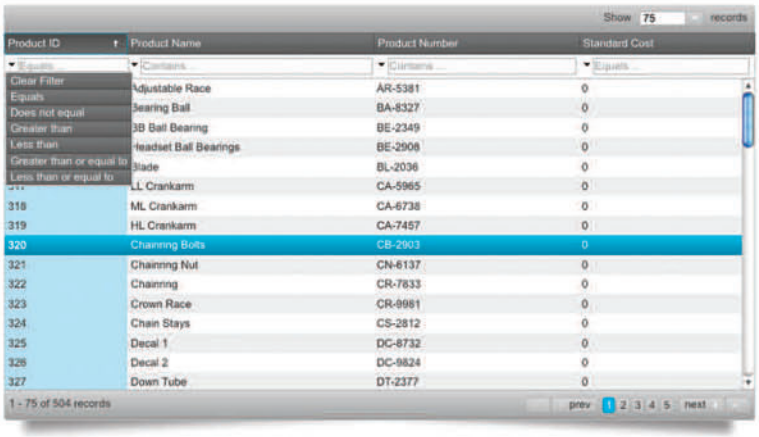
As you can see, the data we're passing contains the full list of products (295 of them in this example, but it's not hard to imagine scenarios with even more data being retrieved). As the amount of data returned increases, you place more and more load on your services and databases, while still rendering the same single page of data. But there's a better approach: server-side paging. In this case, you pull back only the data needed to display the current page (for instance, only five rows).

The first step in implementing server-side paging for WebGrid is to limit the data retrieved from the data source. To do this, you need to know which page is being requested so you can retrieve

Figure 5 **Creating a Derived WebGrid**

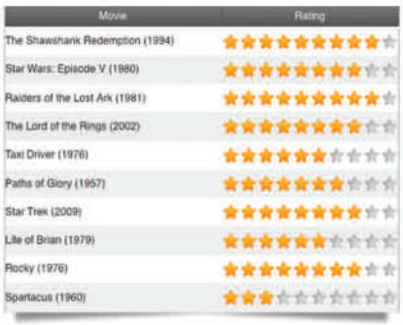
```
public class WebGrid<T> : WebGrid
  {
    public WebGrid(
      IEnumerable<T> source = null,
      ... parameter list omitted for brevity)
    : base(
      source.SafeCast<object>(),
      ... parameter list omitted for brevity)
    { }
  public WebGridColumn Column(
              string columnName = null,
              string header = null,
              Func<T, object> format = null,
              string style = null,
              bool canSort = true)
    {
      Func<dynamic, object> wrappedFormat = null;
      if (format != null)
      {
        wrappedFormat = o => format((T)o.Value);
      }
      WebGridColumn column = base.Column(
                    columnName, header,
                    wrappedFormat, style, canSort);
      return column;
    }
    public WebGrid<T> Bind(
           IEnumerable<T> source,
           IEnumerable<string> columnNames = null,
           bool autoSortAndPage = true,
           int rowCount = -1)
    {
      base.Bind(
           source.SafeCast<object>(),
           columnNames,
           autoSortAndPage,
           rowCount);
      return this;
    }
  }

  public static class WebGridExtensions
  {
    public static WebGrid<T> Grid<T>(
             this HtmlHelper htmlHelper,
             ... parameter list omitted for brevity)
    {
      return new WebGrid<T>(
        source,
        ... parameter list omitted for brevity);
    }
  }
```

NetAdvantage®
for jQuery



IG GRID ALL FEATURES ENABLED

The grid's sophisticated filtering and sorting features allow your users to find the records they need to work with.

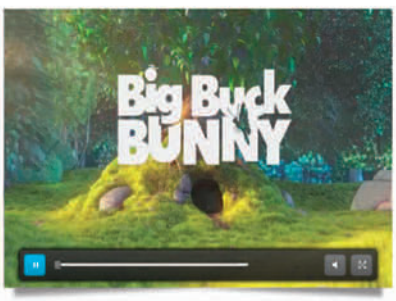


RATING

Enable a social experience for users, by letting them rate their favorite movies or anything else with our rating control.

IG GRID ON HTML PAGE

High performance grid lets users page through any OData source, even streaming movie descriptions.

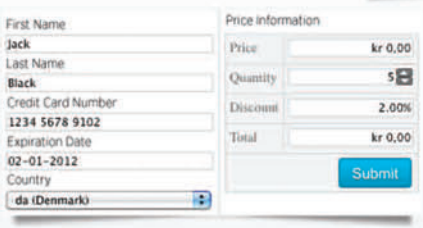


IG VIDEO PLAYER MVC

When a user finds what they want to watch, our HTML5 video player adds streaming video right into your own apps.

IG EDITORS

Robust data entry and editing controls assist users with entering data in the formats your application needs.



UPLOADERS

Accept file uploads of any kind of content straight from your users with background file transfers.

SCAN HERE
for an exclusive
look at jQuery!
www.infragistics.com/jq

TAKE YOUR WEB APPLICATIONS TO THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/JQUERY

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

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

the correct page of data. When WebGrid renders the paging links, it reuses the page URL and attaches a query string parameter with the page number, such as http://localhost:27617/Product/DefaultPagingAndSorting?page=3 (the query string parameter name is configurable via the helper parameters—handy if you want to support pagination of more than one grid on a page). This means you can take a parameter called page on your action method and it will be populated with the query string value.

If you just modify the existing code to pass a single page worth of data to WebGrid, WebGrid will see only a single page of data. Because it has no knowledge that there are more pages, it will no longer render the pager controls. Fortunately, WebGrid has another method, named Bind, that you can use to specify the data. As well as accepting the data, Bind has a parameter that takes the total row count, allowing it to calculate the number of pages. In order to use this method, the List action needs to be updated to retrieve the extra information to pass to the view, as shown in **Figure 7**.

Figure 6 **The Number of Items in the Data Source**

> The first step in implementing server-side paging for WebGrid is to limit the data retrieved from the data source.

With this additional information, the view can be updated to use the WebGrid Bind method. The call to Bind provides the data to render and the total number of rows, and also sets the autoSortAndPage parameter to false. The autoSortAndPage parameter instructs WebGrid that it doesn't need to apply paging, because the List action method is taking care of this. This is illustrated in the following code:

```
<div>
@{
  var grid = new WebGrid<Product>(null, rowsPerPage: Model.PageSize,
    defaultSort:"Name");
  grid.Bind(Model.Products, rowCount: Model.TotalRows, autoSortAndPage: false);
}
@grid.GetHtml(columns: grid.Columns(
 grid.Column("Name", format: @<text>@Html.ActionLink(item.Name,
   "Details", "Product", new { id = item.ProductId }, null)</text>),
  grid.Column("ListPrice", header: "List Price",
    format: @<text>@item.ListPrice.ToString("0.00")</text>)
 )
)

</div>
```

With these changes in place, WebGrid springs back to life, rendering the paging controls but with the paging happening in the service rather than in the view! However, with autoSortAndPage turned off, the sorting functionality is broken. WebGrid uses query string parameters to pass the sort column and direction, but we instructed it not to perform the sorting. The fix is to add the sort and sortDir parameters to the action method and pass these through to the service so that it can perform the necessary sorting, as shown in **Figure 8**.

AJAX: Client-Side Changes

WebGrid supports asynchronously updating the grid content using AJAX. To take advantage of this, you just have to ensure the div that contains the grid has an id, and then pass this id in the ajaxUpdateContainerId parameter to the grid's constructor. You also need a reference to jQuery, but that's already included in the layout view. When the ajaxUpdateContainerId is specified, WebGrid modifies its behavior so that the links for paging and sorting use AJAX for the updates:

```
<div id="grid">

@{
  var grid = new WebGrid<Product>(null, rowsPerPage: Model.PageSize,
  defaultSort: "Name", ajaxUpdateContainerId: "grid");
  grid.Bind(Model.Products, autoSortAndPage: false, rowCount: Model.TotalRows);
}
@grid.GetHtml(columns: grid.Columns(
 grid.Column("Name", format: @<text>@Html.ActionLink(item.Name,
   "Details", "Product", new { id = item.ProductId }, null)</text>),
 grid.Column("ListPrice", header: "List Price",
   format: @<text>@item.ListPrice.ToString("0.00")</text>)
 )
)

</div>
```

While the built-in functionality for using AJAX is good, the generated output doesn't function if scripting is disabled. The reason for this is that, in AJAX mode, WebGrid renders anchor tags with the href set to "#," and injects the AJAX behavior via the onclick handler.

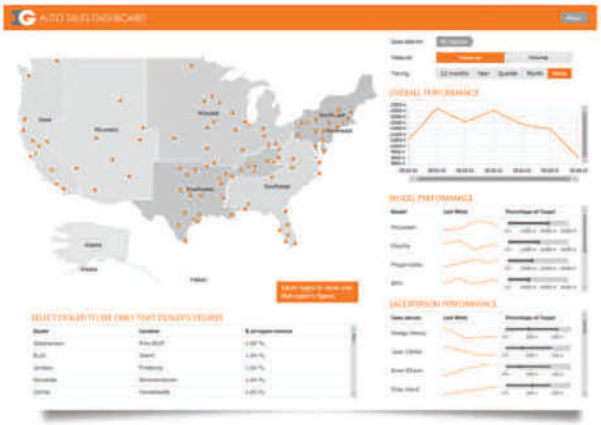
I'm always keen to create pages that degrade gracefully when scripting is disabled, and generally find that the best way to achieve this is through *progressive enhancement* (basically having a page that functions without scripting that's enriched with the addition of scripting). To achieve this, you can revert back to the non-AJAX WebGrid and create the script in **Figure 9** to reapply the AJAX behavior:

To allow the script to be applied just to a WebGrid, it uses jQuery selectors to identify elements with the ajaxGrid class set. The script establishes click handlers for the sorting and paging links (identified via the table header or footer inside the grid container) using the

Figure 7 **Updating the List Action**

```
public ActionResult List(int page = 1)
{
  const int pageSize = 5;

  int totalRecords;
  IEnumerable<Product> products = productService.GetProducts(
    out totalRecords, pageSize:pageSize, pageIndex:page-1);

  PagedProductsModel model = new PagedProductsModel
                             {
                               PageSize= pageSize,
                               PageNumber = page,
                               Products = products,
                               TotalRows = totalRecords
                             };
  return View(model);
}
```

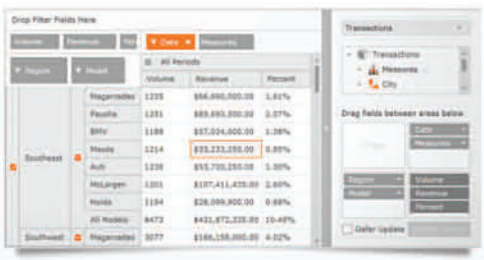
NetAdvantage®
for Data Visualization



INTERACTIVE DASHBOARDS
Build rich dashboards that visualize business data to empower decision makers.

MAP OUT ANYTHING AND EVERYTHING
Use xamMap™ to get an instant view and show anything including seating charts, floor plans, warehouse contents, and yes—geographic maps, too!

MEDIA TIMELINE
The timeline highlights how easily users can search, browse and play popular videos from YouTube.



OLAP PIVOT GRID DATA VISUALIZATION
Work with multidimensional data from your OLAP cubes, data warehouses and Microsoft® SQL Server® Analysis Services.

INFRAGISTICS MOTION FRAMEWORK™
Create an immersive and animated user experience that tells the story of your data over time like no other visualization can.

SCAN HERE
for an exclusive
look at Data Visualization!
www.infragistics.com/dvis

TAKE YOUR APPLICATIONS TO
THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/DV

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

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

Figure 8 **Adding Sorting Parameters to the Action Method**

```
public ActionResult List(
          int page = 1,
          string sort = "Name",
          string sortDir = "Ascending" )
{
  const int pageSize = 5;

  int totalRecords;
  IEnumerable<Product> products =
    _productService.GetProducts(out totalRecords,
                                pageSize: pageSize,
                                pageIndex: page - 1,
                                sort:sort,
                                sortOrder:GetSortDirection(sortDir)
                                );

  PagedProductsModel model = new PagedProductsModel
  {
    PageSize = pageSize,
    PageNumber = page,
    Products = products,
    TotalRows = totalRecords
  };
  return View(model);
}
```

jQuery live method (api.jquery.com/live). This sets up the event handler for existing and future elements that match the selector, which is handy given the script will be replacing the content.

The updateGrid method is set as the event handler and the first thing it does is to call preventDefault to suppress the default behavior. After that it gets the URL to use (from the href attribute on the anchor tag) and then makes an AJAX call to load the updated content into the container element. To use this approach, ensure that the default WebGrid AJAX behavior is disabled, add the ajaxGrid class to the container div and then include the script from **Figure 9**.

AJAX: Server-Side Changes

One additional point to call out is that the script uses functionality in the jQuery load method to isolate a fragment from the returned document. Simply calling load('http://example.com/someurl') will load the contents of the URL. However, load('http://example.com/someurl #someId') will load the content from the specified URL and then return the fragment with the id of "someId." This mirrors the default AJAX behavior of WebGrid and means that you don't have to update your server code to add partial rendering behavior; WebGrid will load the full page and then strip out the new grid from it.

In terms of quickly getting AJAX functionality this is great, but it means you're sending more data over the wire than is necessary, and potentially looking up more data on the server than you need to as well. Fortunately, ASP.NET MVC makes dealing with this pretty simple. The basic idea is to extract the rendering that you want to share in the AJAX and non-AJAX requests into a partial view. The List action in the controller can then either render just the partial view for AJAX calls or the full view (which in turn uses the partial view) for the non-AJAX calls.

Figure 9 **Reapplying the AJAX Behavior**

```
$(document).ready(function () {

  function updateGrid(e) {
    e.preventDefault();
    var url = $(this).attr('href');
    var grid = $(this).parents('.ajaxGrid');
    var id = grid.attr('id');
    grid.load(url + ' #' + id);
  };
  $('.ajaxGrid table thead tr a').live('click', updateGrid);
  $('.ajaxGrid table tfoot tr a').live('click', updateGrid);
 });
```

The approach can be as simple as testing the result of the Request.IsAjaxRequest extension method from inside your action method. This can work well if there are only very minor differences between the AJAX and non-AJAX code paths. However, often there are more significant differences (for example, the full rendering requires more data than the partial rendering). In this scenario you'd probably write an AjaxAttribute so you could write separate methods and then have the MVC framework pick the right method based on whether the request is an AJAX request (in the same way that the HttpGet and HttpPost attributes work). For an example of this, see my blog post at bit.ly/eMIlxU.

WebGrid and the WebForms View Engine

So far, all of the examples outlined have used the Razor view engine. In the simplest case, you don't need to change anything to use WebGrid with the WebForms view engine (aside from differences in view engine syntax). In the preceding examples, I showed how you can customize the rendering of row data using the format parameter:

```
grid.Column("Name",
  format: @<text>@Html.ActionLink((string)item.Name,
  "Details", "Product", new { id = item.ProductId }, null)</text>),
```

The format parameter is actually a Func, but the Razor view engine hides that from us. But you're free to pass a Func—for example, you could use a lambda expression:

```
grid.Column("Name",
  format: item => Html.ActionLink((string)item.Name,
  "Details", "Product", new { id = item.ProductId }, null)),
```

Armed with this simple transition, you can now easily take advantage of WebGrid with the WebForms view engine!

> WebGrid supports asynchronously updating the grid content using AJAX.

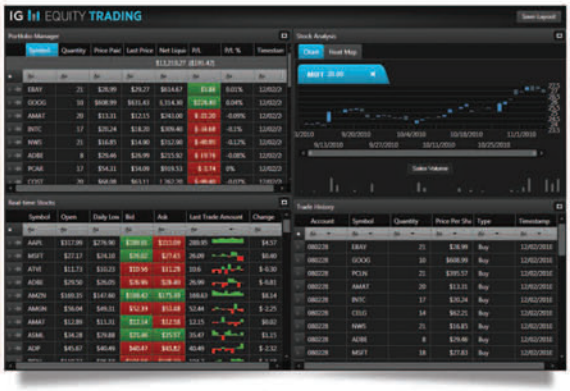
Wrapping Up

In this article I showed how a few simple tweaks let you take advantage of the functionality that WebGrid brings without sacrificing strong typing, IntelliSense or efficient server-side paging. WebGrid has some great functionality to help make you productive when you need to render tabular data. I hope this article gave you a feel for how to make the most of it in an ASP.NET MVC application. ■

STUART LEEKS *is an application development manager for the Premier Support for Development team in the United Kingdom. He has an unhealthy love of keyboard shortcuts. He maintains a blog at blogs.msdn.com/stuartleeks where he talks about technical topics that interest him (including, but not limited to, ASP.NET MVC, Entity Framework and LINQ).*

THANKS to the following technical experts for reviewing this article: *Simon Ince and Carl Nolan*

NetAdvantage® PERFORMANCE

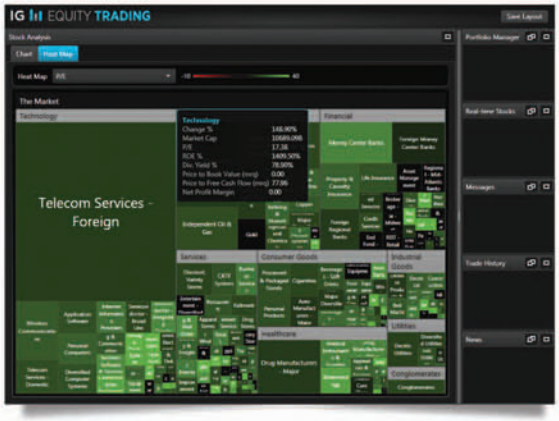


REAL-TIME USER INTERFACES

Refresh your app's user interface tick-by-tick on every value change in the fastest applications imaginable.

ACROSS ALL PLATFORMS

Charts and grids featuring blazing speed whether you are using ASP.NET, Silverlight, WPF or any other .NET platform.



DEEP DRILLDOWN

Dive into the deepest details of your data with crisp, rapidly-updating visual controls designed to handle it all.

TAKE IT ON THE ROAD

Our jQuery controls for iPhone/iPad/Android and Windows Phone 7 support means your UI is always on the go!

SCAN HERE
for an exclusive
look at Performance!
www.infragistics.com/perf

TAKE YOUR APPLICATIONS TO
THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/PERFORMANCE

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

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

Build Workflow Solutions for SharePoint Online

Chris Mayo

Many organizations have adopted SharePoint to facilitate collaboration among their information workers. These organizations often use SharePoint to store information in lists and document libraries to support a manual business process. While storing this information in SharePoint does make it easier to collaborate around that information much more effectively, a significant boost in information worker productivity can be realized by automating those business processes within SharePoint in the form of SharePoint workflows.

With the release of Office 365, SharePoint Online gives organizations many of the same collaboration benefits as SharePoint while also providing the advantages of a cloud-based Software as a Service. SharePoint Online supports workflow through declarative workflows built in SharePoint Designer 2010 and deployed through sandbox solutions. If the built-in workflow actions don't support the requirements of the workflow solution, you can build custom workflow actions using Visual Studio 2010 and deploy them to SharePoint Online via sandboxed solutions.

In this article, I'll provide an overview of workflow support in SharePoint Online, build a declarative workflow using SharePoint Designer 2010, augment that workflow via a custom action and deploy it to run in the cloud as a sandbox solution in SharePoint Online.

This article discusses:

- Developing with SharePoint Designer 2010
- Creating a workflow
- Implementing the workflow in SharePoint Designer
- Building a custom workflow action in Visual Studio 2010
- Deploying the workflow solution to SharePoint Online

Technologies discussed:

SharePoint Online, SharePoint Designer 2010, Visual Studio 2010

Code download available at:

bit.ly/spoworkmsdncode

For information on how SharePoint Online development is similar to and different from SharePoint 2010 development, see my article, "Cloud-Based Collaboration with SharePoint Online," published in the March 2011 issue of *MSDN Magazine* (bit.ly/spodevmsdn). And for more details on SharePoint Online and Office 365, including how to sign up for a trial account, visit office365.com.

SharePoint Online Development Overview

SharePoint Online lets you create collaboration solutions using the same skills and tools used in developing for SharePoint 2010, including Visual Studio 2010, SharePoint Designer 2010, C# or Visual Basic, and the SharePoint APIs and SDKs. While there are a number of similarities between developing solutions for SharePoint 2010 and SharePoint Online, there are differences that will affect what solutions can be built and how those solutions are built.

SharePoint Online only supports sandboxed solutions, which means you can't deploy code-based workflows, including solutions built with the Sequential Workflow and State Machine Workflow project items. However, workflows built using SharePoint Designer 2010 are supported because these workflows are declarative rather than code-based and can be deployed either directly to SharePoint Online or via the Solution Gallery using package files. Furthermore, you can extend these declarative workflows via sandboxed solutions built with Visual Studio 2010 to provide custom workflow actions to support scenarios that SharePoint Designer 2010 doesn't support out of the box.

This article will build on the concepts and solution covered in the previous article. I encourage you to read that article, follow the guidance provided to set up your SharePoint Online development environment and build the purchasing solution examples so you have a solid understanding of the development concepts for SharePoint Online. To illustrate workflow support in SharePoint Online, this article will extend the purchasing solution to include a workflow solution.

SharePoint Designer 2010 Overview

SharePoint Designer 2010 lets power users and developers customize SharePoint 2010 without code. SharePoint Designer 2010 also

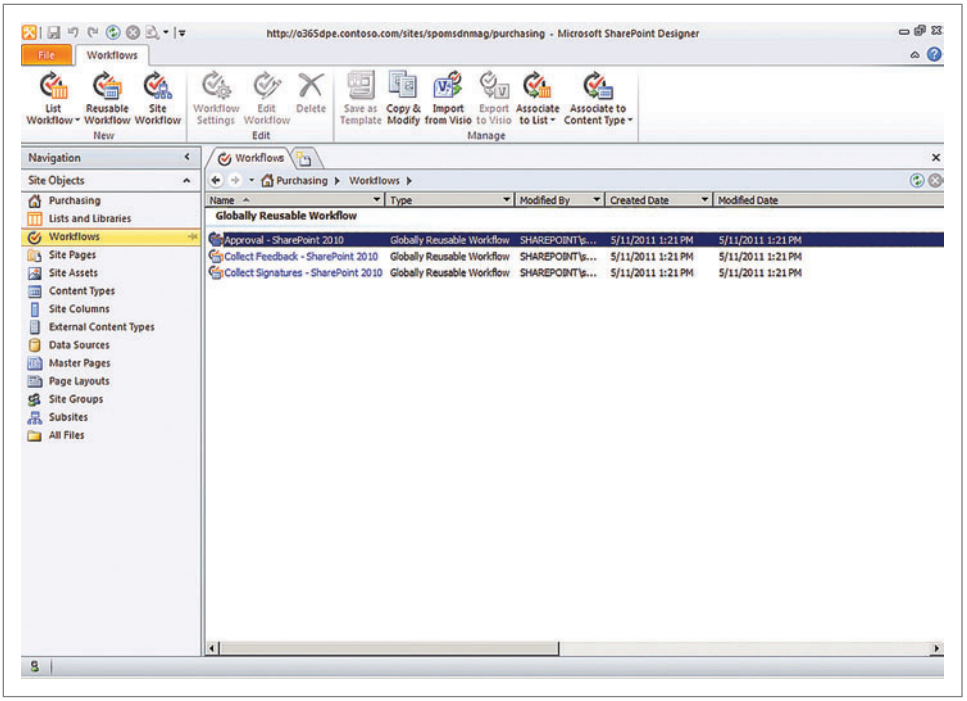


Figure 1 **Workflows in SharePoint Designer 2010**

supports SharePoint Online, with the only differences coming from feature differences between SharePoint 2010 and SharePoint Online, such as BCS and External List support. SharePoint Designer 2010 is a great tool for navigating and managing artifacts in SharePoint, working with data, and for customizing the look and feel of your SharePoint sites. It also lets you create custom workflows that can be deployed as part of an overall collaboration solution. You can learn more about SharePoint Designer 2010 capabilities in the video, "Introducing SharePoint Designer 2010 for SharePoint Online" (bit.ly/spdspointro). SharePoint Designer 2010 is a free download and supports both 32-bit (bit.ly/spd201032) and 64-bit versions (bit.ly/spd201064).

The Purchasing Solution

In the examples in this article, I'll be building on the fictional Contoso Corp. purchasing process introduced in the previous article by adding a SharePoint workflow to automate the approval of purchase requests. When a purchase request requires approval, the user will start the workflow and provide a business justification for the purchase. The workflow will initiate an approval process and create a task for the Approvers group with details on the purchase. After the approval or rejection, an e-mail will be sent to the requesting user about the outcome of the request. The workflow will then examine the request type and in the case of an approved travel request (such as to attend a technical conference), the workflow will create a subsite for the user to fill out a trip report and upload slides. I'm going

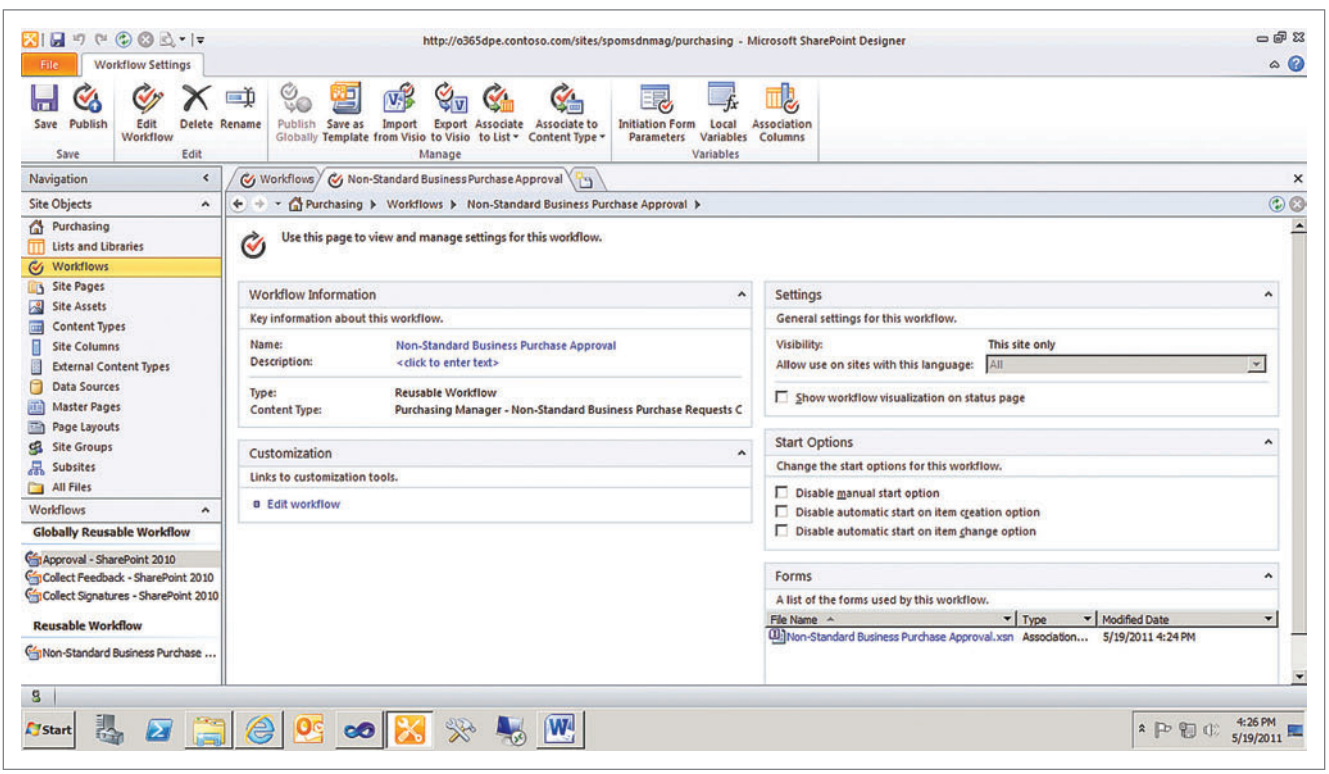


Figure 2 **Reusable Workflow Bound to the Content Type**

Figure 3 **Workflow Variables**

Variable Type	Description
Initiation Form Parameter	A parameter that stores data collected from the user when the workflow is started or associated with a list.
Local Variables	Private variables used to store data used in the processing of the workflow.
Association Columns	Columns that are added to the list when the workflow is associated to the list to guarantee a base set of columns for reusable workflows.

to add a RequestType field to the Non-Standard Business Purchase Requests for this purpose.

I've updated the PurchaseMgr project from the last article to include this RequestType field. If you followed that article, you can remove the prior package and deploy the one supplied with this article or add a required RequestType field with the choices Travel, Equipment and Service Request to the Non-Standard Purchase Requests list. I'll start with the code from this article (bit.ly/spoworkmsdncode) and extract it to the Documents\Visual Studio 2010\Projects\SPOMSDN_Workflow directory on my local machine. I'll then deploy the PurchasingMgr.wsp to the Solution Gallery of the site collection on my local SharePoint 2010 development environment (in my case, http://o365dpe.contoso.com/sites/spomsdnmag) and activate the Purchasing Manager–Content Types and Lists feature in my Purchasing site (http://o365dpe.contoso.com/sites/spomsdnmag/purchasing).

Creating the Workflow

To get started developing my workflow, I'll open the Purchasing site on my local SharePoint 2010 development environment site by opening SharePoint Designer 2010, selecting File | Sites | Open Site and entering the URL of my site (http://o365dpe.contoso.com/sites/spomsdnmag/purchasing). In the Navigation pane, I'll select Workflows to show the currently published workflow and the Workflows ribbon, as shown in **Figure 1**.

SharePoint Online supports List Workflows that act on a specific list, Site Workflows that work on a specific site and Reusable Workflows that can be bound to a list or specific content type at a later date. I want to be able to distribute my workflow as part of an overall solution, so I'll create a Reusable Workflow, which is the only type of workflow that supports distribution.

When creating a workflow in SharePoint Designer, I have a number of choices. I can create my workflow based on one of the built-in workflows (including Approval, Collect Feedback or Collection Signatures) using the Copy & Modify button, import a Visio 2010 diagram based on the Microsoft SharePoint Workflow template using the Import from Visio button, or create the workflow from scratch using the New group on the Workflows tab of the Ribbon. I'll create my workflow from scratch by clicking the Reusable Workflow button in the New group of the ribbon. In the Create Reusable Workflow dialog, I'll name the workflow Non-Standard Business Purchase Request Approval and select Purchasing Manager – Non-Standard Business Purchase Request Content Type so the workflow will be bound to my content type, as shown in **Figure 2**.

Next, I'll go to the Non-Standard Business Purchase Request Approval workflow settings page by clicking the Workflow Settings in the Manage group of the Workflows tab.

In the Settings group, I'll check the "Show workflow visualization on status page" to give my users a real-time visualization of the status of each instance of the workflow. I'll use the Start Options group to make this a human-driven workflow by unchecking "Disable manual start option" and checking the "Disable automatic start on item creation/change option" checkboxes. You can create a machine-driven workflow by doing the reverse.

In many situations, information I need to complete the workflow is not stored in the list or library. I can collect and store that information in workflow-specific variables and columns. This can be done with the Variables group in the Workflows ribbon, as described in **Figure 3**.

In this instance, I want to collect the business justification for the purchase when the workflow is started so I can provide that information to approvers to help them make a decision. To do this, I'll click on Variables | Initiation Form Parameters | Add… and create a Business Rationale parameter in the Add Field dialog; then click Next and Finish.

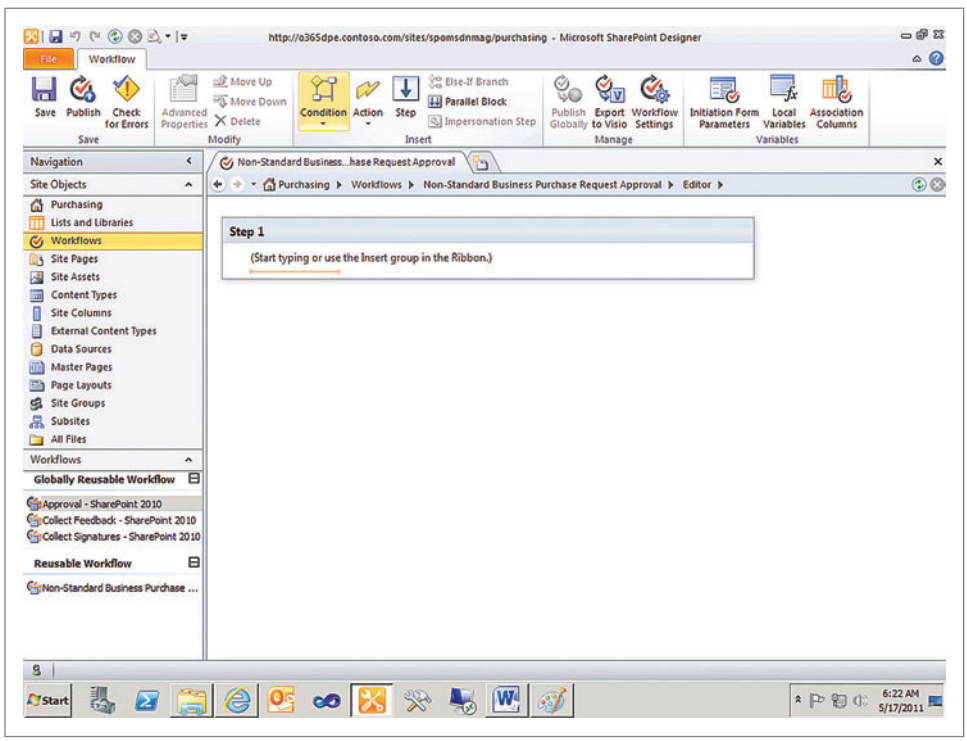


Figure 4 **Workflow Designer—Insert Section**

Figure 5 **Types of Steps**

Step Type	Description
Step	Steps are used to organize Conditions and Actions in a workflow. All conditions and actions must be completed before execution moves to the next Step. Can be nested.
Impersonation Step	Workflows run under the permissions of the user who starts the workflow either manually or automatically. Steps in the Impersonation Step execute as the workflow author. The Impersonation Step can only be added as the first Step in a workflow. Steps can be nested inside the Impersonation Step.
Parallel Block	When Steps are added to a Parallel Block, they are run in parallel rather than sequentially.

Implementing the Workflow in SharePoint Designer

At this point, I'm ready to start implementing the workflow, so I'll select the Editor workflow button in the Edit Group of the Workflow Settings tab. I'm now in the SharePoint Designer 2010 workflow designer where I can implement my workflow using the Condition, Action and Step options in the Insert group of the Workflow ribbon (see **Figure 4**).

Steps are used to organize the conditions and actions in a workflow and to control how those conditions and actions are executed, as **Figure 5** describes.

For example, in **Figure 6**, Step 2 will be executed with the workflow author's permissions while Step 1 will be executed as the user who started the workflow (either manually or automatically). Steps 5 and 6 will be executed in parallel while Steps 3 and 4 will be executed serially.

In my workflow, I'll have just one Step. To document what the Conditions and Actions do in the Step, I'll change the title by clicking on the text "Step 1" and entering "Request Purchase Approval." I'll then click inside the Step so the cursor is where I need to start adding Actions and Conditions.

Actions do work within the workflow and can include starting an approval process, modifying a list item, sending an e-mail and more. For a complete list of the supported actions, refer to the SharePoint Designer 2010 reference (bit.ly/spd2010act). For this scenario, I want to start an approval process for the purchase request, so I'll select the Action drop-down from the Insert section of the ribbon and choose Start Approval Process. This adds a sentence to the designer with hyperlinks, allowing me to complete the Action by selecting each link and providing more information, as shown in **Figure 7**.

Clicking on the Approval link lets me customize the approval task, including changing both how the task is processed and the task outcomes. I can return to the editor by clicking the back button. I'll click the "these users" link to customize the task and specify who will participate in the approval process. In the Select Task Process Participants dialog, I'll select the Approvers group as the Participants and set the title of the task, as shown in **Figure 8**. For the Instructions, I want to provide details about the item and the business rationale to help the approver make a decision. For example, to include the list item title, I click the Add or Change Lookup button, select Current Item as the Data Source and Title as the Field from Source.

The final task will look like **Figure 9**.

At this point in the workflow execution, the workflow will pause until the task is approved or rejected or, if a due date was supplied, expired. To control the flow, I'll use Conditions.

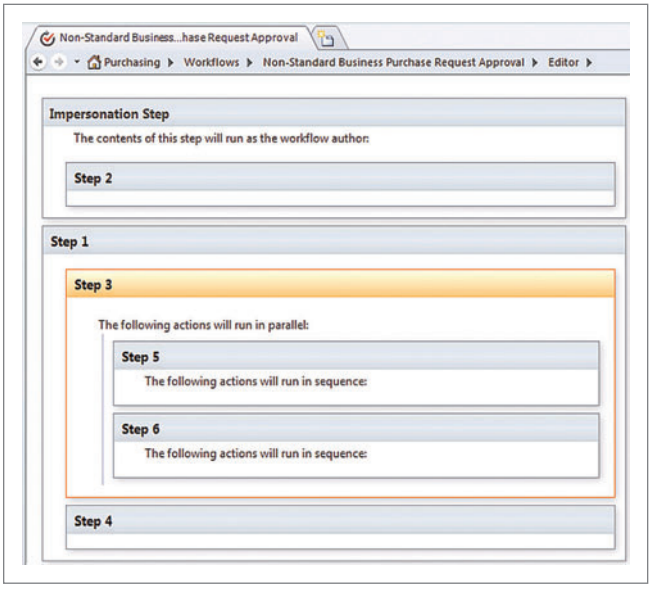


Figure 6 **Steps—Effect on Workflow Execution**

Conditions control the logic of the workflow based on values in workflow variables or fields. For example, if the item is approved, I want to send an e-mail to the requestor and create a site if the request type is of type Travel. If the item is rejected, I want to send e-mail and delete the item. To do this, I'll place the cursor under the approval process, click the Condition drop-down and select "If any value equals any value." I'll click on the "value" hyperlink and click the button displayed to show the Define Workflow Lookup dialog. For the Data source I'll select Workflow Variables and Parameters, and for the Field from source I choose Variable: IsItemApproved. This variable is added to the workflow when a Start Approval Process task is added.

I'll then click the equals link and select equals from the drop-down displayed. Next, I'll click the value link and select Yes from the drop-down. I'll place an Else-If Block under this condition so I can take action when the item is rejected. I'll add Send and Email actions to each branch and use the String Builder to set the e-mail title, as shown in **Figure 10**.

Next, I'll add an "If current item field equals value" condition under the If condition and base it on the RequestType field so I can create a new sub-site when the request is a travel request. The designer now looks like **Figure 11**.

To create the site for Travel requests, I'll need to create a custom workflow action because this action isn't built into SharePoint Designer 2010. To do this, I'll save my workflow, close SharePoint Designer 2010 and open Visual Studio 2010.

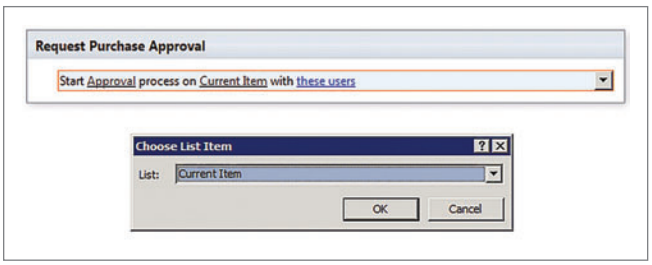


Figure 7 **Adding the Start Approval Process Action**

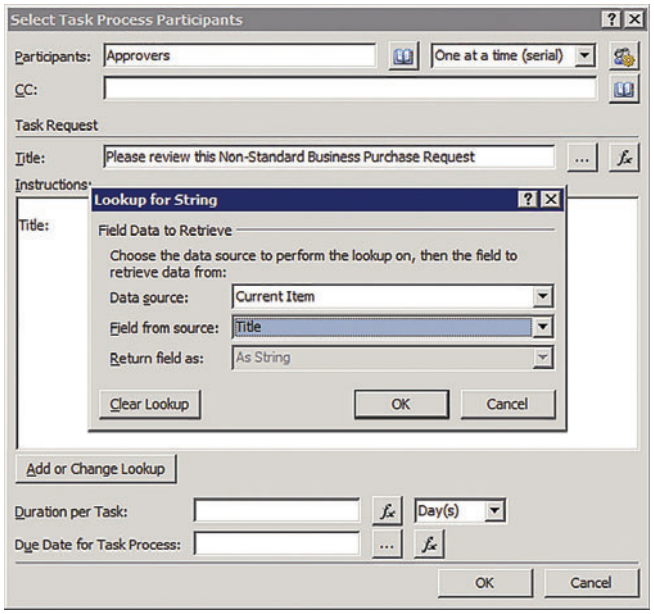


Figure 8 **Customizing the Approval Task**

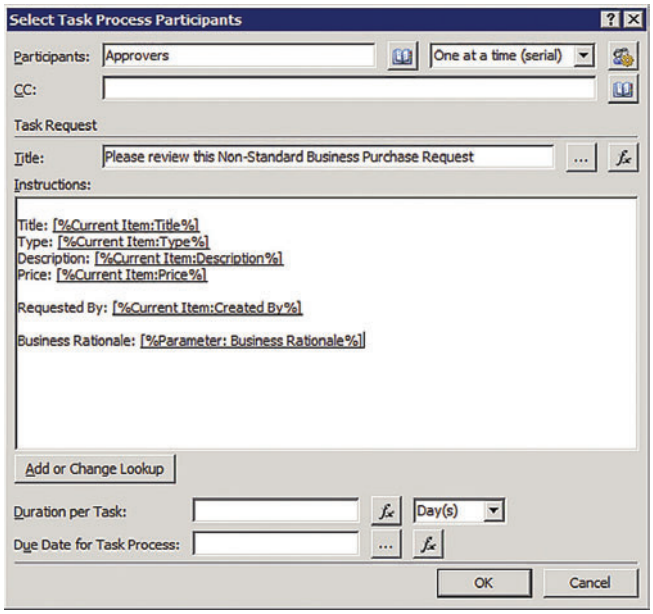


Figure 9 **Customized Approval Task**

Building a Custom Workflow Action in Visual Studio 2010

You can add custom workflow actions to SharePoint Designer 2010 using Visual Studio 2010 and sandboxed solutions. These actions can accomplish any task that can be done in a sandboxed solution, which provides a lot of flexibility to SharePoint Online workflows.

In Visual Studio 2010, I'll select File | New Project and in the New Project dialog I'll select SharePoint 2010 templates and the Empty SharePoint Template. I'll enter PurchasingMgrActions for the Name and Documents\Visual Studio 2010\Projects\SPOMSDN_Workflow\ as the Location and Click OK. In the SharePoint Customization Wizard I'll enter my local development site (http://o365dpe.contoso.com/sites/spomsdnmag/purchasing) for the local site and select "Deploy as a sandboxed solution" and click Finish.

In the Solution Explorer, I'll right click on the PurchasingMgrActions project, select Add | Class…, then name the class CreateSiteAction and click OK. The CreateSiteAction class provides the method the workflow will call to create the new site. I'll add the necessary using statements and implement the CreateSite method, as defined in **Figure 12**.

The CreateSite method follows the method signature for custom actions, passing in SPUserCodeWorkflowContext to provide access to the context the workflow is running under and whatever other parameters I need (in this case the name of the site to create). The method gets access to the site collection (SPSite) and site (SPWeb) via the context and creates the new site via the SPWeb.Webs.Add method. Results are returned via the Results Hashtable variable.

For CreateSiteAction to be added to the Actions drop-down, I need to deploy an Elements.xml file with my Feature to describe the action to SharePoint Designer 2010. I'll add this file to the project by selecting the PurchasingMgrActions project in Solution Explorer, right-clicking and choosing Add | New Item…, then selecting SharePoint 2010 under Installed Templates and choosing the Empty Element template. I'll name the item CreateSiteActionDefinition and click OK.

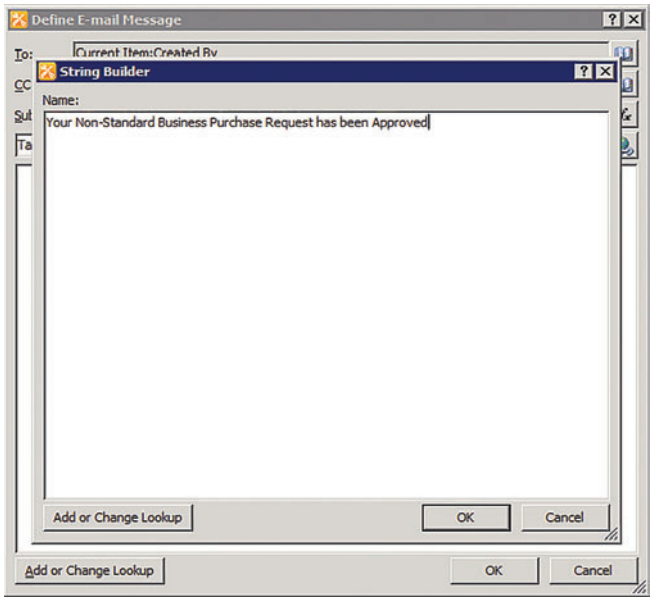


Figure 10 **Approval E-mail**

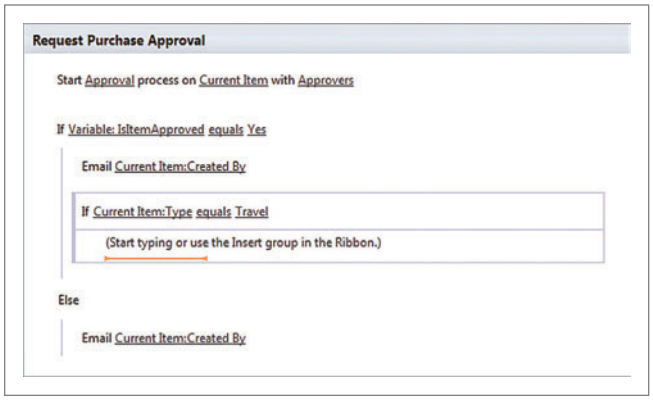


Figure 11 **The Conditional Flow of the Workflow**

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

Figure 12 **Implementing the CreateSite Method**

```
using System;
using System.Collections;

using Microsoft.SharePoint;
using Microsoft.SharePoint.UserCode;

namespace SPDCustomWorkflowActions
{
  public class CreateSiteActivity
  {
    public Hashtable CreateSite(SPUserCodeWorkflowContext context, string siteName)
    {
      Hashtable results = new Hashtable();
      try
      {
        using (SPSite site = new SPSite(context.CurrentWebUrl))
        {
          using (SPWeb web = site.OpenWeb())
          {
            web.Webs.Add(
              siteName,
              "Trip Report: " + siteName,
              string.Empty,
              1033,
              "STS",
              false,
              false);
          }
        }

        results["success"] = true;
        results["exception"] = string.Empty;
      }
      catch (Exception e)
      {
        results = new Hashtable();
        results["exception"] = e.ToString();
        results["success"] = false;
      }

      return results;
    }
  }
}
```

Figure 13 **Defining the Create Site Action via Elements.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <WorkflowActions>
    <Action Name="Create Site"
                    SandboxedFunction="true"
                    Assembly="$SharePoint.Project.AssemblyFullName$"
                    ClassName="PurchasingMgrActions.CreateSiteAction"
                    FunctionName="CreateSite"
                    AppliesTo="all"
                    UsesCurrentItem="true"
                    Category="Purchasing Manager Workflow Actions">
      <RuleDesigner Sentence="Create Site with name %1 (exceptions logged to %2)">
        <FieldBind Field="siteName" Text="Site Name" Id="1" DesignerType="TextBox" />
        <FieldBind Field="exception" Text="Exception" Id="2"
          DesignerType="ParameterNames" />
      </RuleDesigner>
      <Parameters>
        <Parameter Name="__Context"
                    Type="Microsoft.SharePoint.WorkflowActions.WorkflowContext,
                      Microsoft.SharePoint.WorkflowActions"
                    Direction="In"
                    DesignerType="Hide" />
        <Parameter Name="siteName"
                    Type="System.String, mscorlib"
                    Direction="In"
                    DesignerType="TextBox"
                    Description="Name of the site to create" />
        <Parameter Name="exception"
                    Type="System.String, mscorlib"
                    Direction="Out"
                    DesignerType="ParameterNames"
                    Description="Exception encountered"/>
      </Parameters>
    </Action>
  </WorkflowActions>
</Elements>
```

This definition is accomplished by implementing an Action element in the WorkflowActions elements, as seen in **Figure 13**.

The Action element and its attributes describe the assembly, class and method that will be called when the action is executed in the workflow. In this case, it will call the CreateSiteAction.CreateSite method. The RuleDesigner element and its FieldBind elements define the sentence that will be shown in the workflow designer and the name and type of the fields shown as hyperlinks in that sentence. The Parameters element and its Parameter subelements define how the RuleDesigner\FieldBind elements get passed in and out of the call to CreateSiteAction.CreateSite. For example, the __Context parameter is of type WorkflowContext and is used to pass that context into the call without being shown in the designer (by setting the DesigerType attribute to "Hide"). The siteName parameter will receive the value in the siteName field binding. This is done by giving the field and parameter the same name. Exceptions will be passed out via the exception parameter and into the exception field for the same reason.

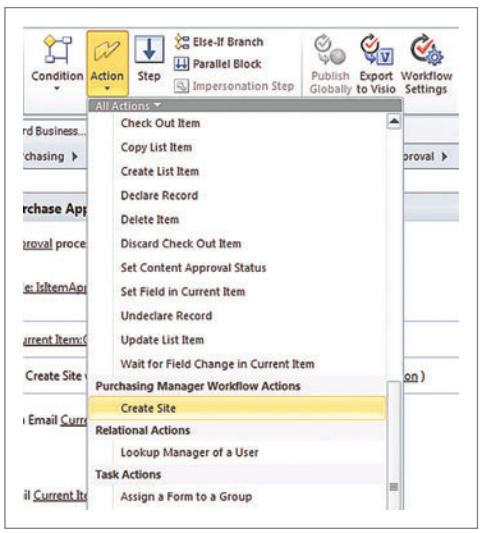


Figure 14 **Create Site Custom Action in Workflow Designer**

Before I test my custom action, I'll open Feature1 and give it a title of Purchasing Manager Workflow Actions to make it more descriptive, and change its scope to Site as required by custom workflow actions.

Deploying the Workflow Solution to SharePoint Online

To test my custom workflow action I'll right-click on the PurchasingMgrActions project in Solution Explorer and select Package to package the solution. I'll then upload the PurchasingMgrActions.wsp to the Solution Gallery in my local development site collection (http://o365dpe.contoso.com/sites/spomsdnmag) to deploy the custom workflow action.

Now, when I open SharePoint Designer 2010 and my Non-Standard Business Purchase Request Workflow Approval workflow in the workflow editor, my custom workflow action is displayed in the Action drop-down under the Purchasing Manager Workflows category, as shown in **Figure 14**.

After setting the Site Name variable to Current Item:Title, my workflow is complete and ready for testing (see **Figure 15**).

Our Name Says It All

activePDF®

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

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

activePDF®
Leading the iPaper Revolution

Compatible with Windows 7 | Windows Vista® | Microsoft Partner Silver Midmarket Solution Provider

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

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

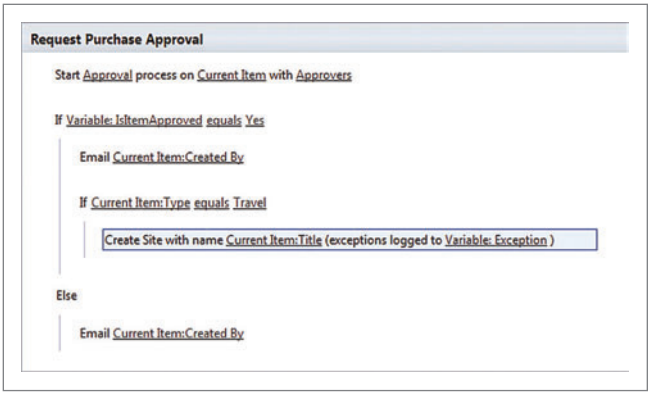


Figure 15 **The Completed Workflow**

To get ready for testing, I need to publish the workflow to my local development site and associate it to my list. To publish, I select Publish in the Save section of the ribbon. When the publication is complete, I'll navigate to my Non-Standard Business Purchase Requests list on my local development site, click the List tab and select Workflow Settings in the Settings section of the ribbon. Under "These workflows are configured to run on items of this type," I'll select Purchasing Manager – Non-Standard Business Purchase Requests and choose the "Add a workflow" link. I'll select my Non-Standard Business Purchase Request Approval workflow as the template and give the workflow the name Non-Standard Business Purchase Approval and click Next and then click Save.

At this point I can run my workflow. I'll select the first item on my list, then choose Workflows in the Workflows section of the ribbon and click on the Non-Standard Business Purchase Approval workflow to start the workflow. I'll then be prompted to enter a business justification as specified in my Business Rationale Initiation Form Parameter.

I'll supply a justification like the one shown and click Start to start the workflow. I can then go into the Workflows page for the item and select the in-progress workflow to see the Workflow Information page, which includes a visualization of the workflow. When the task is approved, the diagram is updated, as seen in **Figure 16**.

I can then go to Site Actions | View All Site Content and see the site that was created by my custom workflow action.

At this point I'm ready to distribute my workflow solution to SharePoint Online. To get the package file for my Non-Standard Business Purchase Request workflow, I'll open the workflow in SharePoint Designer 2010 and select the Save As Template option from the Manage section of the ribbon to save the workflow to the Site Assets library. I can then take the package files Purchasing-Mgr.wsp (or modify the existing list if already deployed), Purchasing-MgrActions.wsp and Non-Standard Business Purchase Request Approval.wsp from the Site Assets library and upload them into the Solution Gallery in my SharePoint Online site collection. Note that the features must be activated in this order based on the dependencies between the features (this can be automated via feature activation dependencies). After activating the workflow feature in my site and associating the workflow to my list, I can follow the same steps to test my workflow in SharePoint Online.

Wrapping Up

Automating business processes via workflows in SharePoint Online can make collaboration even more effective beyond merely storing information in SharePoint Online. In this article I showed how to implement a workflow in SharePoint Designer 2010, extend that workflow with a custom workflow action developed with Visual Studio, and then distribute that workflow via package files created by SharePoint Designer 2010 and Visual Studio 2010. I encourage you to dive deep into workflow development in SharePoint Online. Your users will be impressed at how their manual, error-prone business processes become more efficient with less effort when automated in SharePoint Online workflows. ■

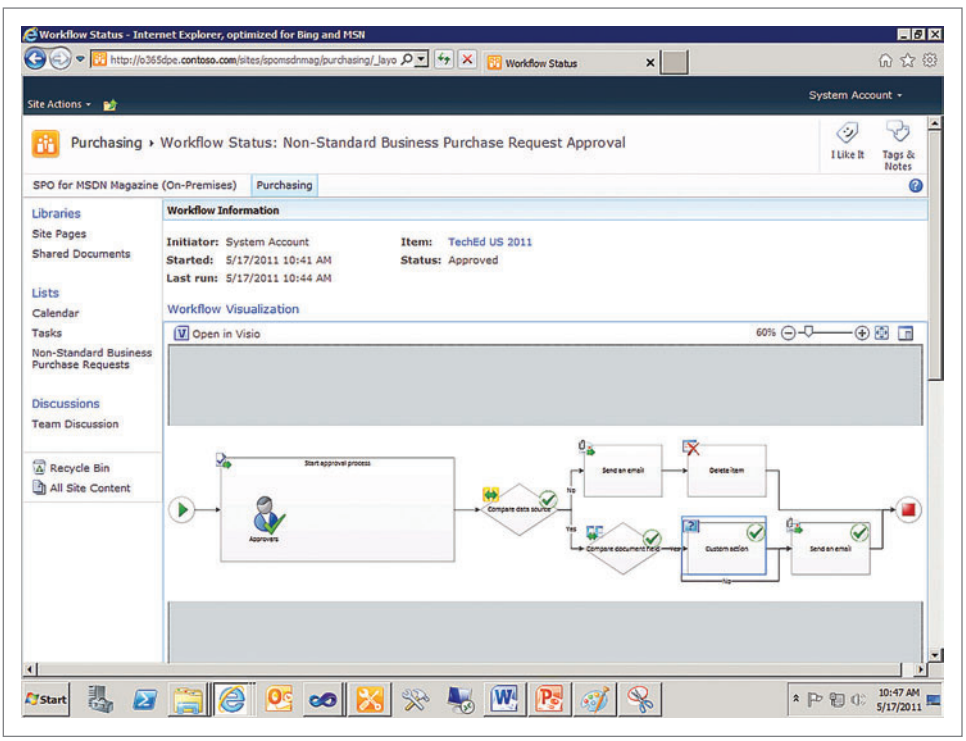


Figure 16 **Workflow Information Page with Visualization**

CHRIS MAYO *is a technology specialist focusing on SharePoint Online and has been with Microsoft for 10 years. He coauthored "Programming for Unified Communications with Microsoft Office Communications Server 2007 R2" (Microsoft Press, 2009). 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/b/cmayo.*

THANKS to the following technical experts for reviewing this article: *Mark Kashman, Keenan Newton, AJ May and George Durzi*

Visual Studio 2010 SP1 for Web Developers

Scott Hanselman and Deepak Verma

Visual Studio 2010 SP1 was released in March. This service pack addresses issues found by customers in partner feedback and internal testing. At the same time Visual Studio 2010 SP1 was released, a number of new components for Web developers came out. This article explores the Web development improvements in Visual Studio 2010 SP1. Most of the improvements were done to integrate Visual Studio SP1 with the new offerings on the Microsoft Web Platform, such as IIS Express 7.5, SQL Server Compact Edition 4.0 (SQL Server CE hereafter for brevity), the new Razor syntax and the Web Platform Installer (Web PI). Other improvements include support for bin-deployable dependencies such as ASP.NET MVC 3, as well as HTML5 and CSS3 support within Visual Studio 2010 SP1 itself.

This article discusses:

- New features available in Visual Studio 2010 SP1
- Integration with the Web Platform Installer
- IIS Express 7.5 support
- Sharing projects with Visual Studio 2010 RTM
- SQL Server Compact Edition support
- Razor support
- Deployment
- HTML5 and CSS3 support

Technologies discussed:

Visual Studio 2010 SP 1, ASP.NET MVC 3, IIS Express 7.5, SQL Server Compact Edition 4.0, Razor syntax, Web Platform Installer

This article assumes that you have Visual Studio 2010 SP1 installed. It can be downloaded as a standalone installer or using the Web PI. Details on where to get it can be found at bit.ly/hnU6mm.

The Web PI is a free tool that makes it easier to get the latest components of the Microsoft Web Platform. More details can be found at bit.ly/a6dLu.

The Web PI is a free tool that makes it easier to get the latest components of the Microsoft Web Platform.

New Features Light Up in Visual Studio 2010 SP1

IIS Express 7.5, SQL Server CE, Visual Studio Tools for SQL Server CE and ASP.NET MVC 3 with Razor are separate components installed after Visual Studio 2010 SP1. In fact, support for these features within Visual Studio 2010 SP1 lies in a dormant state until you've installed them on your machine.

There are various ways to download and install the components. There are a few hard ways, and the easy way. You *could* look for and down-


• REC
RUSS CAM
Become a Star!
Stop by and YOU could be in the next episode!
GrapeCity PowerTools
RUSS CAM
Log in, Find out!
GCPowerTools.com/Videos

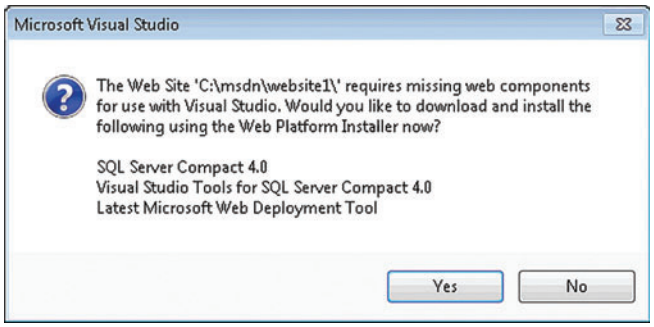


Figure 1 **A Warning About a Missing Component and an Offer to Fetch It**

load each standalone component on your own. Perhaps this would be appropriate for an enterprise or controlled deployment of Visual Studio 2010 SP1. You can use the Web PI to find, download and install each package separately, which is useful if you only want certain features. We'll discuss individual installations later. However, the easiest way is to use a "bundle" of all these features within the Web PI. We've taken the liberty of making this short URL for you that will launch the Web PI and automatically select the right options for you to get *everything*: hnsl.mn/VS2010SP1Bundle.

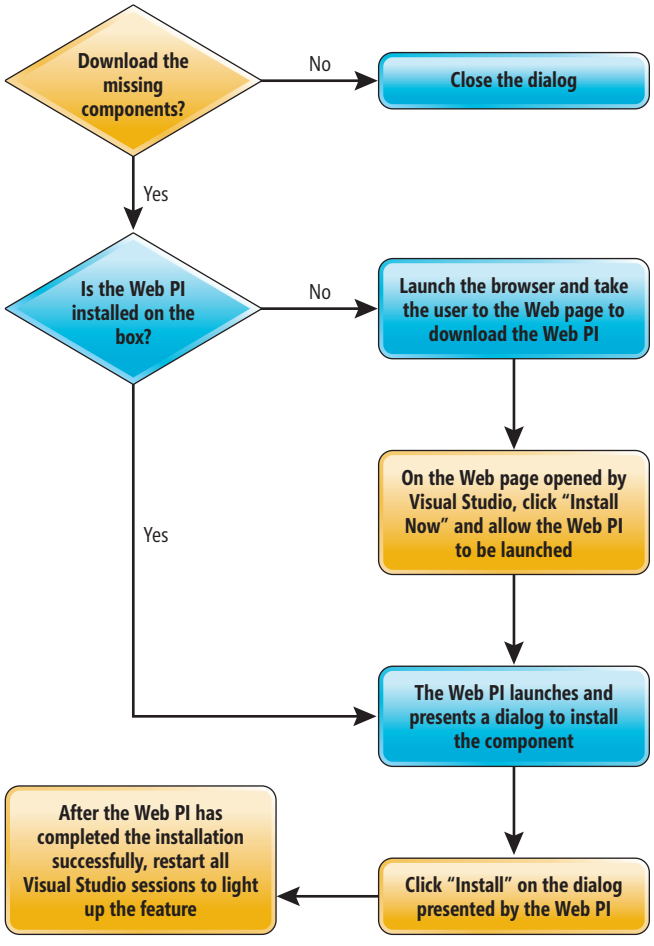


Figure 2 **Steps Taken After You're Presented with the Missing Component Dialog Box**

The Web PI is smart about what you have on your machine. If you have literally nothing—no development tools—on your machine, this link will automatically install the free Visual Studio Web Developer Express SP1 and all the new features. However, if you have any version of Visual Studio already installed, or even if you have Visual Studio 2010 SP1 and *some* of the tools, the Web PI will automatically get just the minimum you need to be up-to-date. It's the "get everything" link.

Make sure that you've shut down all your instances of Visual Studio (remember to check Task Manager for devenv.exe) before you start installation.

> The Web PI is smart about what you have on your machine.

You might find yourself opening a project that uses new components you don't have. Visual Studio 2010 SP1 will detect the missing components automatically if you're opening an existing Web project that uses some of these new features, and use the Web PI to install them.

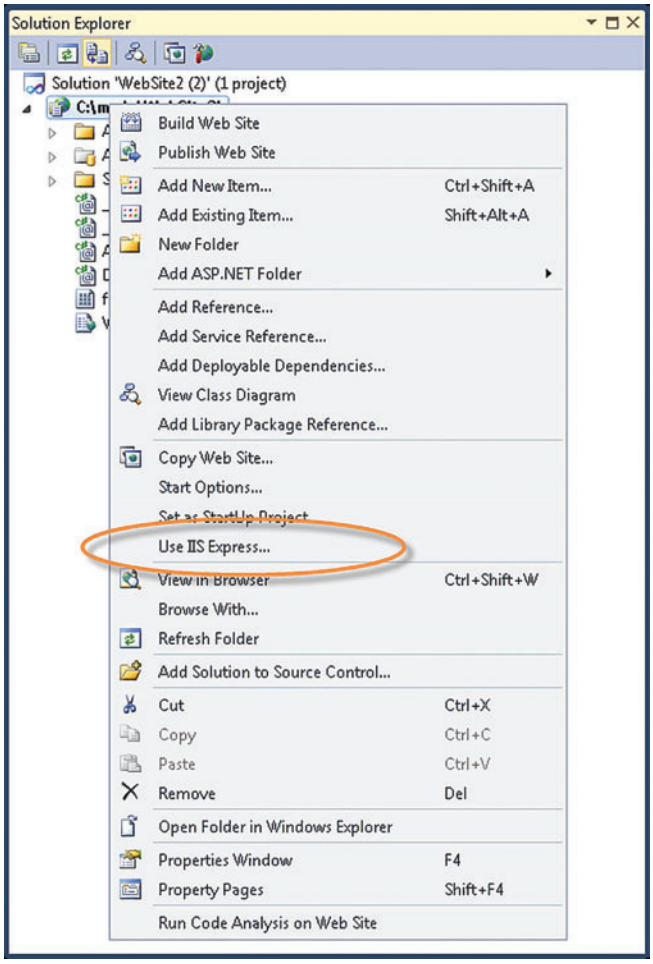


Figure 3 **Converting from Visual Web Development Server to IIS Express 7.5**

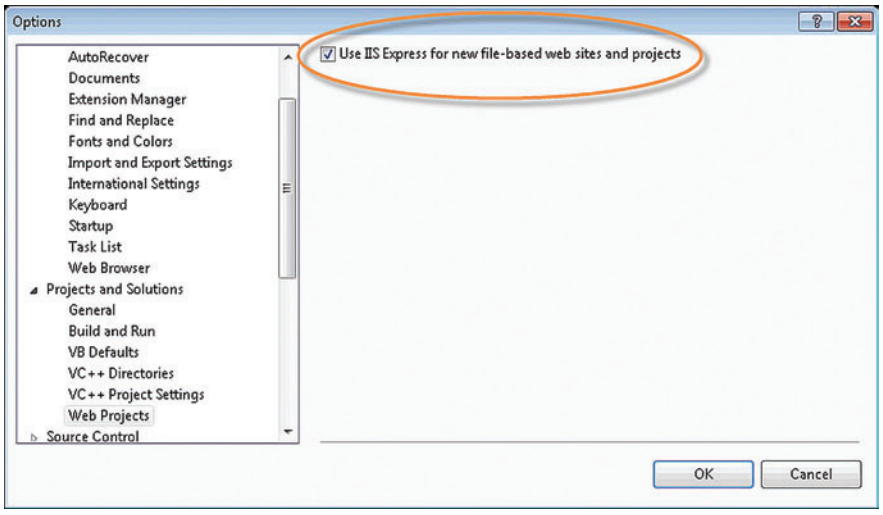


Figure 4 **Making IIS Express 7.5 Your Default Web Server**

Web PI Integration

In this section, we'll explain how Visual Studio 2010 SP1 can automatically detect missing Web components that are required by a Web project, and also introduce the Web PI toolbar.

Missing component dialog This is a new dialog introduced in the service pack. This dialog comes up when you open an existing Web project that uses IIS 7.5 Express, SQL Server CE or ASP.NET MVC Razor syntax—and if any of these components aren't installed on your machine. It offers to download and install the missing components used by the Web project using the Web PI.

For example, **Figure 1** shows what you see on a box without SQL Server CE while opening a Web site that uses a SQL Server CE database.

At this point, if you click Yes, Visual Studio 2010 SP1 will try to detect if you have the Web PI installed on your machine. If not, it will open the download page in a browser window to at least get you the Web PI.

If you already have the Web PI installed, it will launch the Web PI with the missing component installation selected. It'll just download what you need. Don't worry, it's smart. It won't download Visual Studio 2010 SP1, because you already have that!

Figure 2 explains what happens after you click Yes or No on the missing component dialog (the orange figures are where your inputs are required; blue figures are steps performed by Visual Studio 2010 SP1).

The Web PI toolbar If you want to download any other components using the Web PI, a new toolbar with an "Install Web Components" button is also added to Visual Studio.

When you click on the button, it will launch the Web PI, or if you don't have the Web PI, it will take you to the download page. You can select View | Toolbars | Web Platform Installer to show the toolbar if it's not already up. The "Install Web Components" option is also available under the Tools menu.

IIS Express 7.5 Support

Now that you know how to get the various components installed, let's explore the IIS Express 7.5 feature in Visual Studio.

IIS Express 7.5 is a lightweight version of IIS optimized for developers. More details can be found at bit.ly/c0frt4.

IIS Express 7.5 is built from the same code base as IIS 7.5, so it provides Web developers with features of IIS—such as SSL, URL rewrite and MIME types—on the development box, which is similar to the full-product Web server.

IIS Express 7.5 can be downloaded and installed using the Web PI (bit.ly/dfikKe) or the standalone installer (bit.ly/g5RMgc).

> IIS Express 7.5 is a lightweight version of IIS optimized for developers.

Perhaps you have an existing Web site or project that uses Visual Web Development Server (the tiny Web server formerly known as Cassini). You can right-click the Web project in Solution Explorer and convert it to use IIS Express 7.5 (see **Figure 3**).

Visual Studio 2010 SP1 prompts for a confirmation and then shows that Web project has been successfully set to use IIS Express 7.5.

However, setting projects to use IIS Express 7.5 over and over again is tedious. If you want, you can make IIS Express 7.5 the default Web server for all your projects from the Tools | Options dialog, under Project and Solutions | Web Projects. Check the "Use IIS Express for new file-based web site and projects" option (see **Figure 4**).

Of course, you can pick and choose and even convert back from IIS Express 7.5 to the Visual Studio Web Development Server on a project-by-project basis. Just right-click a project in Solution Explorer and converting it to use the Visual Studio Web Development Server.

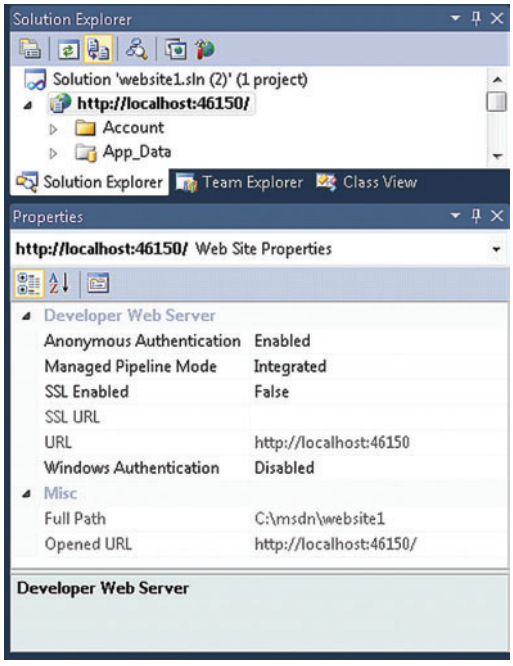


Figure 5 **The Project Properties Window**

IIS Express 7.5 and Non-Administrators

To create or open an existing IIS Web site or Web project, you need to run Visual Studio 2010 SP1 as an elevated user (administrative mode). This is a hassle for many folks who don't have that ability. Perhaps it's restricted by IT, or perhaps you just like to avoid running as an administrator when you don't have a good reason to do so.

Fortunately, you can use IIS Express 7.5 as the Web server for Web projects without the need to run Visual Studio 2010 SP1 in the context of an administrator account. Once it's installed, any user can work with Visual Studio 2010 SP1 and IIS Express 7.5 as a non-admin.

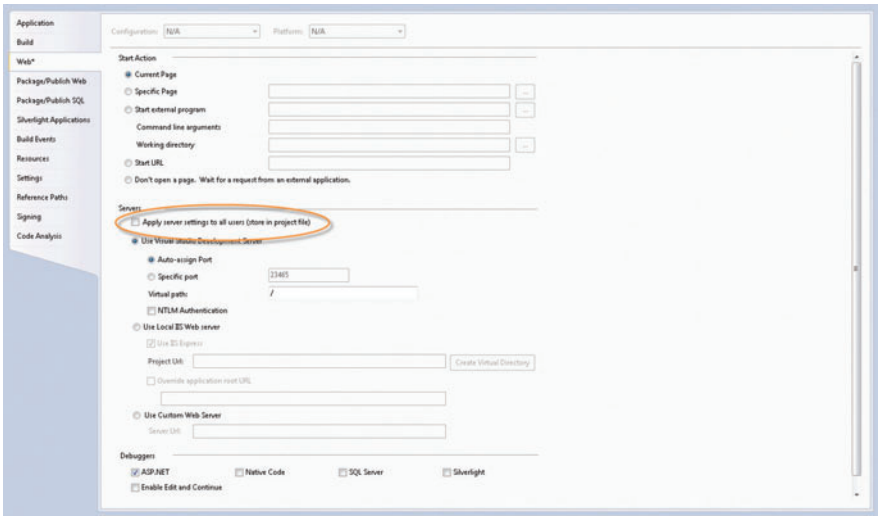


Figure 6 **Make Server Selection and Related Settings on a Per-User Basis**

IIS Express 7.5 Properties

IIS Express 7.5 has a number of custom properties for each project that can be set from the properties window (select the project in Solution Explorer and press F4 to bring up the properties window). **Figure 5** shows the properties window.

Following are the properties specific to IIS Express 7.5:

1. **Anonymous Authentication**—To enable/disable anonymous authentication. Anonymous authentication allows any user to access the site without providing a user name and password challenge to the client browser. By default it's enabled.
2. **Managed Pipeline Mode** (can be integrated or classic): Integrated—The ApplicationPool object will use the integrated request-processing pipelines of IIS 7 and ASP.NET to process requests for managed code.
   Classic—IIS 7 will route requests for managed code through the aspnet_isapi.dll, which processes requests the same as if the application were running in IIS 6.
3. **SSL Enabled**—Set it to true to enable SSL for the site.
4. **SSL URL**—The SSL URL.
5. **URL**—The URL of the Web site.
6. **Windows Authentication**—This can be enabled or disabled; use Windows authentication to authenticate using the Windows NT LAN Manager (NTLM) or Kerberos protocols.

Settings that you change in the properties toolbox are stored in "My Documents\IISExpress\config\applicationhost.config." Note that these settings are not in the "C:\Program Files" folder. Each user has its own IIS Express 7.5 configuration. Because these are strictly set on the local IIS Express 7.5 instance and the config file isn't included in the solution file, it isn't persisted in the source control, so be aware.

Sharing IIS Express 7.5 Projects with Visual Studio 2010 RTM

Say you have a project using IIS Express 7.5 on localhost:20221 and you want to share it with your teammate who still has Visual Studio 2010 RTM. As soon as she opens the Web project on her box, it would throw an error similar to "Could not find the server 'http://localhost:20221' on the local machine."

The reason behind this error is that Visual Studio 2010 RTM doesn't know about IIS Express and is trying to find locahost:20221.

You can work around this in a Web Application Project by un-checking "Apply server settings to all users (store in project file)" in the Property pages | Web tab (see **Figure 6**).

Right-click the project in Solution Explorer and select Properties from the context menu to bring up the Property pages.

This causes the server selection (Visual Web Development Server or IIS Express 7.5) and related settings to be stored on a per-user basis

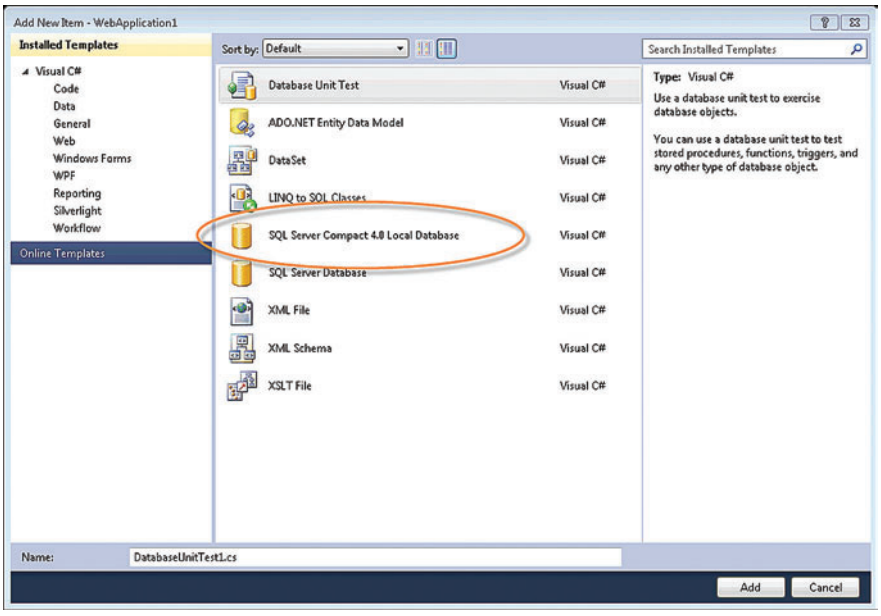


Figure 7 **Creating a SQL Server CE Database**

in the user file rather than in the project file, thus causing the server selection to not persist into source control.

This will allow the Visual Studio 2010 RTM users on the team to still use Visual Studio Development Server while the Visual Studio 2010 SP1 user can use IIS Express 7.5 for the same project. For Web site projects, everyone on the team needs to be using Visual Studio 2010 SP1 to be able to share projects and use IIS Express 7.5.

SQL Server CE Support

Microsoft SQL Server CE is a free, embedded database that can be used for building ASP.NET Web sites and desktop applications. It's a lightweight database that doesn't need any installations. On a production server, a Web project can use a SQL Server CE database by just dropping SQL Server CE database engine assemblies in the bin folder. (See the "Deployable Dependency" section of this article to see how Visual Studio can automatically help add the required assemblies to the bin folder.)

For more details on SQL Server CE, see bit.ly/dsWBbM and bit.ly/hvgQQV. To get SQL Server CE support in Visual Studio 2010 SP1, components can be downloaded using the Web PI (bit.ly/maOfQX) or via a standalone installer. For the latter, install the following two components: the SQL Server CE runtime (bit.ly/f86AyF) and the Visual Studio for SQL Server tools (bit.ly/kvxEQi).

In this section we'll explore SQL Server CE features that light up after you install the required components.

With Visual Studio 2010 SP1, you can now create a SQL Server CE database using the new templates (see **Figure 7**). (To bring up the "Add a new item" dialog, right-click on the project in the Solution Explorer and select "Add new item.") While adding the .sdf file into a Web application project, Visual Studio 2010 SP1 automatically adds a reference to System.Data.SqlServerCE, whereas for a Web site, it updates the web.config to add the assembly.

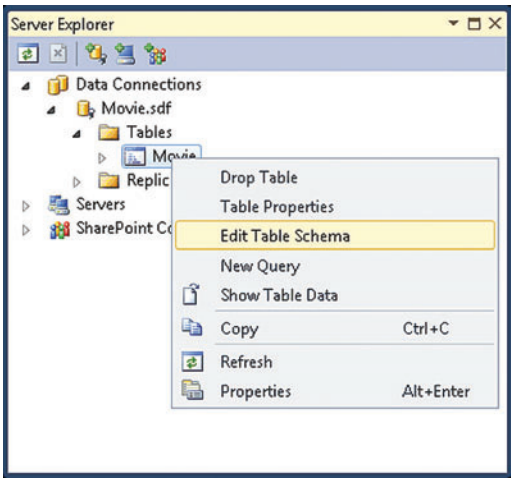


Figure 8 **Right-Clicking on an .sdf File to Perform Various Tasks**

To perform various tasks with .sdf files, double-click the .sdf file in Solution Explorer, which will open up the Server Explorer pane (see **Figure 8**).

The Server Explorer helps perform various tasks such as creating a new table, editing an existing one, editing the table schema, showing table data and more.

Other data features in Visual Studio 2010 SP1 such as keys/index management, Entity Framework Designer and Server/Database Explorer have also been updated to support SQL Server CE databases.

Razor Support

In Visual Studio 2010 SP1, changes are made to the Web project systems and to the HTML editor to support the new Razor syntax. This new syntax is used by ASP.NET Web Pages and by ASP.NET MVC 3.

Microsoft SQL Server CE is a free, embedded database that can be used for building ASP.NET Web sites and desktop applications.

Read more about the ASP.NET MVC Razor syntax at bit.ly/ajOAuM.

To get Razor syntax support in Visual Studio 2010 SP1, you need to download ASP.NET MVC 3.

The Web PI link to download ASP.NET MVC 3 is bit.ly/biXkTD. The ASP.NET MVC 3 standalone installer is at bit.ly/hd2LDs.

Here's a post that explains what gets installed with ASP.NET MVC 3: bit.ly/e7T4A3.

The following support has been added for Razor syntax:

Colorization With Visual Studio 2010 SP1 and ASP.NET MVC 3, when you open a Razor file (.cshtml or .vbhtml), the Visual Studio Editor colorizes the HTML and the C# or Visual Basic content.

You can also change the default background color for Razor code blocks on the Tools | Options dialog, by selecting Environment | Fonts and Colors (see **Figure 9**).

IntelliSense Full HTML and C#/Visual Basic language IntelliSense support is provided in a Razor file. See **Figure 10** for an example.

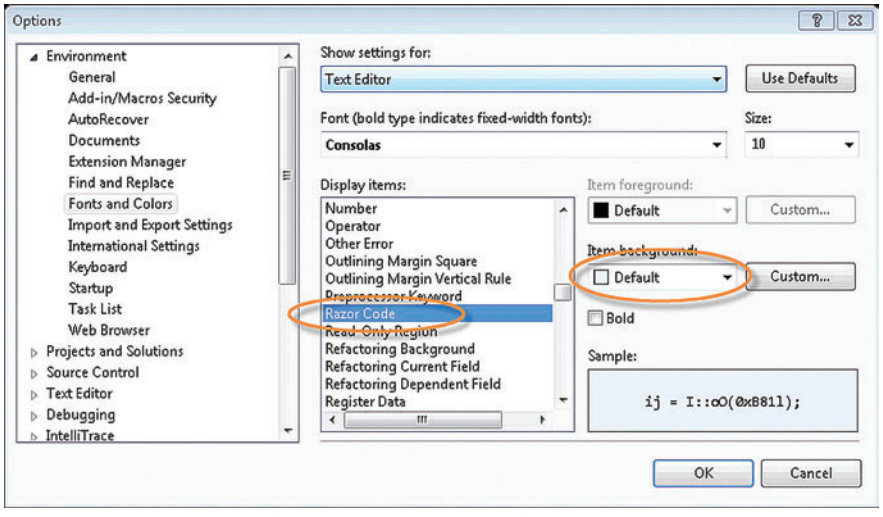


Figure 9 **Changing the Default Background Color for Razor Code Blocks**

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

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

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

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

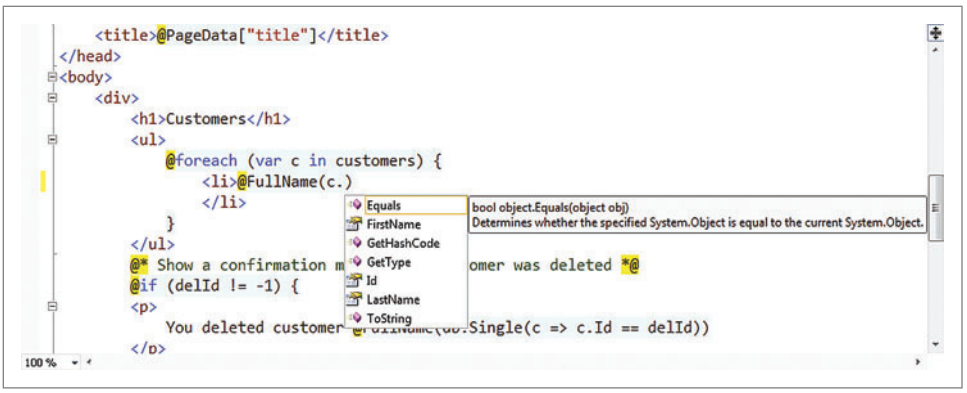


Figure 10 **IntelliSense Support for Razor Code**

Debugging As with support for other language files, you can add breakpoints in a Razor file and use all the debugging features of the IDE (such as step over, watch list and more).

For an inline Razor expression, breakpoints can't be added by clicking on the margin. The trick is to move the cursor to an expression and then press F9 (shortcut to add a breakpoint).

Deployable Dependencies

In previous sections, we talked about how to download, install and explore features for SQL Server CE, Razor and ASP.NET MVC 3. Now we want to publish our Web project that's using these features to a remote Web server. We're not sure if the remote server has SQL Server CE, Razor or ASP.NET MVC 3 installed and we don't have control over what can be installed on it. Visual Studio 2010 SP1 can actually help us "bin deploy" binaries for these components using the Add Deployable Dependencies feature (see **Figure 11**).

The bin folder is a special folder in Web projects that's recognized by ASP.NET. A bin folder can contain compiled assemblies for custom ASP.NET controls, components or other code referenced in the Web application.

Our Web application might have dependency on a component (for example, SQL Server CE), and we want to publish our Web site on a server that doesn't have this dependency installed (it doesn't have SQL Server CE installed).

To "bin deploy" means to deploy an application with the dependencies copied into the application's bin folder rather than installing the dependencies into the Global Assembly Cache (GAC) using an installer.

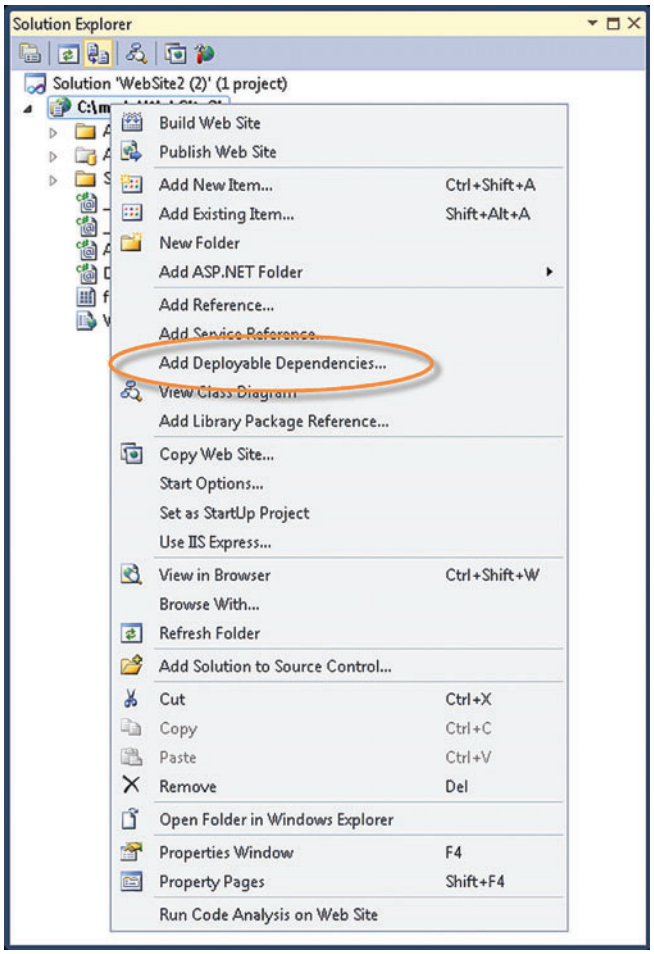


Figure 11 **The Add Deployable Dependencies Feature**

> Full HTML and C#/Visual Basic language IntelliSense support is provided in a Razor file.

To bring up the dialog, right-click on the Web project in the Solution Explorer and select "Add Deployable Dependencies …"

Visual Studio 2010 SP1 will show you a dialog with a list of components that are installed on your machine and can be deployed. It shows ASP.NET MVC as an additional option only for an ASP.NET MVC 3 project (see **Figure 12**).

Here's what happens after you click OK. In the case of a Web site, the required assemblies for the components selected would be copied from the dependency install location on your box to an application's bin folder. Now when you copy or publish your

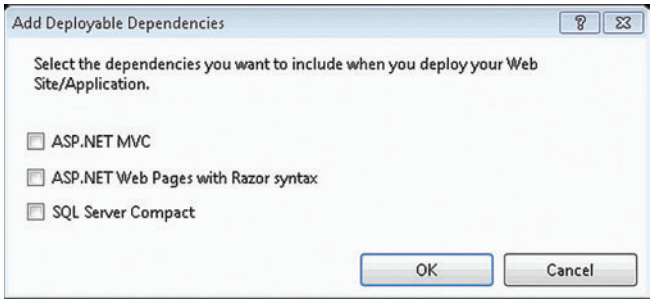


Figure 12 **The Add Deployable Dependencies Dialog Box for a Bin Deploy**

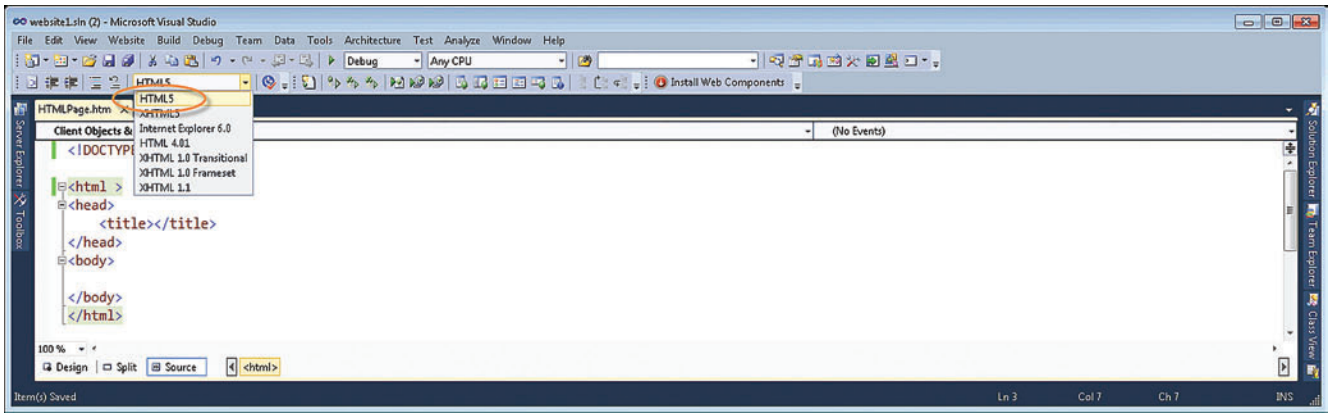


Figure 13 **Selecting HTML5 from the HTML Source Editing Toolbar**

Web site to the Web server, the component works using assemblies from the bin folder.

In a Web application project, assemblies for the components selected would be copied to the _bin_deployableassemblies folder. When the Web application project is built or when we package it for deployment, binaries are copied from _bin_deployableassemblies folder to the bin folder.

This is done so that even if you delete files from the bin during your custom build process, it would still copy the dependency again correctly from the _bin_deployableassemblies folder when the project is published.

HTML5 and CSS3 Support

Preliminary support is added for HTML5 to the HTML editor so that IntelliSense and validation for HTML elements and attributes can be obtained. A complete version that brings a fuller HTML5 experience will be provided in future versions of Visual Studio.

To get HTML5 support, go to Tools | Options, select Text Editor | HTML | Validation, then select HTML5 from the dropdown. Or you can also select HTML5 from the HTML source editing toolbar (see **Figure 13**). To enable the toolbar while editing an HTML or Razor file, select View | Toolbar | HTML Source Editing.

In Visual Studio 2010 SP1, there are a few improvements in CSS3 support, though these improvements aren't as elaborate as those of HTML5. The CSS editor now supports the more advanced selectors such as *div:nth-child(2n+1)* and the new color values *rgba*, *hsl* and *hsla*, and eight-digit hex values.

Wrapping Up

To recap, Visual Studio 2010 SP1 tightens up a few things in Visual Studio, but it's the new Web components working together with things such as IIS Express 7.5, ASP.NET MVC 3 with Razor, SQL Server CE with tooling and HTML5/CSS3—tied together with the Web PI—that really make Web development more enjoyable. We hope you enjoy these tools as much as we do. ■

Scott Hanselman *works as a principal community architect for Microsoft outside of Redmond, Wash. You can follow him on Twitter at twitter.com/shanselman, read his blog at hanselman.com or listen to his podcasts at hanselminutes.com*

Deepak Verma *works as a software development engineer in test for Microsoft in Redmond, Wash. His team blog is at blogs.msdn.com/webdevtools.*

Tips and Tricks for Loading Silverlight Locale Resources, Part 2

Matthew Delisle and John Brodeur

In the first article in this series (msdn.microsoft.com/magazine/gg650657), I covered the loading of resources in Silverlight using a Windows Communication Foundation (WCF) service, a simple database schema and client-side code that notified the UI of a change in resources. In that solution, the default resources were loaded through a Web service call during initialization of the application. In this article, John Brodeur and I will show you how to load the default resources without making any Web service calls.

The Standard Localization Process

In the standard localization process, as outlined in the previous article, there are a few ways to retrieve locale resources. A common method is to embed .resx files in the application at design time, as shown in **Figure 1**.

The downside of any method that embeds resources in the application is that all of the resources are then downloaded with the application. All of the native localization methods available in Silverlight embed the resources into the application in some way.

A better solution is to embed only a default resource set and load any other resources on demand. Loading resources on demand can be accomplished in a variety of ways: through the retrieval of .xap or .resx files, or, as outlined in Part 1 of this series, through the use of Web services to retrieve .resx XML strings. The issue that remains, however, is that the default locale may not be the user's primary locale. Users whose locale differs from the default will always have to retrieve resources from the server.

The best solution is to generate a .xap file on demand with the current user's locale-specific resources embedded in the .xap. With this solution, no Web service calls are needed to load the default resources for any user. A Web service call is needed only when changing the locale at runtime. This is the solution we'll discuss in the rest of this article.

This article discusses:

- Loading localization resources on demand without making Web service calls
- Creating a custom localization solution
- Using a custom resource provider

Technologies discussed:

Silverlight

Code download available at:

code.msdn.microsoft.com/mag201107Silverlight

A Custom Localization Process

The custom localization solution in this article consists of both client and server components and builds on the CustomLocalization project created in Part 1. We'll describe the process beginning with the .aspx file containing the Silverlight object.

Any parameters for the HttpHandler need to be passed in through the URL of the Silverlight application. In order to pass in the browser culture, we add a URL parameter and fill it with the current thread's culture:

```
<param name="source" value="ClientBin/CustomLocalization.xap?c=
  <%= Thread.CurrentThread.CurrentCulture.Name %>&rs=ui"/>
```

We also need to add an import for the System.Threading namespace in order to use the Thread class:

```
<%@ Import Namespace="System.Threading" %>
```

And we added a parameter called rs that represents the resource set to retrieve.

That's all that's needed in the .aspx file. The user's locale is passed into the HttpHandler, which will embed the resources specified by that culture into a .xap file.

Creating the Handler

Now we're going to create a file called XapHandler in the root of the Web project. This class will implement IHttpHandler and we'll specify that it's non-reusable. We'll add three fields to share the CultureInfo, HttpContext and ResourceSet objects among methods. The code so far looks like this:

```
using System.Web;
namespace CustomLocalization.Web {
  public class XapHandler : IHttpHandler {
    private CultureInfo Culture;
    private HttpContext Context;
    private string ResourceSet;

    public bool IsReusable { get { return false; } }

    public void ProcessRequest(HttpContext context) {
      throw new System.NotImplementedException();
    } } }
```

In the ProcessRequest method, we want to retrieve the culture and resource set, validate the culture, and then create a localized .xap file and transmit the file to the client. To retrieve the parameters, we'll access them from the Params array of the Request object:

```
string culture = context.Request.Params["c"];]
ResourceSet = context.Request.Params["rs"];
```

The user's locale is passed into the HttpHandler, which will embed the resources specified by that culture into a .xap file.

To validate the culture, we'll try to create a CultureInfo object; if the constructor fails, the culture is assumed to be invalid:

```
if (!string.IsNullOrEmpty(culture)) {
  try {
    Culture = new CultureInfo(culture);
  }
  catch (Exception ex) {
    // Throw an error
  } }
```

This is a good place to create a Utilities class to hold some commonly used functions for reuse. We'll start with a function that sends a response to the client and then closes the response object. This is useful for sending error messages. Here's the code:

```
public static void SendResponse(HttpContext context, int statusCode,
  string message) {
  if (context == null) return;
  context.Response.StatusCode = statusCode;
  if(!string.IsNullOrEmpty(message)) {
    context.Response.StatusDescription = message;
  }
  context.Response.End();
}
```

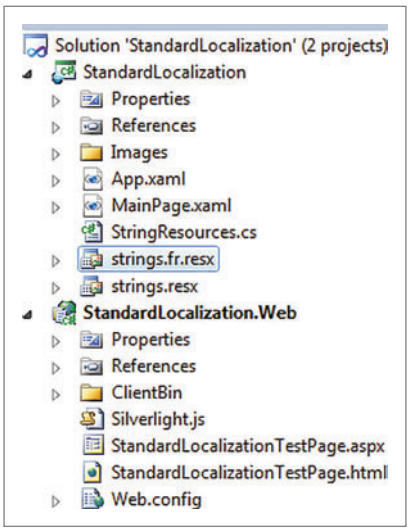


Figure 1 **Resource Files Embedded in the Application**

And we'll use that method to send an error when an invalid culture is specified:

```
if (!string.IsNullOrEmpty(culture)) {
  try {
    Culture = new CultureInfo(culture);
  }
  catch (Exception ex) {
    // Throw an error
    Utilities.SendResponse(Context, 500,
    "The string " + culture + " is not recognized
as a valid culture.");
     return;
  } }
```

After validating the culture, the next step is to create the localized .xap file and return the file path.

Creating the Localized XAP File

This is where all the magic happens. We're going to create a method called CreateLocalizedXapFile with a parameter of type string. The parameter specifies the location on the server of the application .xap file that contains no embedded resources. If the .xap file without resources doesn't exist on the server, the process can't continue, so we throw an error, like so:

```
string xapWithoutResources = Context.Server.MapPath(Context.Request.Path);
if (string.IsNullOrEmpty(xapWithoutResources) || !File.Exists(xapWithoutResources))
  Utilities.SendResponse(Context, 500, "The XAP file does not exist.");
  return;
}
else {
  string localizedXapFilePath = CreateLocalizedXapFile(xapWithoutResources);
}
```

Before diving into the CreateLocalizedXapFile method, let's look at the directory structure of this solution on the Web server. Let's say we have a Web application called acme in the root Web folder. Inside of the acme folder will be the ClientBin directory, where Silverlight applications are normally stored. This is where .xap files without resources are located. Under this directory are other directories named after locale identifiers (en-US, es-MX, fr-FR and so forth), and these directories are where the locale-specific .xap files are created and stored. **Figure 2** shows what the directory structure could look like.

Now let's dive into the CreateLocalizedXapFile method. There are two main paths of execution in this method. The first is if the localized .xap file exists and is up-to-date. In this case, the process is trivial and all that happens is that the full path to the localized .xap file is returned. The second path is when the localized .xap file does not exist or is out of date. The localized .xap file is considered out of date if it's older than the plain .xap file or the .resx file that should be embedded in it. The individual .resx files are stored outside of the localized .xap file so that they can be easily modified, and these files are used to check whether the localized .xap file is current. If the localized .xap file is obsolete, it's overwritten with the plain .xap file and the resources are injected into that file. **Figure 3** shows the commented method.

The GetResourceFilePath method is shown in **Figure 4**. The parameters to this method are the context, resource set and culture. We create a string representing the resource file, check to see if it exists and, if it does, return the file path.

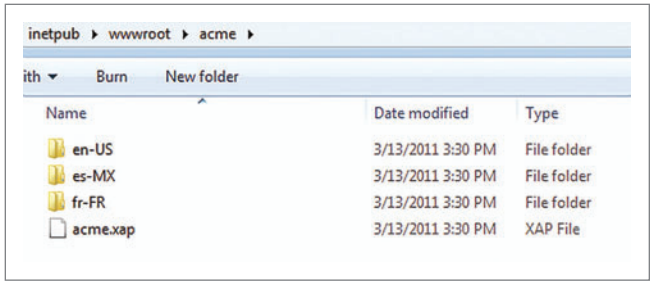


Figure 2 **Directory Structure for Localized XAP Files**

Injecting Resources into a XAP file

Now let's move on to the InjectResourceIntoXAP method. As most Silverlight developers know, a .xap file is a .zip file in disguise. Creating a .xap file is as easy as zipping the correct files together and assigning the result a .xap extension. In this scenario, we need to take an existing .zip file—the .xap file without resources—and add the .resx file of the appropriate culture to it. To assist in the zipping functionality, we'll use the DotNetZip library, located at dotnetzip.codeplex.com. We first attempted to use the System.IO.ZipPackage to do the compression without the external library, but we ran into compatibility issues with the resulting .xap file. The process should be possible using just the System.IO.ZipPackage namespace, but the DotNetZip library made it much easier.

Here's a utility method we created to help with the zip functionality:

```
public static void AddFileToZip(string zipFile, string fileToAdd,
  string directoryPathInZip) {
  if (string.IsNullOrEmpty(zipFile) || string.IsNullOrEmpty(fileToAdd)) return;

  using (ZipFile zip = ZipFile.Read(zipFile)) {
    zip.AddFile(fileToAdd, directoryPathInZip);
    zip.Save();
  } }
```

> Creating a .xap file is as easy as zipping the correct files together and giving the result a .xap extension.

In the InjectResourceIntoXAP method, we're just wrapping a call to the AddFileToZip method with some error handling:

```
private bool InjectResourceIntoXAP(FileInfo localizedXapFile,
  FileInfo localizedResxFile) {
  if (localizedXapFile.Exists && localizedResxFile.Exists) {
    try {
      Utilities.AddFileToZip(localizedXapFile.FullName,
        localizedResxFile.FullName, string.Empty);
      return true;
    }
    catch { return false; }
  }
  return false;
}
```

What we originally thought was going to be one of the most complicated parts of the solution turned out to be the simplest. Think of all the other uses for dynamically created .xap files!

Transmitting the File to the Client

We're going to swim back up to the surface now and finish the ProcessRequest method. When we were last here, we added the code to call the CreateLocalizedXapFile method and returned the path to the .xap file, but we haven't done anything with that file. In order to assist in transmitting files to the client, I'm going to create another utility method. The method, called TransmitFile, sets the headers, content type and cache expiration of the file and then uses the TransmitFile method of the HttpResponse class to send the file directly to the client, without buffering. **Figure 5** shows the code.

In the ProcessRequest method, we call the TransmitFile method, giving it the context and the localized .xap file path and specifying not to delete the file (cache it) after the transmission completes:

```
Utilities.TransmitFile(context, localizedXapFilePath, "application/x-
silverlight-app", false);
```

> As most Silverlight developers know, a .xap file is a .zip file in disguise.

Making It Work

At this point, we have a working .xap handler, and now we need to wire it up in the Web application. We're going to add the handler in the httpHandlers section of the web.config. The path of the handler

Figure 3 **The CreateLocalizedXapFile Method**

```
private string CreateLocalizedXapFile(string filePath) {
  FileInfo plainXap = new FileInfo(filePath);
  string localizedXapFilePath = plainXap.FullName;

  try {
    // Get the localized XAP file
    FileInfo localizedXap = new FileInfo(plainXap.DirectoryName +
      "\\" + Culture.Name + "\\" + plainXap.Name);

    // Get the RESX file for the locale
    FileInfo resxFile = new FileInfo(GetResourceFilePath(
      Context, ResourceSet, Culture.Name));

    // Check to see if the file already exists and is up to date
    if (!localizedXap.Exists || (localizedXap.LastWriteTime <
      plainXap.LastWriteTime) ||
      (localizedXap.LastWriteTime < resxFile.LastWriteTime)) {
    if (!Directory.Exists(localizedXap.DirectoryName))  {
      Directory.CreateDirectory(localizedXap.DirectoryName);
     }

     // Copy the XAP without resources
     localizedXap = plainXap.CopyTo(localizedXap.FullName, true);

     // Inject the resources into the plain XAP, turning it into a localized XAP
     if (!InjectResourceIntoXAP(localizedXap, resxFile)) {
       localizedXap.Delete();
  } }
     if (File.Exists(localizedXap.FullName)) {
       localizedXapFilePath = localizedXap.FullName;
     } }
  catch (Exception ex) {
    // If any error occurs, throw back the error message
    if (!File.Exists(localizedXapFilePath)) {
      Utilities.SendResponse(Context, 500, ex.Message);
    } }
  return localizedXapFilePath;
}
```

Figure 4 **The GetResourceFilePath Method**

```
private static string GetResourceFilePath(
  HttpContext context, string resourceSet, string culture) {
  if (context == null) return null;
  if (string.IsNullOrEmpty(culture)) return null;

  string resxFilePath = resourceSet + "." + culture + ".resx";
string folderPath = context.Server.MapPath(ResourceBasePath);
FileInfo resxFile = new FileInfo(folderPath + resxFilePath);

if (!resxFile.Exists) {
  Utilities.SendResponse(context, 500, "The resx file does not exist
    for the locale " + culture);
}
return resxFile.FullName;
}
```

will be the file path of the .xap file with an asterisk inserted before the extension. This will route any request to that .xap file, no matter the parameters, to the handler. The system.web configuration section is used with Cassini and IIS 6 and the system.webServer section is for use with IIS 7:

```
    <system.web>
      <httpHandlers>
        <add verb="GET" path="ClientBin/CustomLocalization*.xap"
          type="CustomLocalization.Web.XapHandler, CustomLocalization.Web"/>
      </httpHandlers>
    </system.web>
  <system.webServer>
      <handlers>
        <add name="XapHandler" verb="GET" path=
          "ClientBin/CustomLocalization*.xap"
          type="CustomLocalization.Web.XapHandler, CustomLocalization.Web"/>
      </handlers>
    </system.webServer>
```

Now, by moving resource files into folders for each locale on the server, the solution is working. Whenever we update the .resx files, the localized .xap files become obsolete and are regenerated on-demand. Thus we've created a solution that lets us deploy a Silverlight application with resources for any language without making a single Web service call. Now let's take it a step further. The source of truth for the locale information is not the .resx files. The source of truth is the database, and the .resx files are byproducts of the database. In an ideal solution, you wouldn't have to deal with .resx files; you'd only modify the database when resources are added or updated. Right now, the .resx files need to be updated when the database changes, and this can be a tedious process, even with a semi-automated tool. The next section takes a look at automating the process.

Figure 5 **The TransmitFile Method**

```
public static void TransmitFile(HttpContext context, string filePath,
  string contentType, bool deleteFile) {
  if (context == null) return;
  if (string.IsNullOrEmpty(filePath)) return;

  FileInfo file = new FileInfo(filePath);
   try {
     if (file.Exists) {
       context.Response.AppendHeader("Content-Length", file.Length.ToString());
       context.Response.ContentType = contentType;
       if (!context.IsDebuggingEnabled) {
         context.Response.Cache.SetCacheability(HttpCacheability.Public);
         context.Response.ExpiresAbsolute = DateTime.UtcNow.AddDays(1);
         context.Response.Cache.SetLastModified(DateTime.UtcNow);
       }

       context.Response.TransmitFile(file.FullName);
     if (context.Response.IsClientConnected) {
       context.Response.Flush();
     } }
     else {
       Utilities.SendResponse(context, 404, "File Not Found (" + filePath + ")."); }
     }
     finally {
       if (deleteFile && file.Exists) { file.Delete(); }
     } }
```

Using a Custom Resource Provider

Creating a custom resource provider is a complex undertaking and not within the scope of this article, but Rick Strahl has a well-written article discussing a similar implementation at bit.ly/ItVajU. For this article, we're using a subset of his Resource Provider solution. The main method, GenerateResXFileNormalizedForCulture, will query our database for the complete resource set of a given culture string. When constructing the resource set for a culture, the standard .NET resource manager hierarchy is maintained for each key by first matching the Invariant culture, then the Neutral (or language) culture and finally the Specific culture resource.

For example, a request for the en-us culture would result in the combination of the following files: ui.resx, ui.en.resx and ui.en-us.resx.

> In an ideal solution, you wouldn't have to deal with .resx files; you'd only modify the database when resources are added or updated.

Using the Embedded Resources

In Part 1, the solution retrieved all resources using Web service calls, and if the Web service was unavailable, it would fall back to a file stored in the Web directory that contained the default resource strings. Neither of these procedures is necessary anymore. We'll delete the file with the default resource strings and remove the application setting pointing to it. The next step is to modify the SmartResourceManager to load the embedded resources when the application initializes. The ChangeCulture method is the key to integrating the embedded resources into the solution. The method looks like this right now:

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

Instead of making a call to the GetResourcesAsync operation right away, we're going to try to load the resources from an embedded resource file—and if that fails, then make the call to the Web service. If the embedded resources load successfully, we'll update the active resource set. Here's the code:

```
if (!ResourceSets.ContainsKey(culture.Name)) {
  if (!LoadEmbeddedResource(culture)) {
    localeClient.GetResourcesAsync(culture.Name, culture);
  }
else {
  ResourceSet = ResourceSets[culture.Name];
  Thread.CurrentThread.CurrentCulture =
    Thread.CurrentThread.CurrentUICulture = culture;
} }
```

Figure 6 **The LoadEmbeddedResource Method**

```
private bool LoadEmbeddedResource(CultureInfo culture) {
  bool loaded = false;
  try {
    string resxFile = "ui." + culture.Name + ".resx";
    using (XmlReader xmlReader = XmlReader.Create(resxFile)) {
      var rs = ResxToDictionary(xmlReader);
      SetCulture(culture, rs);
      loaded = true;
   } }
   catch (Exception) {
     loaded = false;
  }
return loaded;
}
```

What we want to do in the LoadEmbeddedResource method is search for a file in the application in the format of resourceSet.culture.resx. If we find the file, we want to load it as an XmlDocument, parse it into a dictionary and then add it to the ResourceSets dictionary. **Figure 6** shows what the code looks like.

The SetCulture method is trivial; it updates the resource set if an entry exists or adds one if it doesn't.

Wrapping Up

This article rounded out the solution from Part 1, integrating server-side components to manage .xap and .resx files. With this solution, there's no need for Web service calls to retrieve the default resources. The idea of packaging the default resources in an application can be expanded to include any number of resources the user asks for.

> We've created a solution that lets us deploy a Silverlight application with resources for any language without making a single Web service call.

This solution decreases the maintenance needed for the resource strings. Generating .resx files from the database on-demand means there's little management needed for the .resx files. Rick Strahl has coded a useful localization tool that you can use to read the locale resources from the database, modify them and create .resx files! You'll find the tool at bit.ly/kfjtI2.

There are many places to hook into this solution, so you can customize it to do almost whatever you want. Happy coding! ■

Matthew Delisle *works for Schneider Electric on a leading-edge Silverlight application that deals with saving money by saving energy. Visit his blog at mattdelisle.net.*

John Brodeur *is a software architect for Schneider Electric with extensive Web development and application design experience.*

Thanks to the following technical expert for reviewing this article: *Shawn Wildermuth*

Secrets to Building a WPF Application in Windows PowerShell

Doug Finke

Windows PowerShell provides a new class of task automation. It doesn't displace old technologies so much as amplify them. Using Windows PowerShell (simply PowerShell hereafter for brevity) doesn't mean you should redo your application to take advantage of it. Rather, you can use PowerShell to seamlessly integrate and extend what you already have.

PowerShell is an automation technology presented as a command-line interface (CLI), scripting language and API.

In this article I'll walk through key PowerShell techniques and build a present value calculator (bit.ly/7oEij1) with a Windows Presentation Foundation (WPF) GUI (see **Figure 1**).

I'll introduce several key PowerShell elements: the WPF PowerShell Kit (WPK); the object pipeline; functions; PSObjects with properties; seamless integration with the Microsoft .NET Framework; modules and more.

This article discusses:

- Using PowerShell functions
- Ranges and pipelines
- Building a WPF GUI
- Importing modules
- Integrating .NET

Technologies discussed:

Windows PowerShell, Windows Presentation Foundation, Microsoft .NET Framework

Code download available at:

code.msdn.microsoft.com/mag201107PowerShell

PowerShell is built on the .NET Framework, letting you smoothly access the framework as you would from other .NET languages. Plus, you have access to the rest of the Windows OS and its components such as services, processes and Windows Management Instrumentation (WMI), and access to remote servers (that also have PowerShell version 2 and Windows Remote Management enabled).

All this is accomplished with the new scripting language exposed by PowerShell. All you need is Notepad and a few PowerShell cmdlets (pronounced "command-lets," a cmdlet is a lightweight command that's used in the PowerShell environment). The good news is, these are ready to go. The cmdlets are built into PowerShell and Notepad comes with Windows. The three key cmdlets are: Get-Help, which displays information about PowerShell commands and concepts; Get-Command, which gets basic information about cmdlets and other elements of PowerShell commands; and Get-Member, which gets the "members" (properties and methods) of objects.

It's all about discovery, and these cmdlets help you navigate this new task-automation platform.

Let's Get Started

Question: How many lines of code does it take to create a WPF app with a label saying "Hello World"?

Answer: Two, if you're using PowerShell and WPK, as shown in **Figure 2**.

That's a complete WPF application written in two lines of PowerShell. Line 1, Import-Module WPK, imports the WPK package, which contains a set of PowerShell cmdlets that wrap WPF. Interestingly, you don't need Visual Studio, XAML or C# to get this to work. You do need to install WPK, though (see next section).

PowerShell version 2 is available out of the box in Windows 7 and Windows Server 2008 R2 (and it's downloadable for older Windows systems). At the same time the client and server OSes were released, the PowerShell Pack was released (as a separate download), including the WPK. It's a hat tip to the popular Unix scripting tool, Tcl/Tk.

I'll start by building the application from a simple set of PowerShell variables to an interactive WPF application. I'll use the PowerShell Integrated Scripting Environment (ISE).

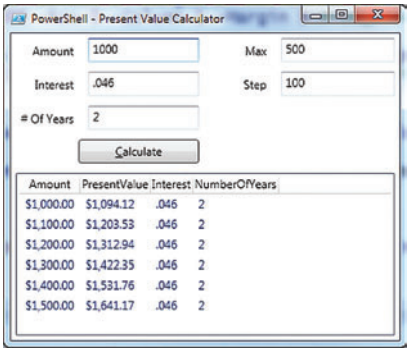


Figure 1 **The PowerShell Present Value Calculator**

Want to Follow Along?

If you have Windows 7, you're almost ready to go (remember, PowerShell is built-in).

If you're not running Windows 7, download and install PowerShell for older OSes. Be sure to choose the correct OS download. See the Windows Management Framework Core package (bit.ly/9POYjq).

No matter your OS version, you need to download and install the WPK (bit.ly/dFVpfL). Part of the Windows 7 Resource Kit, it contains nine other PowerShell modules, including an ISE Pack for use in the PowerShell ISE. The ISE is available out of the box with PowerShell version 2. The ISE Pack is a great learning resource as well, showing how to customize the ISE at several levels.

Figure 2 **A Two-Line PowerShell WPF Application**

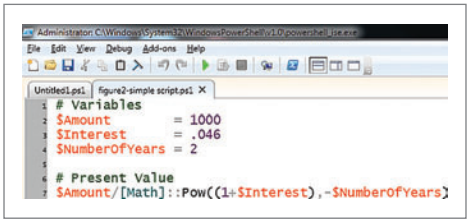


Figure 3 **Present Value Calculation**

Once you've launched PowerShell, run this cmdlet: Set-ExecutionPolicy RemoteSigned. Out of the box, PowerShell is set up to not run scripts; this is a security feature and PowerShell users need to override this. For the Set-ExecutionPolicy to work, you need to have administrator rights and explicitly run PowerShell as an administrator by right-clicking on the PowerShell program file and selecting "Run as administrator."

Download and unzip the scripts from the accompanying code download. The simplest way to run the application is to run it in the ISE. On Windows 7 you can click Start and type ISE. (Note: You can't run PowerShell scripts—which have a .ps1 file extension—by double-clicking on them. The easiest way to run the example scripts is to launch the ISE and use File | Open to open the script file.)

PowerShell 101

I'm going to build a present value calculator; this is a simple formula, shown in **Figure 3**.

Variables in PowerShell begin with a $. In line 7 I use the .NET Framework directly, calling the static method Pow on the System.Math namespace. The Pow method returns a specified number raised to the specified power. The syntax needed to call a static .NET method is brackets around the class followed by two colons and then the name of the method: [System.Math]::Pow(2,3). If you're running this in the ISE, press F5 (or click the Run button) to see the results in the output pane. You could also copy and paste this code into the PowerShell command-line console; it will run and print the result.

This is a great start but not very reusable. I could keep typing new values and running this script, but I want to call it from other scripts. I'll add the *function* keyword and turn the variables into parameters so I can vary the output and make the script interactive (see line 2 in **Figure 4**).

Adding the *Function* Keyword

I'm going to name the function Get-PresentValue. It's good practice to follow the PowerShell verb-noun convention when naming functions—this is a fundamental concept in using and developing for PowerShell. It has predefined verbs such as Get, Invoke, New and Remove (type Get-Verb to see the entire list). Try typing Get-Command; this returns all the cmdlets (and more) defined in your PowerShell session, and it's a huge list.

Creating a function is as simple as typing *function Get-PresentValue {}*. From here I'll create the parameters, with default values and the body of the function, as seen in **Figure 4**.

Comparing **Figure 3** to **Figure 4**, I transposed the variables—both the names and the values—to a single line and separated them by commas, wrapped them in parentheses and placed them after the name of the function, making them parameters to the function. These parameters now have default values. I left the present value calculation as is. A key tenet in PowerShell is "less typing." In line 3 in **Figure 4**, I could've used the *return* statement. Omitting it has the same behavior, although there are times you want to use the return statement to break the flow of logic.

In this example I show a few ways you can call the Get-PresentValue function. First, without parameters, all the parameters are defaulted. Parameters can be supplied either by position (see line 8 in **Figure 4**) or they can be named (see line 9). I recommend reading up on PowerShell parameters; PowerShell supports a powerful parameter-binding engine.

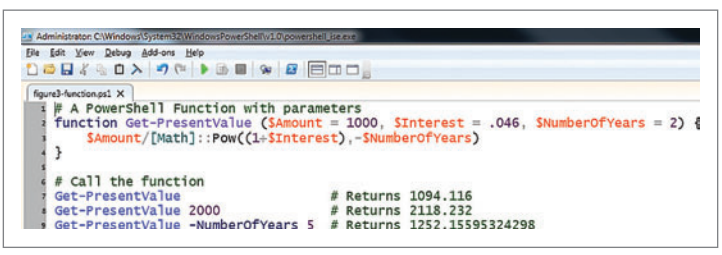


Figure 4 **Creating a PowerShell Function**

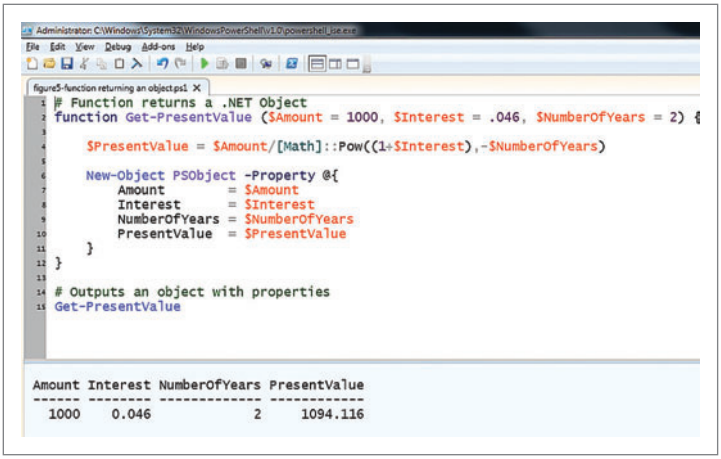


Figure 5 **Return Fully Typed Objects from PowerShell Functions**

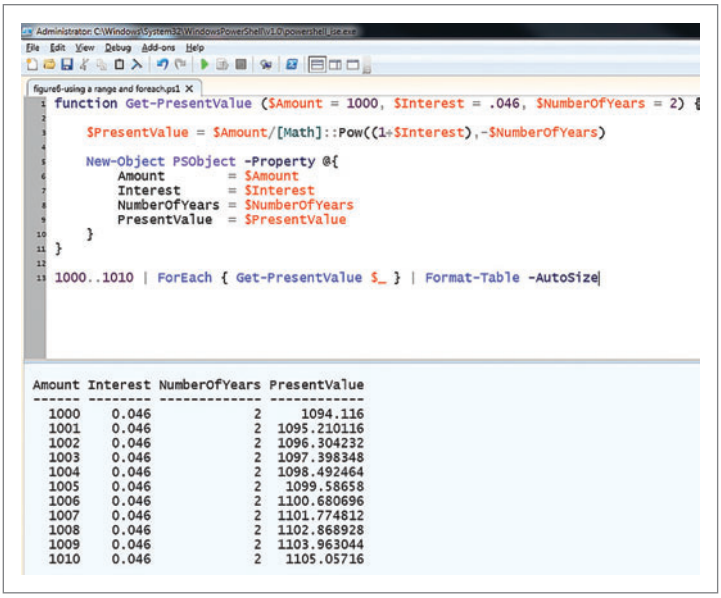


Figure 6 **PowerShell Ranges and Pipelines**

Next up: changing the Get-PresentValue function to return a .NET object rather than simple text.

PowerShell Is Based on .NET

A key innovation in PowerShell is the ability to pass data as fully typed objects. **Figure 5** introduces the concept of creating a PowerShell object, setting properties, returning the object and leveraging the PowerShell engine to print it out.

In line 6, I use the cmdlet New-Object, which creates an instance of a .NET Framework object. I tell it the type of object to create, a PSObject, which allows for a consistent view of any object within the PowerShell environment. Also in line 6, I'm using the -Property parameter, which takes a hash table. A shorthand syntax for hash tables in PowerShell is @{}. The key/value pairs defined in the hash table—and passed in the -Property parameter—are transformed into properties and values on the new PSObject.

Finally, in line 15, I call the function, and results can be seen in the output pane (shaded blue in the ISE). Notice that PowerShell "knows" how to print the object. I don't have to do any reflection to figure out what properties to print or how to print them—a key strength of PowerShell.

PowerShell Ranges and Pipelines

Next, I'll use PowerShell pipelines and present two more PowerShell cmdlets: ForEach-Object and Format-Table (see **Figure 6**).

Line 13 is the chewy piece and begins to give insight into the flexible and compositional quality of PowerShell. There are three sections and two pipes defined here. The first section shows the range operator (consisting of two periods), the second section is the ForEach and the last section contains the Format-Table. I'll discuss each.

> A key innovation in PowerShell is the ability to pass data as fully typed objects.

First Section, 1000..1010 *1000..1010* represents an array of integers from 1,000 to 1,010, inclusive. Now PowerShell will start pushing these one at a time down the pipeline as soon as one is available. PowerShell, like Unix/Linux-based shells, implements a pipeline, which enables the output of one cmdlet to be piped as input to another cmdlet. With PowerShell, the pipeline consists of .NET objects. Using objects eliminates the need to parse arbitrary text output from one command to extract data, as all objects export a consistent interface (see bit.ly/IVJarT).

Second Section, ForEach { Get-PresentValue $_ } This section uses ForEach (also aliased to %), which takes a scriptblock. Think of a scriptblock as an anonymous method (sometimes called lambda expressions in other languages). For more on this, see the book, "PowerShell in Action, Second Edition," by Bruce Payette (Manning Publications, 2011).

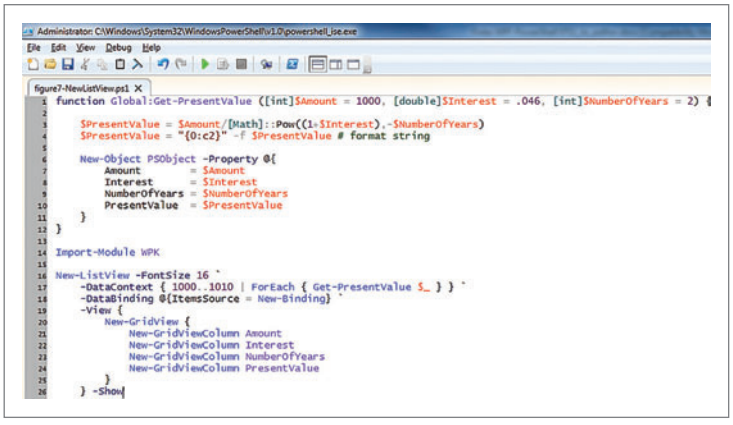
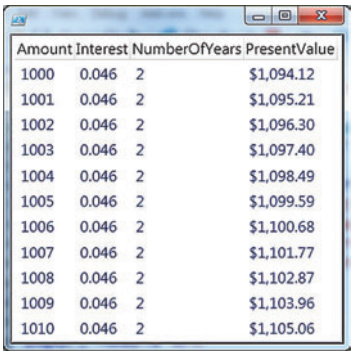


Figure 7 **The WPK New-ListView**

$_ is a PowerShell automatic variable and contains the current object in the pipeline. The end result, an array of 10 integers, is passed one integer at a time to Get-PresentValue. Because we aren't naming the parameter, I'm passing it as a positional parameter to $Amount, as seen in the output pane in **Figure 6**.

Last Section, Format-Table Format-Table does what it says; it formats the output as a table. I use the -AutoSize parameter because it adjusts the column size based on the width of the data.

Note that I'm not managing the iteration over the collection, and PowerShell "knows" how and what to print about the object being pushed over the pipeline. This results in writing fewer lines of code, which means fewer lines to debug. Because I spend 90 percent of my time debugging and the other 10 percent writing bugs, I come out nicely ahead.



Amount	Interest	NumberOfYears	PresentValue
1000	0.046	2	$1,094.12
1001	0.046	2	$1,095.21
1002	0.046	2	$1,096.30
1003	0.046	2	$1,097.40
1004	0.046	2	$1,098.49
1005	0.046	2	$1,099.59
1006	0.046	2	$1,100.68
1007	0.046	2	$1,101.77
1008	0.046	2	$1,102.87
1009	0.046	2	$1,103.96
1010	0.046	2	$1,105.06

Figure 8 **Viewing the GUI**

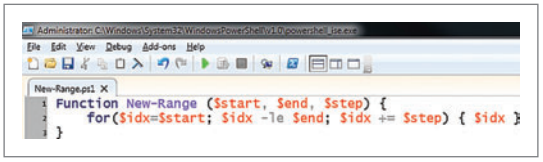


Figure 9 **The New-Range Function**

It's GUI Time—Round 1

It's time to use the WPK. The script in **Figure 7** produces the GUI in **Figure 8**. I added type information to the parameters of Get-PresentValue function (see line 1 in **Figure 7**). This helps others using this function to easily detect if they passed along wrong data—for example, strings rather than numerics.

The essence of the original code in **Figure 6** has been retained and the calling of Get-PresentValue has been added to the DataContext of the ListView, which is a WPF control providing the infrastructure to display a set of data items. The rest of the the WPK pieces integrate with WPF databinding and set up the view in the ListView so the data can be displayed.

The WPK follows the fundamental tenet of PowerShell, using the verb-noun pair. So, if I want to create a Window, Grid, Canvas or ListBox, I simply add "New-" to them—New-Window, New-Grid, New-Canvas or New-ListBox—and these framework elements are ready to use.

Import-Module

A module is a package that contains members—such as cmdlets, scripts, functions, variables, and other tools and files—that can be used in PowerShell. After a module is imported, you can use the module's members in your session. As noted earlier, the WPK is part of the PowerShell Pack, and the WPK contains more than 700 PowerShell functions that simplify layering a WPF GUI over PowerShell.

Coming from a traditional WPF background, lines 17 and 18 in **Figure 7** may seem unusual. The WPK supports data binding

The Ad Hoc Development Model and the Origin of Windows PowerShell

Doug Finke's article is an excellent example of the ad hoc development model. PowerShell differentiates itself from other programming technologies in many ways: its emphasis on high-level, task-oriented abstractions; its adaptive type system that normalizes different type systems (.NET, Windows Management Instrumentation [WMI], XML, ADSI, ADO and so on) and allows you to add members to types and instances; its dataflow engine that eliminates much of the API impedance mismatch code developers have to write; and its support for the ad hoc development model.

The ad hoc model is where you start off solving a problem using informal techniques. When you decide you're going to use it more, you convert it into an informal script, and if you share it, you make it more formal. As tool builders, we often build tools for people with different skill sets, so it's important to meet the needs and expectations of our entire audience. Often that means delivering a GUI.

Doug's article starts off with a nameless, hard-wired script to produce the PresentValue for a certain amount of money, a fixed interest rate and time. He then turns it into a named function with named parameters and initial values returning a single value. Next he returns an object so that it can be manipulated by other tools. And eventually he turns it into a simple GUI, and then a richer one. He only invests when he needs to, and each script adds small increments to the previous one. In the end, Doug shares his script, allowing other people to use his tool and also offer suggestions about how to make it better (such as, when suggested, he "typed" his parameters so his tool didn't barf if someone passed strings). We all benefit from sharing. I got a cool tool and owe a debt of gratitude to Doug. I partially repaid that debt by reviewing his code and offering suggestions. I know a bit about PowerShell, and yet I still benefit greatly from all the suggestions the community gives me on my scripts. [*Snover is the inventor of Windows PowerShell and one of the principal designers, along with Bruce Payette and James Truher; see* bit.ly/696Jor.—*Ed.*]

The ad hoc development model comes from the PowerShell goal to be both a great interactive shell and a great scripting language. Bruce Payette, one of the language designers, once said that the lifespan of 99 percent of PowerShell scripts starts with the command prompt and ends with the carriage return. PowerShell supports a wide range of scripting styles, starting from interactive one-liners at a command prompt to Bash-style functions using $args, to more formal scripts where parameters are named, typed and decorated with validation, data binding and help attributes. The reason we took this approach stems from my many years as a Unix developer, when I wrote tons of shell scripts. As people used my scripts and requested more features, I found myself throwing them away and rewriting in Tcl or Perl. Often I'd end up throwing those away as well and rewriting it all in C. It struck me that, while different problems require different levels of formalism and performance, it was insane that there wasn't a single tool that could span this wide range of scripting needs. That way, people could invest in becoming an expert in that tool versus being sort of competent in a large set of tools. It took awhile, but I finally got around to producing a tool that would do just that. I hope you enjoy PowerShell.

— Jeffrey Snover, Distinguished Engineer and Lead Architect for Windows Server

```
New-Window -WindowStartupLocation CenterScreen -Width 500 -Height 500 -DataContext {
    New-Range 1000 1500 100 | % { Get-PresentValue $_ }
} {

    New-ListView -Margin 5 -DataBinding @{ ItemsSource = New-Binding } -View {
        New-GridView -Columns {
            New-GridViewColumn Amount
            New-GridViewColumn PresentValue
            New-GridViewColumn Interest
            New-GridViewColumn NumberOfYears
        }
    }
} -Show
```

Figure 10 **The New-Window Function**

by surfacing two parameters: DataContext and DataBinding. The DataContext parameter takes a scriptblock, so here I pass a line of PowerShell code that creates the 10 present value objects I had in **Figure 6**, line 13. Next, I set up the ItemsSource property of the ListView to bind to in line 18. The DataBinding parameter takes a hash table (note the @{}). The keys are the names of the properties of the GUI object you want to bind to.

The WPK helps you write less code. The New-GridViewColumn function takes a string—for example, Amount (which is the name of a property on the object emitted from the Get-PresentValue function)—sets it as the Header and automatically does the data binding for you.

I started out with a simple function, Get-PresentValue, and ended up with the output shown in **Figure 8** by adding parameters to make it reusable, emitting a .NET PowerShell PSObject and leveraging the PowerShell object pipeline and range operator to generate present value items. Then I imported the WPK, injecting the PowerShell objects into the DataContext of a WPF ListView control, binding the ItemsSource parameter and creating the ListView view with column names that match the property names of the object injected. Finally, I have a simple present value PowerShell/WPK application. This is nice, but it's hardcoded.

Up next, I want to interact with this application so I can see what happens to my investment by changing the amount, interest and other parameters.

It's GUI Time—Round 2

The current PowerShell/WPK script requires me to change the parameters, save the file and rerun the script, which isn't very agile.

I'm going to rework it so I can tweak five parameters from the GUI and display new values in the ListView.

New-Range Function First up, I want to add a function called New-Range (see **Figure 9**).

The range operator PowerShell provides doesn't let you vary how much you increment by. In other words, I can't specify (1..10 by 2). The New-Range function will let me specify my own increment, so "New-Range 1 10 2" prints 1 3 5 7 9.

Next, I'll flesh out the WPK GUI by wrapping New-ListView in a New-Window function. Using the New-ListView by itself, the WPK will wrap in a window for you. Specifying New-Window gives me more control (see **Figure 10**).

I also lifted the DataContext from the ListView up to the Window scope and applied the new function New-Range. Now I want to add five text boxes, labels and a button. This will allow me to keep the application running and tweak the output of Get-PresentValue. I'm going to use the WPK New-Grid, New-Label, New-TextBox and New-Button functions to create the controls I need. Using New-Grid to create a grid control gives me flexibility in resizing the window and control placement.

> The WPK helps you write less code.

In order to lay out the GUI, I'm going to nest grids within grids as well as controls. I'm still using the New-Range and Get-PresentValue functions in the DataContext, as shown in line 2 of **Figure 11**.

Also, the New-ListView is still there in **Figure 11**, lines 30-37. I added two additional parameters in line 30, –Row and –Column, which tell the ListView which row and column it should be located in, in the New-Grid defined in line 5. The New-Grid defined in line 5 uses the Rows and Columns to lay out a grid. I want two columns with a width of 75 pixels and three rows with a height of 35 pixels each. The asterisk after the second 75 in the Rows parameter indicates it will take all the space available.

```
New-Window -WindowStartupLocation CenterScreen -Width 500 -Height 500 -DataContext {
    New-Range 1000 1500 100 | % { Get-PresentValue $_ }
} {

    New-Grid -Rows 105, 35, 100* -Columns 200, 200* {

        New-Grid -Row 0 -Column 0 -Columns 75, 75* -Rows 35, 35, 35 {

            New-Label   -Content "Amount" -Row 0 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtAmount -Text 1000 -Row 0 -Column 1 -Margin 5

            New-Label   -Content "Interest" -Row 1 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtInterest -Text .046 -Row 1 -Column 1 -Margin 5

            New-Label   -Content "# Of Years"  -Row 2 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtNumberOfYears -Text 2 -Row 2 -Column 1 -Margin 5
        }

        New-Grid -Row 0 -Column 1 -Columns 75, 75* -Rows 35, 35{

            New-Label   -Content "Max" -Row 0 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtMax -Text 500 -Row 0 -Column 1 -Margin 5

            New-Label   -Content "Step" -Row 1 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtStep -Text 100 -Row 1 -Column 1 -Margin 5
        }

        New-Button -Content "_Calculate" -Row 1 -Column 0 -Margin 5 -Width 125 -HorizontalAlignment Right

        New-ListView -Row 2 -Column 0 -ColumnSpan 2 -Margin 5 -DataBinding @{ ItemsSource = New-Binding } -View {
            New-GridView -Columns {
                New-GridViewColumn Amount
                New-GridViewColumn PresentValue
                New-GridViewColumn Interest
                New-GridViewColumn NumberOfYears
            }
        }
    }
} -Show
```

Figure 11 **The New-Grid Function**

```
Function Do-Calculation ($window) {

    $amount        = ($window | Get-ChildControl txtAmount).Text
    $interest      = ($window | Get-ChildControl txtInterest).Text
    $numberOfYears = ($window | Get-ChildControl txtNumberOfYears).Text
    $max           = ($window | Get-ChildControl txtMax).Text
    $step          = ($window | Get-ChildControl txtStep).Text

    $window.DataContext = (
        New-Range $amount ([int]$amount+$max) $step |
            ForEach { Get-PresentValue $_ $interest $numberOfYears }
    )
}
```

Figure 12 **The Do-Calculation Function**

Now I place five pairs of labels and text boxes in the window, telling the controls what quadrant of the grid to anchor to via the Row and Column parameters. Also, I name the text box so I can access it later, give it a default value with the -Text parameter and embellish the controls with the Margin and HorizontalAlignment parameters.

Finally, I place the button in the grid with the New-Button function. Notice I place an underscore in front of Calculate. This lets me access the button with the keystroke Alt-C.

The present value calculator is almost complete. I have to hook up the click event to some PowerShell code, read the values in the text boxes, pass them as parameters to the Get-PresentValue and bind it to the ListView.

Do-Calculation Function

I'm going to add a Do-Calculation function that takes a single parameter, $window (see **Figure 12**).

I'll call Do-Calculation from the -On-Click property on the New-Button, the one with the content "_Calculate."

Do-Calculation is straightforward. It grabs the information in all the text boxes, sets them to PowerShell variables and passes them as parameters to the New-Range and Get-PresentValue functions. I pass the parameter $window (see line 2 in **Figure 13**), which contains a reference to the New-Window created in **Figure 10**. Using this, I can get to all the properties of the window as well as the child controls, specifically the text boxes. Because I named each of the text box controls, I can pipe $window to Get-ChildControl, passing the name of the control and retrieving the text through the .Text property (see lines 3-7 in **Figure 12**).

Gathering all the details from the text boxes, I set the $window.DataContext to the result of the New-Range being piped to Get-PresentValue, which generates an array of PowerShell objects containing the results of PresentValue calculations (see lines 9-12 in **Figure 12**).

Figure 13 shows how to connect up the Calculate button's click event to call the Do-Calculation function and pass the $window variable (see lines 28-29). I added the -On_Click parameter, which takes a ScriptBlock. In there I call the Do-Calculation function, passing in the $window variable that's created for me when I use the New-Window function. Every time I click the button, the screen will recalculate. Note that I also changed line 2 in **Figure 13** to also call the Do-Calculation function.

Download the completed application in the accompanying source code samples to see the entire script.

> PowerShell manifests in both a scripting language and command-line console.

A Simple, Interactive WPF Application

I presented here a script of fewer than 75 lines of code, resulting in an interactive WPF application layered over the Microsoft automation platform, PowerShell. PowerShell manifests in both a scripting language and command-line console. Its deep integration with both the .NET Framework and Windows enables exciting automation opportunities. Of course, this means one must invest time in learning this new platform. The good news: You can engage with the low-hanging fruit to become more productive and then, when you need to, deep dive into the large offering of automation that PowerShell provides, both from Microsoft and the general PowerShell community. ■

```
New-Window -Title "PowerShell - WPK Present Value Calculator" -WindowStartupLocation CenterScreen -Width 500 -Height 500 -On_Loaded {
    Do-Calculation $window
} {

    New-Grid -Rows 105, 35, 100* -Columns 200, 200* {

        New-Grid -Row 0 -Column 0 -Columns 75, 75* -Rows 35, 35, 35 {

            New-Label   -Content "Amount" -Row 0 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtAmount -Text 1000 -Row 0 -Column 1 -Margin 5

            New-Label   -Content "Interest" -Row 1 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtInterest -Text .046 -Row 1 -Column 1 -Margin 5

            New-Label   -Content "# Of Years"  -Row 2 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtNumberOfYears -Text 2 -Row 2 -Column 1 -Margin 5
        }

        New-Grid -Row 0 -Column 1 -Columns 75, 75* -Rows 35, 35{

            New-Label   -Content "Max" -Row 0 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtMax -Text 500 -Row 0 -Column 1 -Margin 5

            New-Label   -Content "Step" -Row 1 -Column 0 -Margin 5 -HorizontalAlignment Right
            New-TextBox -Name txtStep -Text 100 -Row 1 -Column 1 -Margin 5
        }

        New-Button -Content "_Calculate" -Row 1 -Column 0 -Margin 5 -Width 115 -HorizontalAlignment Right -On_Click {
            Do-Calculation $window
        }

        New-ListView -Row 2 -Column 0 -ColumnSpan 2 -Margin 5 -DataBinding @{ ItemsSource = New-Binding } -View {
            New-GridView -Columns {
                New-GridViewColumn Amount
                New-GridViewColumn PresentValue
                New-GridViewColumn Interest
                New-GridViewColumn NumberOfYears
            }
        }
    }
} -Show
```

Figure 13 **Completed Grid**

DOUG FINKE, *a Microsoft MVP for Windows PowerShell, is a software developer at Lab49, a company that builds advanced applications for the financial services industry. For the past 20 years, he has been a developer and author working with numerous technologies. You can catch up with him at his blog, Development in a Blink, at dougfinke.com/blog.*

THANKS to the following technical experts for reviewing this article: *James Brundage, Sal Mangano, Sivabalan Muthukumar, Marco Shaw, Jeffrey Snover and Sylvain Whissell*

TEAM FOUNDATION SERVER HOSTING IS HERE!

ONLY $20/mo per user

LIMITED TIME SPECIAL OFFER: FIRST 30 DAYS FREE & NO SETUP FEES

Instead of spending your valuable time and resources maintaining a TFS server environment, you have a NEW option - to use the DiscountASP.NET Hosted TFS SaaS solution The DiscountASP.NET Hosted TFS service is a SaaS solution for source code version control and bug tracking. Start saving money and time so you can focus on what you do best – developing killer software!

SIMPLIFY YOUR LIFE CYCLE TODAY!

TFS HOSTING FEATURES:

- Source Code Version Control
- Bug/Issue Tracking
- Web Team Access
- Unlimited Projects
- 5gb of Disk Space
- Visual Studio 2010/2008 Integration
- Web-Based Control Panel
- Multi-User Discounts
- Month-to-Month Billing
- Data Centers Available in the USA & Europe

NOW AVAILABLE:

TFS Build Server

FREE VSS to hosted TFS migration

FOR MORE INFO: www.DiscountASP.NET/tfs/simplify

discountASP.net
Team Foundation Server Hosting

Members Choice Winner The Code Project 2011

DevPro 2010 Community Choice GOLD

2010 Readers Merit Visual Studio Magazine

Microsoft Partner Gold Hosting

Easily Add Performance Counters to Your MVC Application

Ben Grover

Working on enterprise Web applications usually involves a host of additional code to help with monitoring and operation of the apps. In this article, I'll explain how I'm using Model-View-Controller (MVC) filters to clean up and replace repeated, confusing code that was spread throughout numerous methods in an application.

Operational managers often set up Microsoft Operations Manager (MOM) to monitor the health of a Web site (or service) and use performance counters to trigger alarms based on threshold values. These alarms help ensure that degraded experiences on the Web site are found quickly.

Problem Code

I'm working on a related project (using the ASP.NET MVC Framework) where the requirements are to add performance counters to Web pages and Web services to aid the operations team. The operations team requires counters for each page: Request Latency, Total Requests per Second and Failure Ratio.

Problems always seem to arise with the implementation of such requirements. I started looking at the current implementation of these counters from more diligent developers who had added them as part of previous coding milestones. I was disappointed. I'm sure you've all been there—you look at the code and cringe. What did I see? Repeated code sprinkled through *each* method, with just a few changes to a variable name here and there. I wasn't happy with the current implementation.

The code that made me cringe can be seen in **Figure 1**.

Looking at this code, I knew I wanted to remove it from each of my project's action methods. This type of pattern makes it extremely hard to see where the true method code is because of all the superfluous code to account for performance monitoring. I was looking for a clever way to refactor this code so that it wouldn't litter each of the action methods. Enter MVC filters.

This article discusses:

- Problem code
- MVC filters
- Managing performance counters
- Implementing the design

Technologies discussed:

ASP.NET MVC, Reflection

MVC Filters

MVC filters are custom attributes that you put onto action methods (or controllers) to add common functionality. MVC filters allow you to add pre- and post-processing behaviors. The list of built-in MVC filters can be found here: bit.ly/jSaD5N. I had used some of the built-in filters, such as OutputCache, but I knew that MVC filters had a lot of power hidden that I'd never tapped into (to read more about the filter attribute, see bit.ly/kMPBYB).

Figure 1 **Code that Made Me Cringe**

```
public ActionResult AccountProfileInformation()
{
  try
  {
    totalRequestsAccountInfoCounter.Increment();
    // Start counter for latency
    long startTime = Stopwatch.GetTimestamp();

    // Perform some operations action here
    long stopTime = Stopwatch.GetTimestamp();
    latencyAccountInfoCounter.IncrementBy((stopTime - startTime) /
      Stopwatch.Frequency);
    latencyAccountInfoBaseCounter.Increment();
  }
  catch (Exception e)
  {
    failureAccountInfoCounterCounter.Increment();
  }
  return View();
}
```

So, I began thinking to myself: What if I could encapsulate all of the logic for these performance counters in an MVC filter attribute? An idea was born! I could meet the requirements of each performance counter listed earlier with the following actions:

1. Total Requests per Second
   a. Implement IActionFilter, which has two methods: OnActionExecuting and OnActionExecuted
   b. Increment the counter in the OnActionExecuting
2. Request Latency
   a. Implement IResultFilter, which has two methods: OnResultExecuting and OnResultExecuted
   b. Start the timer in the OnActionExecuting and record the latency during OnResultExecuted
3. Failure Ratio
   a. Implement IExceptionFilter, which has the method OnException

The process is illustrated in **Figure 2**.

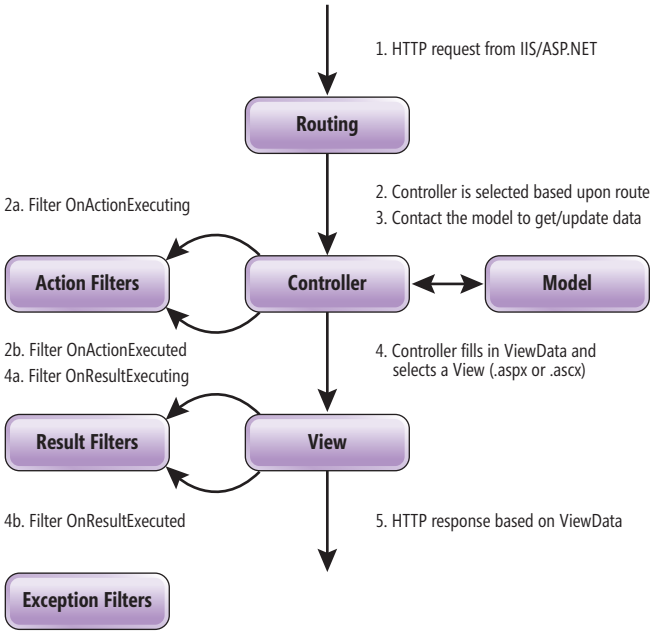


Figure 2 **MVC Filter-Processing Pipeline**

I'll discuss the use of each filter as shown in **Figure 2**.

IActionFilter OnActionExecuting (line 2a) executes before the action method is executed. OnActionExecuted (line 2b) executes after the action method is executed, but before the result is executed.

IResultFilter OnResultExecuting (line 4a) executes before the action result is executed (such as rendering a view). OnResultExecuted (line 4b) executes after the action result is executed.

IExceptionFilter OnException (not shown in **Figure 2** for clarity) executes whenever an exception is thrown (and unhandled).

IAuthorizationFilter OnAuthorization (not included in **Figure 2**, nor used in this article) is called when authorization is required.

Managing the Counters

However, if I used this performance counter filter attribute, I'd have a problem: How would I get the performance counter (for each action) into each of these filters at run time? I didn't want to have a separate filter attribute class for each action. In that case, I'd have to hardcode the performance counter name in the attribute. That would cause an explosion in the number of class names needed to implement

Figure 3 **Reflecting Over Assemblies**

```
/// <summary>
/// This method reflects over the given assembly(ies) in a given path
/// and creates the base operations required  perf counters
/// </summary>
/// <param name="assemblyPath"></param>
/// <param name="assemblyFilter"></param>
public void Create(string assemblyPath, string assemblyFilter)
{
  counterMap = new Dictionary<string, PerformanceCounter>();

  foreach (string assemblyName in Directory.EnumerateFileSystemEntries(
    assemblyPath, assemblyFilter))
  {
    Type[] allTypes = Assembly.LoadFrom(assemblyName).GetTypes();

    foreach (Type t in allTypes)
    {
      if (typeof(IController).IsAssignableFrom(t))
      {
        MemberInfo[] infos = Type.GetType(t.AssemblyQualifiedName).GetMembers();

        foreach (MemberInfo memberInfo in infos)
        {
          foreach (object info in memberInfo.GetCustomAttributes(
            typeof(WebCounterAttribute), true))
          {
            WebCounterAttribute webPerfCounter = info as WebCounterAttribute;
            string category = webPerfCounter.Category;
            string instance = webPerfCounter.Instance;
            // Create total rollup instances, if they don't exist
            foreach (string type in CounterTypeNames)
            {
              if (!counterMap.ContainsKey(KeyBuilder(Total, type)))
              {
                counterMap.Add(KeyBuilder(Total, type),
                  CreateInstance(category, type, Total));
              }
            }
            // Create performance counters
            foreach (string type in CounterTypeNames)
            {
              counterMap.Add(KeyBuilder(instance, type),
                CreateInstance(category, type, instance));
            }
          }
        }
      }
    }
  }
}
```

Figure 4 **WebCounterManager RecordLatency**

```
/// <summary>
/// Record the latency for a given instance name
/// </summary>
/// <param name="instance"></param>
/// <param name="latency"></param>
public void RecordLatency(string instance, long latency)
{
  if (counterMap.ContainsKey(KeyBuilder(instance,
    CounterTypeNames[(int)CounterTypes.AverageLatency]))
    && counterMap.ContainsKey(KeyBuilder(instance,
    CounterTypeNames[(int)CounterTypes.AverageLatencyBase])))
  {
    counterMap[KeyBuilder(instance,
      CounterTypeNames[(int)CounterTypes.AverageLatency])].IncrementBy(latency);
    counterMap[KeyBuilder(Total,
      CounterTypeNames[(int)CounterTypes.AverageLatency])].IncrementBy(latency);
    counterMap[KeyBuilder(instance,
      CounterTypeNames[(int)CounterTypes.AverageLatencyBase])].Increment();
    counterMap[KeyBuilder(Total,
      CounterTypeNames[(int)CounterTypes.AverageLatencyBase])].Increment();
  }
}
```

the solution. I reflected back on a technology I used when I had first worked with the Microsoft .NET Framework: reflection (pun intended!). Reflection is heavily leveraged by the MVC framework as well. You can learn more about reflection here: bit.ly/iPHdHz.

My idea was to create two classes:

1. WebCounterAttribute
   a. Implements the MVC filter interfaces (IException-Filter, IActionFilter and IResultFilter)
   b. Increment counters stored in WebCounterManager
2. WebCounterManager
   a. Implements the reflection code for loading the WebCounterAttributes from each MVC action
   b. Stores a map to facilitate the lookup of the performance counter objects
   c. Provides methods to increment the counters stored in that map

Reflection is heavily leveraged by the MVC framework.

Implementing the Design

Having those classes, I could decorate WebCounterAttribute on each of the action methods on which I wanted performance counters to be implemented, as shown here:

```
public sealed class WebCounterAttribute : FilterAttribute,
IActionFilter, IExceptionFilter, IResultFilter
{
  /// Interface implementations not shown
}
```

Here's a sample action method:

```
[WebCounter("Contoso Site", "AccountProfileInformation")]
public ActionResult AccountProfileInformation()
{
  // Some model loading
  return View();
}
```

Then I could read in these attributes during the Application_Start method using reflection and create a counter for each of these actions, as shown in **Figure 3**. (Note that counters are registered with the system in a setup program, but instances of the counters are created in code.)

Figure 5 **The WebCounterAttribute Invoking the WebCounterManager RecordLatency**

```
/// <summary>
/// This method occurs when the result has been executed (this is just
/// before the response is returned).
/// This method records the latency from request begin to response return.
/// </summary>
/// <param name="filterContext"></param>
public void  OnResultExecuted(ResultExecutedContext filterContext)
{
  // Stop counter for latency
  long time = Stopwatch.GetTimestamp() - startTime;
  WebCounterManager countManager = GetWebCounterManager(filterContext.HttpContext);
  if (countManager != null)
  {
    countManager.RecordLatency(Instance, time);
    ...
  }
}
private WebCounterManager GetWebCounterManager(HttpContextBase context)
{
  WebCounterManager manager =
    context.Application[WebCounterManager.WebCounterManagerApplicationKey]
    as WebCounterManager;
  return manager;
}
```

Notice the important line where the map is populated:

```
(counterMap.Add(KeyBuilder(instance, type), CreateInstance(category,
type, instance));),
```

It creates a mapping between the particular instance of a WebCounterAttribute on an action, including the counter type, and maps it to the created PerformanceCounter instance.

I could then write the code that enables me to use this mapping to look up the PerformanceCounter instance (and increment it) for a given instance of the WebCounterAttribute (see **Figure 4**).

Then I could record the performance counter data when these filters are run. For example, in **Figure 5**, you see an implementation of recording performance latency.

You'll note in this call that I'm getting the WebCounterManager from the Application State. In order for this to work, you'll need to add code to your global.asax.cs:

```
WebCounterManager webCounterMgr = new WebCounterManager();
webCounterMgr.Create(Server.Map("~/bin"), "*.dll");
Application[WebCounterManager.WebCounterManagerApplicationKey] = webCounterMgr;
```

Wrapping up, MVC filters provide an elegant solution to heavily repeated code patterns. They'll help you refactor common code that will make your code cleaner and easier to maintain. Obviously, a balance has to be struck between elegance and ease of implementation. In my case, I had to add performance counters to about 50 Web pages. The savings in terms of code readability and clarity was definitely worth the additional effort.

MVC filters are a great way to add behaviors without being obtrusive, so whether you're dealing with performance counters, logging or auditing, you'll find limitless possibilities for clean implementation of necessary logic. ■

Ben Grover *is a programmer at Microsoft in Redmond, Wash., where he has worked on multiple teams, from Exchange to Lync to Windows.*

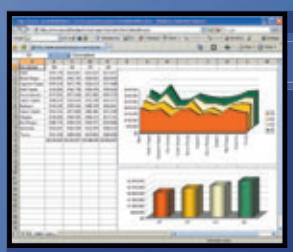
Thanks to the following technical expert for reviewing this article: *Eilon Lipton*

Microsoft Chose SpreadsheetGear...

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

Chris Donohue, MSN Money Program Manager

SpreadsheetGear 2010



ASP.NET Excel Reporting

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

Excel Compatible Windows Forms Control

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

Create Dashboards from Excel Charts and Ranges

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

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

www.SpreadsheetGear.com.

SpreadsheetGear

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

Build a Better Mobile Browsing Experience

Steven Sanderson

Who accesses the Web through a mobile device these days? In 2005, you'd have pictured the average mobile Internet user as a geeky, affluent westerner—probably a software developer—who'd take the time to connect his bulky cell phone to a slow data network to endure a painfully limited browsing experience—and pay top-dollar for the privilege. In other words, an edge case.

Now mobile Web access has skyrocketed into the global mainstream. And I don't just mean the teenagers, students and retired folks showing each other their smartphones and tablet devices in every coffee shop across Europe and North America. Today there are around 1 *billion* active mobile broadband subscriptions, enough for around one in seven people on the planet (and only two in seven regularly use the Internet by any means). Mobile devices are on track to become the single most common way to access the Web within five years. Already, in some of the fastest-growing countries—especially India—mobiles are the only way for many people to get online. Even in America, 25 percent of mobile Web users say they "never" or "infrequently" access the Web using a traditional PC. (For information sources, see "Mobile Web Access.")

Clearly, if you're building a public Web site, you need to think about supporting mobile browsers.

This article discusses:

- Mobile support in ASP.NET
- Detecting browser types
- Controlling page rendering on mobile browsers
- Varying output according to browser type

Technologies discussed:

ASP.NET

Why Mobile Browsing Is Different

As you know, just about every mobile browser supports some form of HTML. Many, especially on high-end devices such as iPhones and Windows Phone 7, support the latest HTML, CSS and JavaScript standards and render pixel-perfect copies of what you'd see in a traditional PC browser.

Your cheapest option for supporting mobile browsers, then, is to do nothing. You can just serve the same desktop-oriented pages for all devices, and trust mobile browsers to handle them. But choosing this option leads to a very poor mobile browsing experience, for several reasons:

- **Mobile phone screens are small.** Some mobile browsers, such as Opera Mini, handle desktop-width pages by dynamically reformatting the page layout and styles. The resulting appearance is rarely what your designer had in mind. Other mobile browsers, such as Safari for iPhone or Internet Explorer for Windows Phone 7, render desktop-width pages and then force the user to zoom in and out and pan around to read the text. This is a test of your visitors' patience.

- **Mobile data networks are often slow.** Don't assume that your visitors have the same bandwidth as fixed-line broadband users. They may even be paying by the megabyte, so heavyweight sites won't be popular.
- **Mobile devices often don't have a mouse or keyboard.** Familiar desktop user-interaction mechanisms don't always make sense on mobile devices. For example, clicking on small links or buttons may be difficult and error-prone on touch-oriented devices, and the concept of "hovering" may not even exist.

So, if you want to provide a first-rate mobile browsing experience, it's time to apply your engineering skills and account for the differences between major device types.

Mobile Support in ASP.NET

There are two main aspects to supporting mobile browsers:

1. **Detecting which kind of device a given visitor is using.** ASP.NET has built-in support for browser detection. In the next section, I'll examine this mechanism, and how it can be customized and extended.
2. **Producing output that works well on the detected device.** If you scan back through the preceding list of challenges, you'll realize that these aren't things your technology platform can handle automatically. Mobile support is primarily a matter of user experience (UX) design, *not* primarily a matter of markup. Later in this article I'll describe technical means to produce different outputs for different devices, but it's still up to you to design and implement different layouts and user workflows for mobiles.

Note that until around ASP.NET 2.0, released in 2005, producing output to work on mobile devices was a matter of markup, because common devices at that time used specialist protocols and markup languages, including WAP, WML and cHTML. ASP.NET 2.0 contained "Mobile Controls" to support those formats. However, those formats are now entirely obsolete, because all mainstream browsers now use HTML, so the ASP.NET Mobile Controls are also obsolete.

> The core ASP.NET platform, which underpins both Web Forms and MVC, has built-in support for browser detection.

Browser Detection

The core ASP.NET platform, which underpins both Web Forms and Model-View-Controller (MVC), has built-in support for browser detection. You can find out whether or not a visitor is using a mobile browser using the Request.Browser.IsMobileDevice Boolean property. However, you should understand how this detection works to be aware of accuracy limitations that may affect you.

ASP.NET determines what kind of browser is making a request and what capabilities that browser has (screen size, JavaScript support and so on) by comparing the incoming request userAgent header string against a series of regular expressions in XML files that describe common browsers.

Figure 1 **The iphone.browser File**

```
<browsers>
  <!-- Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+
       (KHTML, like Gecko) Version/3.0 Mobile/1A543a Safari/419.3 -->
  <gateway id="IPhone" parentID="Safari">
    <identification>
      <userAgent match="iPhone" />
    </identification>
    <capabilities>
      <capability name="mobileDeviceModel" value="IPhone" />
      <capability name="mobileDeviceManufacturer" value="Apple" />
      <capability name="isMobileDevice" value="true" />
      <capability name="canInitiateVoiceCall" value="true" />
    </capabilities>
  </gateway>
  ...
</browsers>
```

These regular expressions—and information about corresponding device capabilities—are stored in a set of .browser files in the folder C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\Browsers (or your installation's equivalent). For example, the standard iphone.browser file includes the code shown in **Figure 1**.

> Your cheapest option for supporting mobile browsers is to do nothing.

In the file, the following element defines the regular expression to be matched against incoming userAgent header strings:

```
<userAgent match="iPhone" />
```

Once the system finds a matching userAgent regular expression, the remainder of the XML data specifies the type and capabilities of that device.

The limitation of this system is clear: It can only detect mobile devices that were known and described in these files when ASP.NET 4.0 was first released. Unfortunately, this does not include common modern browsers such as Opera Mobile or the default browser for Google Android. Request.Browser.IsMobileDevice will incorrectly be set to false for those browsers.

Customizing and Enhancing Browser Detection

You have two main options for overcoming the limitations of the default browser-detection facility:

1. You can supply your own .browser files to represent newer devices.
2. You can use a third-party browser-detection library.

To take the first option, right-click on your project's name in the Visual Studio Solution Explorer and choose Add | Add ASP.NET Folder | App_Browsers. You can then add .browser files to that folder; for example, by copying an existing file from your C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\Browsers folder and then editing its identification regular expression and device capability description to represent the desired browser.

If you don't want to be responsible for tracking all of the hundreds of newly released mobile browsers and keeping your .browser files up-to-date, you can take the second option and use a third-party browser-detection library.

Currently, the one I'm recommending is 51degrees.Mobi Foundation, an open source (MPL license) library hosted on CodePlex at 51degrees.codeplex.com. This library does not use .browser files. Instead, it identifies devices by matching them against the Wireless Universal Resource File (WURFL) database, which can be used free of charge in both commercial and non-commercial applications. For more information about WURFL, see wurfl.sourceforge.net.

The easiest way to install 51degrees.Mobi Foundation into either Web Forms or MVC projects is by using the NuGet package manager. If you're running ASP.NET MVC 3, you already have NuGet. If not, you can use the Visual Studio Extension Manager (it's on the Tools menu) to search for and install NuGet. Once you have NuGet, go to Tools | Library Package Manager | Package Manager Console, and then issue the following command in the console:

```
Install-Package 51Degrees.mobi
```

This adds to your project:

- A recent copy of the WURFL database to your project at /App_Data/wurfl.xml.gz
- A reference to FiftyOne.Foundation.dll, the library's main assembly
- Web.config entries to enable 51degrees.Mobi Foundation

51degrees.Mobi Foundation plugs into and enhances the ASP.NET standard Request.Browser API. Just by having the package installed, you'll get much more accurate results from Request.Browser.IsMobileDevice, because recent versions of the WURFL database can detect today's common mobile browsers, including Opera Mobile and the Google Android browser.

> Many modern mobile browsers try to make rendered pages look just as they do on a desktop browser.

Note that the default 51degrees.Mobi Foundation Web.config settings also configure it to redirect all requests from mobile browsers to the URL ~/Mobile/Default.aspx. In many cases—and especially for ASP.NET MVC applications—that won't be the behavior you want. You can disable the redirection by commenting out or deleting the <redirect /> element that the package adds to your Web.config file, and then you can also delete its /Mobile/Default.aspx file if you want.

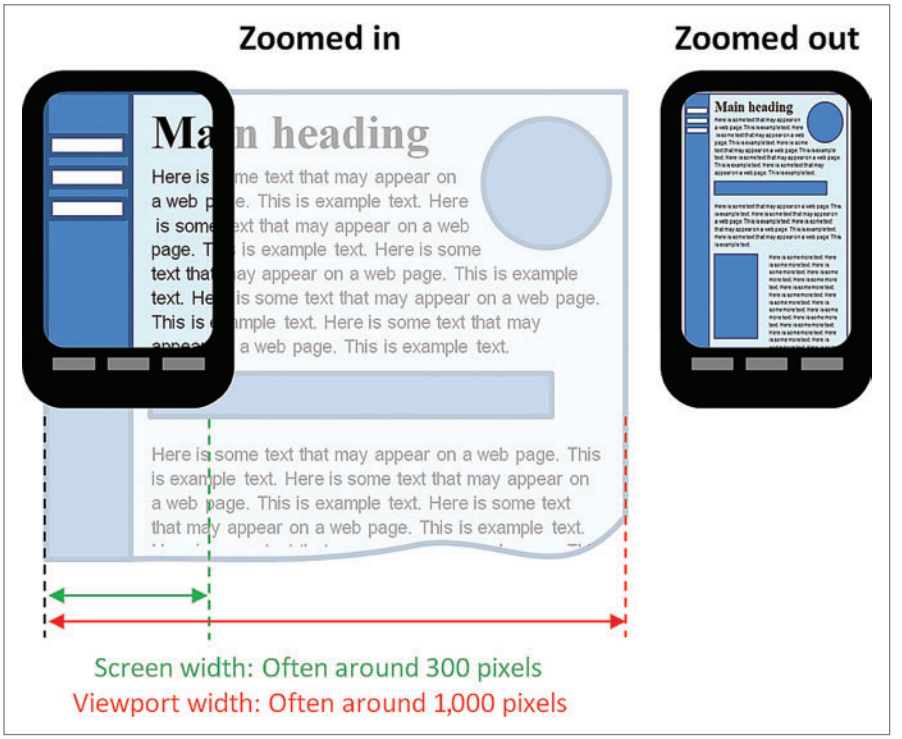


Figure 2 **A Mobile Browser Rendering a Desktop-Width Page onto a Virtual Viewport**

Styling for Mobiles

Now that you have an idea of how to detect mobile browsers reliably, I'll show a key way to control how pages are rendered by mobile browsers. After that, I'll describe some architectural options for varying the rendered markup by device type.

Many modern mobile browsers, including Safari for iOS and Internet Explorer for Windows Phone 7, try to make rendered pages look just as they do on a desktop browser. They know that most pages are designed for screens around 1,000 pixels wide and the designer has most likely not accounted for much smaller widths.

To solve this, they typically render the page onto a virtual canvas known as a "viewport," usually around 1,000 virtual pixels wide. The browser can then scale the visual display of that virtual canvas arbitrarily, allowing the user to zoom in and out and pan around. This arrangement is illustrated in **Figure 2**.

While this allows the page to render as the designer intended, it suffers the significant usability drawback that when the user is zoomed in sufficiently to read the text, he can't see the full width of the page and must scroll horizontally to do so.

Controlling the Viewport Width

If you've actually designed pages for the small screen, you won't want them to be laid out on a virtual viewport around 1,000 pixels wide. Instead, you'll want your pages to be laid out on a viewport that's the same width as the actual screen, so that it neatly fits horizontally with no zooming required.

Many of the most popular mobile browsers support a nonstandard "viewport" meta tag that lets you control the width of the virtual viewport. For example, if you add the following to your page's <head> section, the browser will lay out the page on a viewport 320 pixels wide:

```
<meta name="viewport" content="width=320"/>
```

This is usually a much better fit for mobile phones.

Keep in mind that some mobile devices have screens with much higher horizontal resolution. For example, the iPhone 4 has 640 physical pixels per row. However, it still makes sense to use a virtual viewport of around 320 pixels; otherwise, the resulting text will be too small to read without zooming in.

If you want, you can let the virtual viewport vary in size according to the device being used, using the following syntax:

```
<meta name="viewport" content="width=device-width"/>
```

> Many mobile browsers support a nonstandard "viewport" meta tag that lets you control the width of the virtual viewport.

Note that some mobile devices won't give you a literal device width. They interpret "device-width" as meaning "the virtual viewport width that the manufacturer thinks gives the most pleasing result." So, for example, iPhone 4 defines device-width as 320 pixels, despite its higher physical resolution.

Markup Recommendations

Whenever you're designing pages for mobile browsers:

- Use the viewport meta tag to make the viewport fit the horizontal width of the screen.
- Adjust your page layouts, CSS styles and the like to account for this narrow width. If visitors don't need to zoom or scroll horizontally, your page feels more like a native application designed for their device—a far better experience.
- Make sure your links and buttons are large enough to be tapped imprecisely. Fingertips are much bigger than the tip of a mouse pointer.
- Minimize bandwidth requirements by not using very high-resolution images or massive JavaScript files.

Architectural Options

You've seen how to detect mobile browsers, and I've provided some recommendations for markup that better suits them. Now I'll describe three straightforward options for structuring your application to produce different output for different browser types:

1. Showing and hiding sections of markup according to browser type.
2. Switching master pages according to browser type.
3. Presenting entirely different content according to browser type.

Each of these has its benefits and limitations, so it's up to you to pick an approach that suits your requirements.

Showing and Hiding Markup

If you only need to include or exclude meta tags and CSS file references according to browser type, this is extremely simple. For example, in a Web Forms master page, you can add an "if" statement inside your <head> section:

```
<head runat="server">
  <title>My site</title>
  <link href="~/Styles/Site.css" rel="stylesheet" type="text/css" />

  <% if (Request.Browser.IsMobileDevice) { %>
    <meta name="viewport" content="width=device-width"/>
    <link href="~/Styles/MobileSite.css" rel="stylesheet" type="text/css" />
  <% } %>
</head>
```

The equivalent for a Razor layout for an ASP.NET MVC 3 application looks like this:

```
<head>
  <title>@ViewBag.Title</title>
  <link href="@Url.Content(
    "~/Content/Site.css")" rel="stylesheet" type="text/css" />
  @if (Request.Browser.IsMobileDevice) {
    <meta name="viewport" content="width=device-width"/>
    <link href="@Url.Content(
      "~/Styles/MobileSite.css")" rel="stylesheet" type="text/css" />
    }
</head>
```

This is an extremely basic technique, but it may be adequate if you can adapt your existing markup to fit the small screen purely by adding additional CSS rules to a separate MobileStyles.css file. Of course, you can use the same mechanism elsewhere in your master pages and views to modify output by browser type.

This technique works best if you're building an entirely new Web site and can design its markup so that it suits both desktop and mobile screens depending only on the CSS used. In that case, the additional development effort required is very low. For many sites this simple technique won't be sufficient, but there are two alternatives: switching the master page or presenting different content.

Switching Master Pages

You may be able to keep your existing content pages unchanged, and merely adapt the layout for the small screen using a different master page or layout. For example, if you're building a Web Forms application, you could define a standard page base class that switches its master page dynamically:

```
public class PageBase : Page
{
  protected override void OnPreInit(EventArgs e)
  {
    if (Request.Browser.IsMobileDevice)
        MasterPageFile = "~/Mobile.Master";
  }
}
```

Then, for any page whose layout should vary by device type, set its codebehind class to inherit from PageBase instead of the usual System.Web.UI.Page. You can then create a master page at /Mobile.Master whose layout and CSS styling are optimized for mobile devicess.

It's even easier for ASP.NET MVC 3 developers using Razor layouts—you can make all of your views switch layouts dynamically by editing your /Views/Shared/_Layout.cshtml file to contain the following:

```
@{
  Layout = Request.Browser.IsMobileDevice
             ? "~/Views/Shared/_MobileLayout.cshtml"
             : "~/Views/Shared/_Layout.cshtml";
}
```

Then, add a new Razor layout file at /Views/Shared/_MobileLayout.cshtml, and modify its structure and CSS styles to suit the mobile device as you wish.

This gives more flexibility than the previous technique of varying CSS and occasional markup segments alone, but still has the limitation that both desktop and mobile pages must show essentially the same information and use the same interaction mechanisms.

Presenting Different Content

For some applications, you won't be able to adapt your desktop pages to suit mobile devices merely using different CSS or master pages and layouts because:

- **Your business requirements might be too demanding.** If you want a truly slick mobile experience, you may need to display different (perhaps less) information to mobile devices, and possibly guide the user through different workflows. For example, your user-registration process may have fewer steps and collect less information for mobile visitors. This is more than a matter of CSS.
- **You may be working with legacy code that's not amenable to such change.** For example, your existing markup may contain hardcoded element sizes and styles. Modifying this using CSS or a different master page might be impossible, or might just make things more complicated and less maintainable.

In either case, the ultimate solution is to use entirely separate logic and markup for different device types. The drawback is that you then have two versions to maintain, but the key benefit is that the behavior of the two can vary independently in any way you want. For Web Forms developers, the implementation is usually a set of mobile-specific ASPX pages, and for MVC developers, it usually means creating a new area for mobile-specific controllers and views. Either way, you'll need some logic to redirect incoming visitors to the correct page depending on their device type.

For code samples showing ways to implement redirection logic in a way that's compatible with both output caching and forms authentication on both Web Forms and MVC, see the white paper at bit.ly/gHT3Ap.

Conclusions and Final Recommendations

In this article, you've learned about:

- Why mobile browsers are increasingly important.
- Why great mobile support is primarily a matter of UX design, not just different markup.
- How the core ASP.NET platform detects mobile browsers by default.
- How the default browser detection approach is limited, and how you can extend or replace it.
- How mobile browsers display desktop-sized pages on small screens, and how you can influence that.
- Architectural options for sending different output to different browser types.

As you select a combination of techniques to best suit your application and end users, my top recommendations are *prioritize usability* and *test it*. There's no point implementing mobile support if it ends up giving users a worse browsing experience! Here are some things to consider:

Mobile Web Access

Global Internet-access statistics from International Telecom Union
itu.int/ITU-D/ict/statistics/at_glance/KeyTelecom.html

World population 6.9 billion according to U.S. Census Bureau
census.gov/ipc/www/popclockworld.html

Morgan Stanley Internet Trends Report, December 2010
morganstanley.com/institutional/techresearch/pdfs/Internet_Trends_041210.pdf

OnDevice Research
slideshare.net/OnDevice/the-mobile-only-internet-generation

Offer mobile users a way of switching back to the regular desktop view. Typically, this means placing a link at the top of your pages saying "Switch to desktop view." The way the switch is implemented depends on your architecture; it may simply link to the desktop version of a given URL, or it may set a cookie that overrides your normal browser-detection mechanism.

This facility is especially important if your mobile pages show less information than your desktop pages, because power users will be frustrated if they can't access information or features that they know should be there.

> My top recommendations are prioritize usability and test it.

Don't lose information when redirecting to a mobile view. On some Web sites, incoming mobile visitors are redirected to the mobile homepage, no matter what page they were requesting. This is enormously frustrating for users, and essentially breaks almost every incoming link. If you don't have a mobile version of the page being requested, just show the desktop version of it.

Validate your implementation using actual devices or emulators. Your mobile-friendly layouts, CSS and meta tags may not be handled as you expect by all devices. You must test on actual devices or emulators. For a list of emulators for popular mobile devices, see asp.net/mobile/device-simulators.

It's OK to start small. You don't have to create a mobile-optimized version of every page and feature on your whole site at once. For many businesses, the majority of the value will come from having a mobile-enabled homepage, and perhaps a few other key user workflows such as registration and catalog browsing.

For some intranet applications, it may never be relevant to support mobile devices. But for any public Internet site, you'll almost certainly need to consider mobile browsers if you are to remain relevant in the coming years. ■

STEVEN SANDERSON *works for Microsoft as a program manager on the team that brings you ASP.NET MVC, Web Forms, NuGet and other Web-related goodness.*

THANKS to the following technical expert for reviewing this article:
Scott Hunter

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

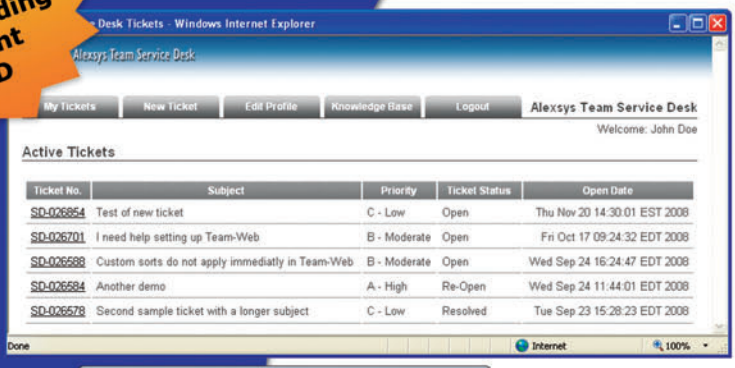
Service Desk Features

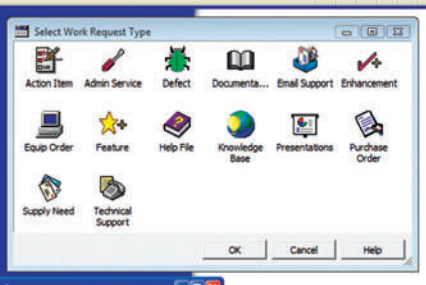
- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

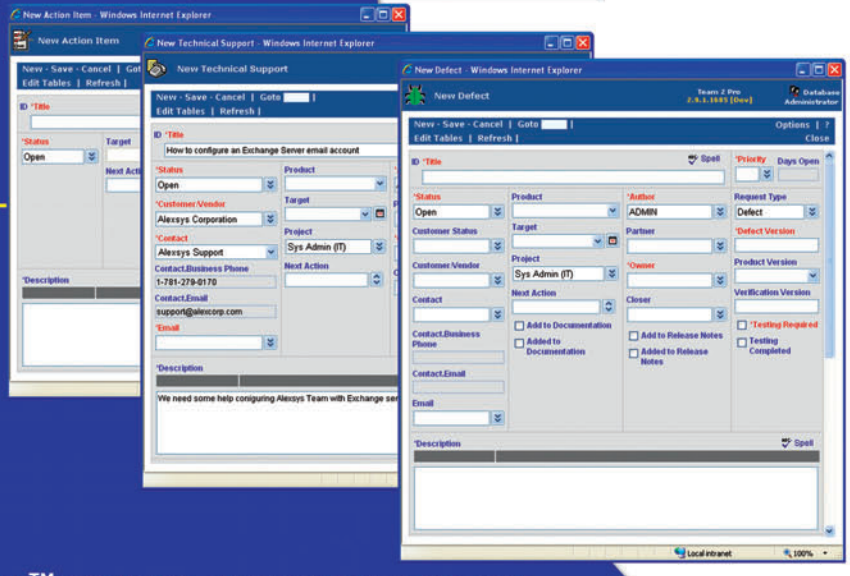
Team 2 Features

- 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.

ARTHUR BIERER AND BORIS FELDMAN

Make Money with Microsoft Ad Control

While some developers create mobile applications as a hobby or as a way to see their names in lights, for others it's all about the money. As you may already know, the Windows Phone Marketplace offers a simple way to get paid for your hard work by selling apps to users. However, there's also another, complementary way for you to make money from your Windows Phone 7 applications: advertising. The Microsoft Advertising SDK makes the process of incorporating ads into your apps easy—and you'll get paid when consumers use your app, even if it's free to download.

This article will:

- Show you how to get started with Microsoft Advertising in your application.
- Walk you through creating an advertising-supported app using XAML or in code—going from File | New all the way to Build and Run.
- Go beyond the basics to help you improve the advertising experience for your end users and make more money.

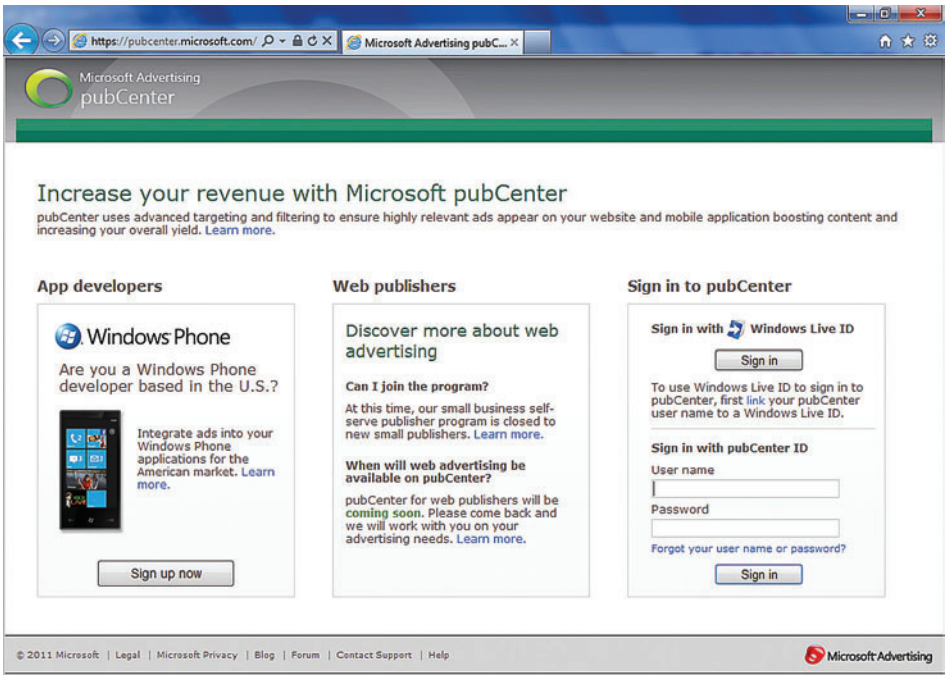


Figure 1 **The Microsoft Advertising pubCenter Portal**

About Advertising

The reality is that mobile users are, for the lack of a better word, "thrifty." Many who won't hesitate for a second to spend $4 on a double latte will agonize over a 99-cent app purchase. Advertising enables you to still make money from those users who aren't ready or willing to spend their hard-earned pennies to buy your app. Free apps, "lite" versions of paid apps and trials of paid apps all present ad monetization opportunities. Given that downloads of free and trial apps greatly outpace those of paid apps in the Windows Phone Marketplace, showing ads in your app is simply good math.

The Microsoft Advertising SDK is a free download and the service is free to use. Furthermore, the ads shown when you use the Microsoft Ad Control aren't just from Microsoft adCenter. Instead, Microsoft has created a real-time ad exchange where top-tier mobile ad networks bid for the right to show their ads in your app. This improves ad rates overall and ensures that advertisers are paying top dollar for ads in mobile apps on Windows Phone 7. You get paid 70 percent of the revenue the ad impressions in your app generate monthly.

As you'll see later in this article, using the Microsoft Ad Control is easy enough that even part-time developers and students are getting into the act of building ad-supported Windows Phone 7 apps. Your mileage may vary, of course; if your app engages end users and they use it often, the money-making potential goes far beyond pocket change. Many developers have made more from advertising than they have through app sales, and Microsoft has published a couple success stories that highlight a few of these experiences (bit.ly/9RIMFW).

Code download available at code.msdn.microsoft.com/mag201107MobileMatters.

Getting Started

Getting started with the Microsoft Ad Control is easy and, even if you're still a novice Windows Phone 7 developer, you can be up and running in no time. Here are the steps to include Microsoft Advertising in your application:

1. **Sign up on the Microsoft pubCenter portal with your Windows Live ID.** Here you'll be able to register your ad-supported applications and the locations within those apps where ads will appear.
2. **Get the free Microsoft Advertising SDK for Windows Phone 7.** Today, you can download this SDK from the pubCenter portal, but later this year you'll get it automatically when you install the upcoming version of the Windows Phone Developer Tools.
3. **Add the Microsoft Ad Control to your application.** Set a couple of properties in the Ad Control to identify your app and ad location, and you're ready to submit your ad-enabled app to the Windows Phone Marketplace.

Once your app is live in the Windows Phone Marketplace and starts getting used by end users, it will automatically start generating ad impressions. You'll get paid for showing these ads on a CPM-basis (cost per thousand views; see "Advertising 101" for more on the cost breakdown) and after you've hit a minimum threshold of $50, you'll automatically get issued a check or direct deposit—your choice—from Microsoft.

pubCenter Setup

Your first step in working with ads from Microsoft is to sign up at the Microsoft Advertising pubCenter portal at pubcenter.microsoft.com using your Windows Live ID (see **Figure 1**).

You'll be asked to enter some information about yourself or your company, then you can jump right in and register your first application and "ad unit."

In order to define a new application in pubCenter, all you have to do is give it a friendly name. Later on, if you have more than one ad-supported application, you'll be able to break out the statistics to see how the ads in each of your applications are performing.

On the same screen, you can also define your first ad unit. Each ad unit has a name and represents a location in your app where advertising will be shown. Ad units can be associated with up to three categories that best describe the content surrounding them (see **Figure 2**).

The categories you select will be one of the many factors that determine which ads are displayed in your app.

Once you click Save, you'll see a summary of the information for your new pubCenter account and also the IDs for the application and ad unit you just created. When you're ready to go live, you'll use these values with the Ad Control to identify your app to the Microsoft Advertising servers.

> You'll get paid when consumers use your app, even if it's free to download.

At this point, you're done with pubCenter for the time being and are ready to dive into Visual Studio. You can return to pubCenter

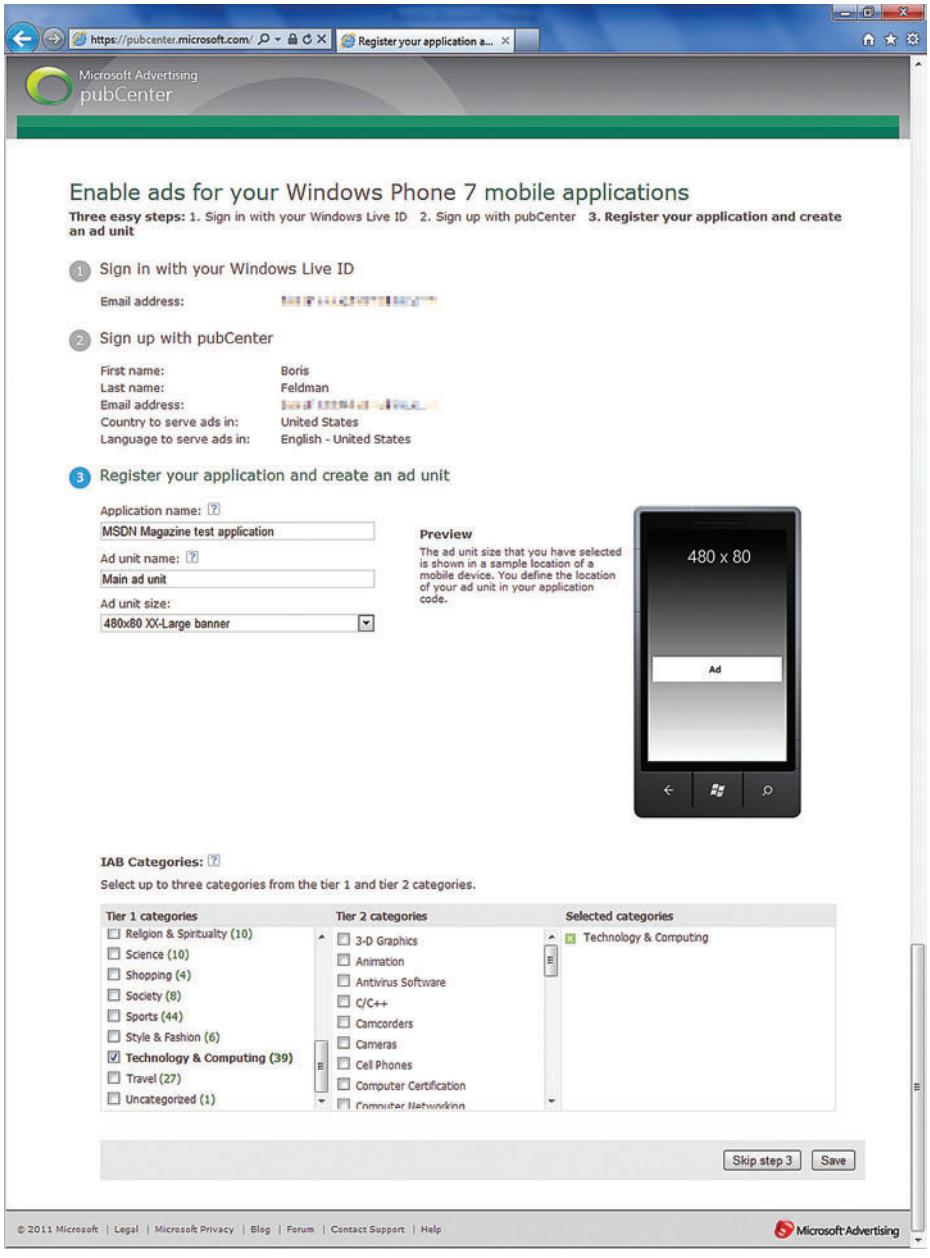


Figure 2 **Registering Your App and Creating an Ad Unit**

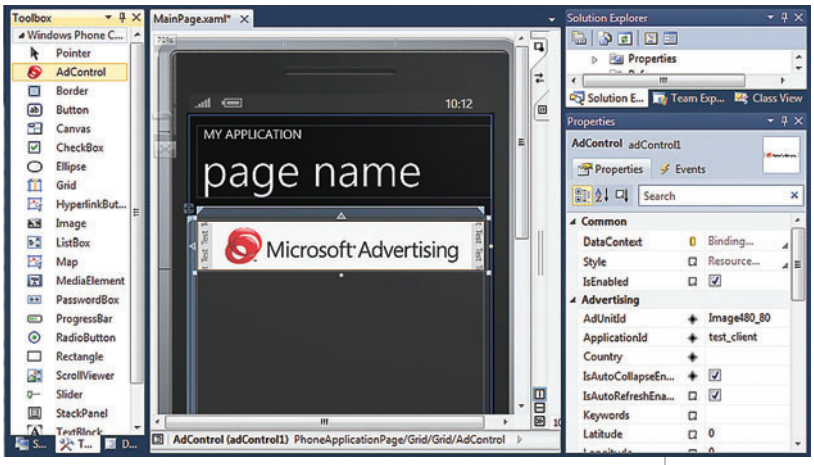


Figure 3 **Placing the AdControl**

at any time to define more applications or ad units, see in-depth advertising reports and configure advanced settings such as competitive exclusions.

Adding Ads with Visual Studio

If you're not using the latest Windows Phone Developer Tools yet, you can download the newest version of the Microsoft Advertising SDK from pubCenter. This SDK includes the Microsoft Ad Control as well as documentation and code samples to help you get started. The June 2011 version of the Ad Control offers rich media support and an improved API set. Make sure you have it installed so that you can follow along. Let's look at how to use it from the Visual Studio Designer.

Start by creating a new Windows Phone 7 Silverlight project with File | New, and opening up the XAML surface where you want to show ads. Pick a suitable spot where the ad can be frequently shown, but won't annoy the user; typically, the bottom or top of the screen is best. Drag the AdControl (note: "Ad Control" refers to the Microsoft product, while "AdControl" refers to the class or object) directly from the Toolbox pane to add it directly to a Grid control (see **Figure 3**).

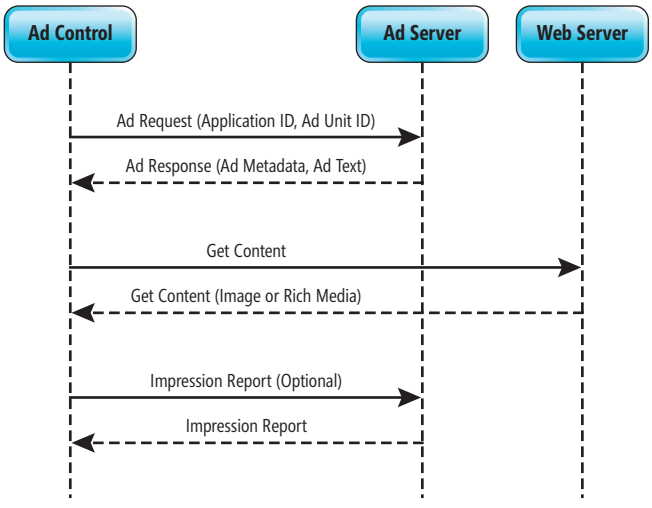


Figure 4 **How the AdControl Works**

Once the AdControl is on the page, you need to set some properties for it to work correctly in your application. First, you'll need to double-check the width and height of the AdControl. It's recommended to keep the control at the standard dimension of 480 x 80 pixels for mobile ads. Setting a smaller size will prevent standard-sized ads from being displayed on your page. Setting a larger size will mean that there will be some empty space between where the ad is rendered and the border of the AdControl.

Next, you'll need to link the AdControl with the Application ID and Ad Unit ID you created in the pubCenter portal. It helps to have the portal Web page open so you can copy and paste these values into the Visual Studio window.

Finally, there are two properties you'll want to set to meet your app's needs. If set to true, the IsAutoRefreshEnabled property automatically shows a new ad every 60 seconds. This enables you to increase the number of ads a user sees, which should lead to better revenue for your app (see "Advertising 101" for more on ad revenue). Set the IsAutoCollapseEnabled property to true if you want the AdControl to automatically hide itself from view before an ad is downloaded.

> Microsoft has created a real-time ad exchange where top-tier mobile ad networks bid for the right to show their ads in your app.

The XAML generated by the Visual Studio Designer should look something like this:

```
<Grid x:Name="ContentPanel" Grid.Row="1">
  <my:AdControl
    Height="80"
    Name="adControl1"
    Width="480"
    IsAutoCollapseEnabled="True"
    IsAutoRefreshEnabled="True"
    AdUnitId="test_client"
    ApplicationId="Image480_80"
  />
</Grid>
```

Now hit Run to see your first ads.

AdControl API

If you're finding that you want to do some fancier things, such as controlling the hiding or showing of ads, or you want to have a bit more control over when the ad shows in your application, you can chose to instantiate the AdControl directly in your code.

But before we get there, let's start by dissecting how the AdControl works (see **Figure 4**). Behind the scenes, when your app starts and the AdControl gets initialized, it requests an ad from the Microsoft Advertising servers. The request is formed with the Application ID and Ad Unit ID that you specified in your code.

Figure 5 **Adding the AdControl to the Visual Tree**

```
using Microsoft.Advertising.Mobile.UI;

// Constructor
public MainPage()
{
  InitializeComponent();

  AdControl adControl = new AdControl();

  // Add the control to a grid control
  Grid grid = (Grid)this.LayoutRoot.Children[1];
  grid.Children.Add(adControl);

  // Insert real values from pubCenter before
  // submitting your app to Windows Phone Marketplace!
#ifdef SHOW_TEST_ADS
  adControl.AppId = "test_client";
  adControl.AdUnitId = "Image480_80";
#else
  // Use your real Application ID and Ad Unit ID here
  adControl.AppId = "12ab456c-de7f-89ab-0123-4567890c1d23";
  adControl.AdUnitId = "12345";
#endif

  // Make the AdControl fit the standard 480 x 80 dimension
  adControl.Width = 480;
  adControl.Height = 80;

  // Let the AdControl collapse and refresh automatically
  adControl.IsAutoRefreshEnabled = true;
  adControl.IsAutoCollapseEnabled = true;
}
```

It's critical that this information matches what you copied from the pubCenter portal or to the test IDs provided in the documentation. The request also contains some other useful information about your app, such as the region to which ads will be matched.

If the ad it receives back uses images or HTML-based rich media, the AdControl will reconnect with a Web server to download the advertising content (such as JPEG, GIF, HTML or PNG files). Once the user has seen the ad, the AdControl may connect again with the server to record the ad impression. This whole cycle repeats when the ad gets refreshed and a new ad gets downloaded.

There isn't much in the way of rocket science when it comes to the code required to use the AdControl.

There isn't much in the way of rocket science when it comes to the code required to use the AdControl. You'll need to start by adding the Microsoft.Advertising.Mobile.UI assembly as a reference in your project. You'll also want to add a *using* statement for the Microsoft.Advertising.Mobile.UI namespace. And in your page's constructor, you'll place the code to create the AdControl. You'll then need to add the AdControl instance into the page's visual tree. In this case, we're adding it to a grid that we have in our layout (see **Figure 5**). With the code in **Figure 5** in place, your app should be ready to build and show ads using the code you've written.

Tips and Tricks

Here are a couple of important things to keep in mind when working with the AdControl in your app, and a few ways—such as targeting—that will help you maximize the value of the advertising in your app.

First, always make sure you can see both "test" and "real" ads running in your application. To view test ads, set the ApplicationID to "test_client" and Ad Unit ID to a standard test value such as "Image480_80." A well-placed #ifdef can come in handy here. You should use test ads while developing and testing your app, but once you're ready to release, it's critical you set this back to your pubCenter-issued Application ID and Ad Unit ID. However, testing with your own app IDs is important because it ensures your application works with "real" live ads.

Next, to get the best ads, it's important to provide as much information as possible to the ad system. The AdControl often gets this information automatically, so your application manifest must have the following capabilities:

- ID_CAP_PHONEDIALER
- ID_CAP_NETWORKING
- ID_CAP_WEBBROWSERCOMPONENT
- ID_CAP_IDENTITY_USER
- ID_CAP_MEDIALIB

If it makes sense for your application, it's also recommended to use the ID_CAP_LOCATION capability. Enabling location-based advertising improves the quality and relevance of the ads that your users will see.

Typically you don't need to use the AdManager class, because it's automatically initialized by the AdControl, but it's useful to know about. The AdManager.Current property allows an app to access a shared advertising state, such as the user's location, for example. In some cases, you may want to set this manually and the

Figure 6 **The AdRefreshed Event**

```
public void AdControl_AdRefreshed(object sender, EventArgs args)
{
  AdControl ad = (AdControl)sender;

  Dispatcher.BeginInvoke(() =>
  {
    ad.Visibility = System.Windows.Visibility.Visible;
    Debug.WriteLine(
      "ad control '" + ad.Name + "' got ad, visibility = " + ad.Visibility);
  });
}

public void AdControl_ErrorOccurred(object sender, AdErrorEventArgs args)
{
  try
  {
    AdControl ad = (AdControl)sender;

    Dispatcher.BeginInvoke(() =>
    {
      ad.Visibility = System.Windows.Visibility.Collapsed;
      Debug.WriteLine(
        "error in ad control '" + ad.Name + "': " + args.Error.Message);
      Debug.WriteLine("ad control '" + ad.Name + "' visibility = " + ad.Visibility);
    });
  }
  catch (Exception e)
  {
    Debug.WriteLine("oh no! " + e.Message);
  }
}
```

Advertising 101

If you're new to online or mobile advertising, here's a quick primer on the terms and concepts you'll need to know.

Types of Ads

- **Search** ads appear in search results such as on Bing or Google. Ad relevance is derived from the keywords in the query the user types in. These are typically text ads and are generally sold as CPC (see "Buying Models").
- **Contextual** ads are similar to Search ads but are based on the content of the Web page they appear on—typically determined via crawler—rather than on an explicit query entered by the user. These are most often text ads but can also be images in some cases. Also typically sold as CPC.
- **Display** ads are focused on reaching an audience and are more about brand awareness. These ads are almost always images or interactive rich media and are mostly sold via CPM (see "Buying Models").

Buying Models

- **CPM**–Cost per thousand (M) impressions. The advertiser pays when a user sees an ad. The focus is on brand awareness.
- **CPC**–Cost per click. The advertiser pays when a user clicks on the ad. The focus is on driving traffic to the advertiser's site.
- **CPA**–Cost per action. The advertiser pays when a user completes a certain action (called a "conversion"). This could be making an online purchase, filling out a form and so on. The focus is on business results.

Other Key Advertising Metrics

- **CTR**–Click-through rate. A measure of how often people click on an ad. This is a key measure of how effective an ad is and is important to advertisers even for CPM and CPA campaigns.
- **eCPM** or **Yield**–eCPM stands for Effective CPM and can be used interchangeably with yield. This measure is derived from revenue divided by the total number of impressions shown. This can be used to compare monetization across buying models.
- **Fill-rate**–The percentage of ad requests that are being fulfilled. Note that this isn't generally 100 percent and that there are important trade-offs between fill-rate versus yield for overall end-user experience.

Ad Targeting

Targeting refers to using signals about the end user to help advertisers reach a particular audience.

The major types of ad targeting include:

- **Intent/Context**–What is the user doing? Examples: search query, application category.
- **Demographic**–Who is the user? Examples: age, gender and income range.
- **Location**–Where is the user? Example: The user is in ZIP code 98004 or at latitude/longitude 47.617255, -122.191877.
- **Behavioral**–What has the user expressed interest in before? Example: Past online experiences or actions.

Additional Resources

Microsoft Advertising (bit.ly/9RIMFW)

- "How do I" videos
- Frequently asked questions
- Monetization best practices

Microsoft Advertising SDK for Windows Phone 7 (bit.ly/aTcRi2)

Publisher Support E-mail (psupport@microsoft.com)

AdManager class is how you do that. In the following example, a Beverly Hills movie star app would generally know that its users are around that location:

```
using Microsoft.Advertising;

// Make sure the AdControl is created
// or the AdManager is initialized before setting
AdManager.Current.Country = "US";
AdManager.Current.PostalCode = "90210";
```

Similarly, it might also be useful to handle a few things about the ad experience in your code. If you've set the IsAutoRefreshEnabled property to false, then you'll be able to refresh ads manually. Call the adControl.Refresh method to bring a new ad into view, but make sure not to call it more than every 30 seconds in order to give users enough time to click on ads they might be interested in.

If you've set the IsAutoCollapseEnabled property to false, you can listen for an ErrorOccurred event. This event will fire whenever an ad fails to download or display. You can use this event to collapse the AdControl and reclaim that space for use by your application's other UI elements. When the AdRefreshed event fires, you'll know a new ad is ready to be seen and you can restore its visibility (see **Figure 6**).

As you've seen, using the AdControl in your application is easy. And if you run into trouble, there are generally only a few things that can go wrong and cause ads not to be shown:

- No network connectivity
- Typo in Application ID or Ad Unit ID
- The user is in a location where ads from Microsoft Advertising aren't available
- The AdControl is set to a dimension that's too small
- Missing capabilities

Finally, if you're writing a game and want to use XNA, look at the Microsoft Advertising XNA advertising game component that's also part of the Microsoft Advertising SDK. Unfortunately, there isn't enough space to go in-depth on XNA in this article, but there is documentation available online at bit.ly/jhGOVV.

Now go add some advertising and start getting paid for all that hard work you put into your application! ■

Arthur Bierer *is a senior program manager on the Universal Ad Client team at Microsoft in New York. Before working on the Microsoft Advertising SDK for Windows Phone 7, he had 17 years of experience with the Dynamics NAV, .NET Framework, Internet Explorer and Windows products. You can follow him on Twitter at twitter.com/ArthurBiererDev or e-mail him at arthurbi@microsoft.com.*

Boris Feldman *is a group product planning manager for the Mobile Display Advertising business at Microsoft. During his eight years at Microsoft, he has also worked on a number of developer products, including the .NET Framework and Expression Web. You can reach him by e-mail at borisf@microsoft.com.*

Thanks to the following technical experts for reviewing this article: *Anzor Balkar, Richard Carr, Sloan Ginn, Ali Heron, Darryl Hudgin, Marty Kauhanen, Mark Masterson, Tim McClelland, Valeriy Ovechkin, Chris Quon, Matt Sullivan and Jason White*

we are Countersoft
you are Control

GEMINI
Project Platform

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

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

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

ATLAS
Product Support Optimized

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

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

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

COUNTERSOFT ENABLING COLLECTIVE CAPABILITY www.countersoft.com

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

TED NEWARD

Multiparadigmatic .NET, Part 9: Functional Programming

Any time an article series gets close to double digits, one of two things is happening: either the author is pretentious enough to think that his readers are actually interested in that subject that many times in a row, or he is just too boneheaded to think to come up with a new topic. Or, I suppose, sometimes the subject just merits that much coverage. No matter which is the case here, rest assured we're in the home stretch now.

In the previous piece in the June issue (msdn.microsoft.com/magazine/hh205754), the idea of providing variability along a name-based axis came under the microscope, using naming conventions and dynamic programming—that is, binding by name, which in .NET typically means reflection at some level—to solve some interesting design problems. Most .NET developers, I imagine, expect that most of the dynamic programming they encounter will be through the "dynamic" keyword that C# 4 provides. As old-hand Visual Basic developers know, however, C# only came by its dynamism recently, whereas Visual Basic programmers have known it—and used it, in many cases quite successfully—for decades.

But that's not the last of the paradigms—one more remains to be explored, and, again, it's one that's been hiding in plain sight for a few years now. And while it's certainly easy (if a tad cheeky) to describe functional design as design along the algorithmic axis of commonality-variability, this simultaneously oversimplifies and obscures its capabilities.

> Functional programming is about treating functions as values, just like any other data value type.

In a single sentence, functional programming is about treating functions as values, just like any other data value type, meaning we can pass functions around just as we can data values, as well as derive new values out of those values. Or, put more accurately, functions should be treated as first-class citizens within the language: they can be created, passed to methods and returned from methods just as other values are. But that explanation, again, doesn't precisely enlighten, so let's begin with a simple case study.

Figure 1 **A Simple Calculator**

```
class Program
{
  static void Main(string[] args)
  {
    if (args.Length < 3)
        throw new Exception("Must have at least three command-line arguments");

    int lhs = Int32.Parse(args[0]);
    string op = args[1];
    int rhs = Int32.Parse(args[2]);
    switch (op)
    {
      case "+": Console.WriteLine("{0}", lhs + rhs); break;
      case "-": Console.WriteLine("{0}", lhs - rhs); break;
      case "*": Console.WriteLine("{0}", lhs * rhs); break;
      case "/": Console.WriteLine("{0}", lhs / rhs); break;
      default:
        throw new Exception(String.Format("Unrecognized operator: {0}", op));
    }
  }
}
```

Imagine the design exercise is to create a small command-line calculator: a user types (or pipes) a mathematical expression in, and the calculator parses it and prints the result. Designing this is pretty straightforward, as shown in **Figure 1**.

As written, it works—until the calculator receives something other than the cardinal four operators. What's worse, though, is that a significant amount of code (compared to the overall size of the program) is duplicate code, and will continue to be duplicate code as we add new mathematical operations to the system (such as the modulo operator, %, or the exponent operator, ^).

Stepping back for a moment, it's clear that the actual operation—what's being done to the two numbers—is what varies here, and it would be nice to be able to rewrite this in a more generic format, as shown in **Figure 2**.

Obviously, we could simply recreate the switch/case block in Operate, but that doesn't really gain much. Ideally, we'd like some kind of string-to-operation lookup (which, on the surface, is a form of dynamic programming again, binding the name "+" to an additive operation, for example).

Within the design patterns world, this would be a case for the Strategy pattern, where concrete subclasses implement a base class or interface, providing the necessary signature and compile-time typechecking for safety, something along the lines of:

```
interface ICalcOp
{
  int Execute(int lhs, int rhs);
}
class AddOp : ICalcOp { int Execute(int lhs, int rhs) { return lhs + rhs; } }
```

Figure 2 **A More Generic Calculator**

```
class Program
  {
    static void Main(string[] args)
    {
      if (args.Length < 3)
          throw new Exception("Must have at least three command-line arguments");

      int lhs = Int32.Parse(args[0]);
      string op = args[1];
      int rhs = Int32.Parse(args[2]);
      Console.WriteLine("{0}", Operate(lhs, op, rhs));
    }
    static int Operate(int lhs, string op, int rhs)
    {
      // ...
    }
  }
```

Which works … sort of. It's pretty verbose, requiring a new class to be created for each operation we want. It's also not very object-oriented, because we really only need one instance of it, ever, hosted inside of a lookup table for matching and execution:

```
private static Dictionary<string, ICalcOp> Operations;
static int Operate(int lhs, string op, int rhs)
{
  ICalcOp oper = Operations[op];
  return oper.Execute(lhs, rhs);
}
```

It somehow feels like this could be simplified; and, as some readers have probably already realized, this is a problem that has already been solved once before, except in the context of event-handler callbacks. This is exactly what the delegate construct was created for in C#:

```
delegate int CalcOp(int lhs, int rhs);
static Dictionary<string, CalcOp> Operations =
  new Dictionary<string, CalcOp>();
static int Operate(int lhs, string op, int rhs)
{
  CalcOp oper = Operations[op];
  return oper(lhs, rhs);
}
```

> Passing functions around is not something we're used to in mainstream .NET development.

And, of course, Operations has to be initialized properly with the operations that the calculator recognizes, but adding new ones becomes a bit easier:

```
static Program()
{
  Operations["+"] = delegate(int lhs, int rhs) { return lhs + rhs; }
}
```

Savvy C# 3 programmers will immediately recognize that this can be shortened even further, using *lambda expressions*, which were introduced in that version of the language. Visual Basic can, in Visual Studio 2010, do something similar:

```
static Program()
{
  Operations["+"] = (int lhs, int rhs) => lhs + rhs;
}
```

This is where most C# and Visual Basic developers' ideas about delegates and lambdas stop. But lambdas and delegates are far more interesting, particularly when we start extending the idea even further. And this idea, of passing functions around and using them in various ways, goes deeper.

Reduces, Maps and Folds—Oh My!

Passing functions around is not something we're used to in mainstream .NET development, so a more concrete example of how this can benefit design is necessary.

Assume for a moment that we have a collection of Person objects, as shown in **Figure 3**.

Now, it so happens that Management wants to celebrate something (perhaps they all made quota). What they want to do is give each of these people a beer, which is pretty easily accomplished using the traditional foreach loop, as shown in **Figure 4**.

> The biggest problem with this code? It's intrinsically un-reusable.

There are some minor bugs here (mostly in that your code is handing a beer to my 11-year-old son), but the biggest problem with this code? It's intrinsically un-reusable. Attempts to later give everybody another beer require another foreach loop, which violates the Don't Repeat Yourself (DRY) principle. We could, of course, gather the beer-issuing code into a method (classic procedural commonality response), like so:

```
static void GiveBeer(List<Person> people)
{
  foreach (var p in people)
    if (p.Age >= 21)
        Console.WriteLine("Have a beer, {0}!", p.FirstName);
}
```

(Notice that I added the over-21 age check; my wife, Charlotte, insisted I include it before this article could go to publication.) But what if the desire is to find everybody who is over the age of 16 and give them a free R-rated movie ticket, instead? Or to find everybody who is over the age of 39 and give them a "Holy Cow You're

Figure 3 **A Collection of Person Objects**

```
class Person
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public int Age { get; set; }
}

class Program
{
  static void Main(string[] args)
  {
    List<Person> people = new List<Person>()
    {
      new Person() { FirstName = "Ted", LastName = "Neward", Age = 40 },
      new Person() { FirstName = "Charlotte", LastName = "Neward", Age = 39 },
      new Person() { FirstName = "Michael", LastName = "Neward", Age = 17 },
      new Person() { FirstName = "Matthew", LastName = "Neward", Age = 11 },
      new Person() { FirstName = "Neal", LastName = "Ford", Age = 43 },
      new Person() { FirstName = "Candy", LastName = "Ford", Age = 39 }
    };
  }
}
```

Figure 4 **The Traditional foreach Loop**

```
static void Main(string[] args)
{
  List<Person> people = new List<Person>()
  {
    new Person() { FirstName = "Ted", LastName = "Neward", Age = 40 },
    new Person() { FirstName = "Charlotte", LastName = "Neward", Age = 39 },
    new Person() { FirstName = "Michael", LastName = "Neward", Age = 17 },
    new Person() { FirstName = "Matthew", LastName = "Neward", Age = 11 },
    new Person() { FirstName = "Neal", LastName = "Ford", Age = 43 },
    new Person() { FirstName = "Candy", LastName = "Ford", Age = 39 }
  };
  foreach (var p in people)
    Console.WriteLine("Have a beer, {0}!", p.FirstName);
}
```

Old!" balloon? Or find everybody over the age of 65 and give each of them a small notebook to write things down that they're likely to forget (like their name, age, address …)? Or find everybody with the last name other than "Ford" and invite them to a Halloween party?

> Given the power of delegates, we can create commonality while still exposing the necessary variability.

The more of these examples we toss off, the more it becomes clear that each of these cases presents two elements of variability: filtering the Person objects, and the action to take with each of those Person objects. Given the power of delegates (and the Action<T> and Predicate<T> types introduced in .NET 2.0), we can create commonality while still exposing the necessary variability, as shown in **Figure 5**.

One more common operation is to "transform" (or, to be more accurate about it, "project") an object into another type, such as when we want to extract the last names from the list of Person objects into a list of strings (see **Figure 6**).

Notice that, thanks to the generics use in the declarations of Filter, Execute and Transform (more commonality/variability!), we can reuse Execute to display each of the found last names.

Figure 5 **Filtering Person Objects**

```
static List<T> Filter<T>(List<T> src, Predicate<T> criteria)
{
  List<T> results = new List<T>();
  foreach (var it in src)
    if (criteria(it))
      results.Add(it);
  return results;
}
static void Execute<T>(List<T> src, Action<T> action)
{
  foreach (var it in src)
    action(it);
}
static void GiveBeer(List<Person> people)
{
  var drinkers = Filter(people, (Person p) => p.Age >= 21);
  Execute(drinkers,
      (Person p) => Console.WriteLine("Have a beer, {0}!", p.FirstName));
}
```

Notice, too, how the use of the lambda expressions make an interesting implication begin to come clear—one that becomes even more obvious when we write yet another common functional operation, reduce, which "collapses" a collection down into a single value by combining all values together in a specified way. For example, we could add up everybody's age to retrieve a sum-of-all-ages value using a foreach loop, like so:

```
int seed = 0;
foreach (var p in people)
  seed = seed + p.Age;
Console.WriteLine("Total sum of everybody's ages is {0}", seed);
```

Or we can write it using a generic reduce, as shown in **Figure 7**.

This reduction operation is often referred to as a "fold," by the way. (To the discerning functional programmer, the two terms are slightly different, but the difference isn't critical to the main discussion.) And, yes, if you'd begun to suspect that these operations were really nothing more than what LINQ provides for objects (the LINQ-to-Objects feature that got so little love when it was originally released), you'd be spot-on (see **Figure 8**).

To the working enterprise .NET developer, this seems silly. It's not like *real* programmers spend time looking for ways to reuse age-summation code. Real programmers write code that iterates over a collection of objects, concatenating each one's first name into an XML representation inside of a string, suitable for use in an OData request or something:

```
Console.WriteLine("XML: {0}", people.Aggregate("<people>",
  (string current, Person p) =>
    current + "<person>" + p.FirstName + "</person>")
  + "</people>");
```

Whoops. Guess that LINQ-to-Object stuff might be useful after all.

> If you're a classically trained object-oriented developer, this seems ridiculous yet elegant at the same time.

More Functional?

If you're a classically trained object-oriented developer, this seems ridiculous yet elegant at the same time. It can be a mind-blowing

Figure 6 **Converting from a List of Objects to a List of Strings**

```
public delegate T2 TransformProc<T1,T2>(T1 obj);
static List<T2> Transform<T1, T2>(List<T1> src,
  TransformProc<T1, T2> transformer)
{
  List<T2> results = new List<T2>();
  foreach (var it in src)
    results.Add(transformer(it));
  return results;
}
static void Main(string[] args)
{
  List<Person> people = // ...

  List<string> lastnames = Transform(people, (Person p) => p.LastName);
  Execute(lastnames, (s) => Console.WriteLine("Hey, we found a {0}!", s);
}
```

Figure 7 **Using a Generic Reduce**

```
public delegate T2 Reducer<T1,T2>(T2 accumulator, T1 obj);
static T2 Reduce<T1,T2>(T2 seed, List<T1> src, Reducer<T1,T2> reducer)
{
  foreach (var it in src)
    seed = reducer(seed, it);
  return seed;
}
static void Main(string[] args)
{
  List<Person> people = // ...

  Console.WriteLine("Total sum of everybody's ages is {0}",
    Reduce(0, people, (int current, Person p) => current + p.Age));
}
```

moment, because this approach quite literally comes at software design in a nearly diametrically opposite way from objects: rather than focusing on the "things" in the system, and making behavior something that is attached to each of those things, functional programming looks to identify the "verbs" in the system and see how they can operate on different types of data. Neither approach is more right than the other—each captures commonality and offers variability along very different axes, and as might be imagined, there are places where each is elegant and simple, and where each can be ugly and clumsy.

> This approach quite literally comes at software design in a nearly diametrically opposite way from objects.

Remember, in classic object orientation, variability comes at a structural level, offering the ability to create positive variability by adding fields and methods or replacing existing methods (via override), but nothing about capturing ad hoc algorithmic behavior. In fact, it wasn't until .NET got anonymous methods that this axis of commonality/variability became feasible. It was possible to do something like this in C# 1.0, but each lambda had to be a named method declared somewhere, and each method had to be typed in System.Object terms (which meant downcasting inside those methods), because C# 1.0 didn't have parameterized types.

Figure 8 **Fold Operations**

```
static void Main(string[] args)
{
  List<Person> people = new List<Person>()
  {
    new Person() { FirstName = "Ted", LastName = "Neward", Age = 40 },
    new Person() { FirstName = "Charlotte", LastName = "Neward", Age = 39 },
    new Person() { FirstName = "Michael", LastName = "Neward", Age = 17 },
    new Person() { FirstName = "Matthew", LastName = "Neward", Age = 11 },
    new Person() { FirstName = "Neal", LastName = "Ford", Age = 43 },
    new Person() { FirstName = "Candy", LastName = "Ford", Age = 39 }
  };
  // Filter and hand out beer:
  foreach (var p in people.Where((Person p) => p.Age >= 21))
    Console.WriteLine("Have a beer, {0}!", p.FirstName);

  // Print out each last name:
  foreach (var s in people.Select((Person p) => p.LastName))
    Console.WriteLine("Hey, we found a {0}!", s);

  // Get the sum of ages:
  Console.WriteLine("Total sum of everybody's ages is {0}",
    people.Aggregate(0, (int current, Person p) => current + p.Age));
}
```

> There are numerous other things a functional language can do besides just pass functions around as values.

Longtime practitioners of functional languages will cringe at the fact that I end the article here, because there are numerous other things a functional language can do besides just pass functions around as values—partial application of functions are a huge concept that make much of functional programming incredibly tight and elegant, in languages that support it directly—but editorial needs must be satisfied, and I'm pushing my length limitations as it is. Even so, seeing just this much of the functional approach (and armed with the functional capabilities already present in LINQ) can offer some powerful new design insight.

More importantly, developers interested in seeing more of this should take a long, hard look at F#, which, of all the .NET languages, is the only one that captures these functional concepts (partial application and currying) as first-class citizens within the language. C# and Visual Basic developers can do similar things, but require some library assistance (new types and methods to do what F# does naturally). Fortunately, several such efforts are underway, including the "Functional C#" library, available on CodePlex (functionalcsharp.codeplex.com).

Various Paradigms

Like it or not, multiparadigm languages are in common use, and they look like they're here to stay. The Visual Studio 2010 languages each exhibit some degree of each of the various paradigms; C++, for example, has some parametric metaprogramming facilities that aren't possible in managed code, owing to the way that the C++ compiler operates, and just recently (in the latest C++0x standard) gained lambda expressions. Even the much-maligned ECMAScript/JavaScript/JScript language can do objects, procedures, metaprogramming, dynamic and functional paradigms; in fact, much of JQuery is built on these ideas.

Happy coding! ■

TED NEWARD *is a principal with Neward & Associates, an independent firm specializing in enterprise .NET Framework and Java platform systems. He has written more than 100 articles, is a C# MVP and INETA speaker, and has authored or coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He consults and mentors regularly—reach him at ted@tedneward.com if you're interested in having him come work with your team, or read his blog at blogs.tedneward.com.*

THANKS to the following technical expert for reviewing this article: *Matthew Podwysocki*

AGENDA

OCTOBER 17–21
REDMOND, WASHINGTON

VISUAL STUDIO LIVE! REDMOND

Silverlight/WPF	Developing Services	Visual Studio 2010/.NET 4	Cloud	Data Management	LightSwitch	Programming Practices	Web/HTML5	Mobile Dev

Visual Studio Live! Pre-Conference Workshops: Monday, October 17, 2011

MWK1 Workshop: ALM in 2011: Visual Studio 2010 and the Next Big Release *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, October 18, 2011

T1 Microsoft Session—Details TBA	**T2** Silverlight Intense Intro *Billy Hollis*	**T3** AppFabric, Workflow and WCF—The Next Generation Middleware *Ron Jacobs*	**T4** If not IaaS, When Should I Use Azure VM Role? *Eric Boyd*	**T5** Best Kept Secrets in Visual Studio 2010 and .NET 4.0 *Deborah Kurata*
T6 Microsoft Session—Details TBA	**T7** XAML: Achieving Your Moment Of Clarity *Miguel Castro*	**T8** What's New in WCF 4 *Ido Flatow*	**T9** What is Microsoft Marketplace DataMarket? *Michael Stiefel*	**T10** The LINQ Programming Model *Marcel de Vries*
T11 Microsoft Session—Details TBA	**T12** Fundamental Design Principles for UI Developers *Billy Hollis*	**T13** Creating Scalable State Full Services Using WCF and WF *Marcel de Vries*	**T14** Deciding Between Relational Databases and Tables in the Cloud *Michael Stiefel*	**T15** NoSQL—Beyond the Key-Value Store *Robert Green*
T16 HTML5 and Internet Explorer 9: Developer Overview *Ben Hoelting*	**T17** Bind Anything to Anything in XAML *Rockford Lhotka*	**T18** AppFabric Caching: How It Works and When You Should Use It *Jon Flanders*	**T19** Microsoft Session—Details TBA	**T20** How to Take WCF Data Services to the Next level *Rob Daigneau*

Microsoft Ask the Experts & Welcome Reception

Visual Studio Live! Day 2: Wednesday, October 19, 2011

W1 HTML 5 and Your Web Sites *Robert Boedigheimer*	**W2** Microsoft Session—Details TBA	**W3** Building Native Mobile Apps with HTML5 & jQuery *Jon Flanders*	**W4** Azure Platform Overview *Vishwas Lele*	**W5** ALM *Brian Randell*
W6 Styling Web Pages with CSS 3 *Robert Boedigheimer*	**W7** What's new and cool in Silverlight 5 *Pete Brown*	**W8** Getting Started with Windows Phone 7 *Scott Golightly*	**W9** Building Azure Applications *Vishwas Lele*	**W10** Visual Studio *Brian Randell*
W11 The Best of jQuery *Robert Boedigheimer*	**W12** Silverlight, WCF RIA Services, and Your Business Objects *Deborah Kurata*	**W13** Windows Azure and Windows Phone—Creating Great Apps *Scott Golightly*	**W14** Building Compute-Intensive Apps in Azure *Vishwas Lele*	**W15** Details TBA
W16 SP.NET MVC, Razor, and jQuery—The New Face of ASP.NET *Ido Flatow*	**W17** Light up on Windows 7 with Silverlight and WPF *Pete Brown*	**W18** Handling Offline data in Silverlight and Windows Phone 7 *John Papa*	**W19** Building and Running the Windows Azure Developer Portal *Chris Mullins*	**W20** Microsoft Session—Details TBA

Sponsor Reception / Wild Wednesday

Visual Studio Live! Day 3: Thursday, October 20, 2011

TH1 WebMatrix and Razor *Rachel Appel*	**TH2** Bringing the Silverlight PivotViewer to your Applications *Tony Champion*	**TH3** CSLA 4 for WP7, Android, and iOS *Rockford Lhotka*	**TH4** Microsoft Session—Details TBA	**TH5** Design for Testability: Mocks, Stubs, Refactoring, and User Interfaces *Ben Day*
TH6 Orchard *Rachel Appel*	**TH7** MVVM in Practice aka "Code Behind"—Free WPF *Tiberiu Covaci*	**TH8** Working with Data on Windows Phone 7 *Sergey Barskiy*	**TH9** Microsoft Session—Details TBA	**TH10** Team Foundation Server 2010 Builds: Understand, Configure, and Customize *Ben Day*
TH11 Busy Developer's Guide to (ECMA/Java)Script *Ted Neward*	**TH12** Radically Advanced Templates for WPF and Silverlight *Billy Hollis*	**TH13** Advanced patterns with MVVM in Silverlight and WP7 *John Papa*	**TH14** Microsoft Session—Details TBA	**TH15** LightSwitch 202 *Andrew Brust*
TH16 Getting Started with ASP.NET MVC *Philip Japikse*	**TH17** Using MEF to Develop Composable Applications *Ben Hoelting*	**TH18** WP7 Instrumentation—How to Learn from your App *Tony Champion*	**TH19** So Many Choices, So Little Time: Understanding Your .NET 4.0 Data Access Options *Lenni Lobel*	**TH20** Static Analysis in .NET *Jason Bock*
TH21 Test Driving ASP.NET MVC *Philip Japikse*	**TH22** Patterns for Parallel Programming *Tiberiu Covaci*	**TH23** Microsoft Session—Details TBA	**TH24** Using Code First (Code Only) approach with Entity Framework *Sergey Barskiy*	**TH25** Modern .NET Development Practices and Principles *Jason Bock*

Visual Studio Live! Post-Conference Workshops: Friday, October 21, 2011

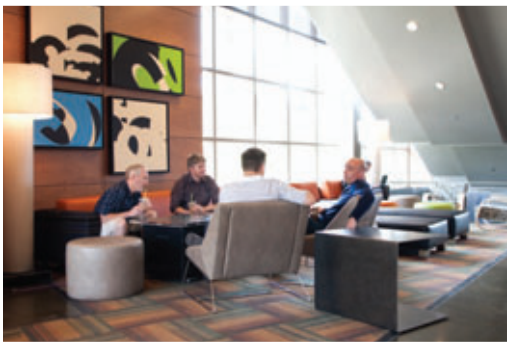
FWK1 Architectural Katas Workshop *Ted Neward*	**FWK2** SQL Server Workshop for Developers *Andrew Brust & Leonard Lobel*

VISIT US ONLINE AT VSLIVE.COM/REDMOND FOR MORE DETAILS!

Sessions and speakers are subject to change.

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS



5 DAYS OF TRAINING @ MICROSOFT HEADQUARTERS!

PICK YOUR SESSIONS NOW!

IF YOU'RE LOOKING FOR HARD-HITTING .NET DEVELOPER TRAINING, look no further than Visual Studio Live! Redmond. Join industry pros and Microsoft insiders for relevant, practical, and up-to-date education that will solve your everyday development challenges.

Plus, experience life on Campus like a blue badge! Get employee-level access to the company store, lunch in Microsoft's "town center," and more.

SUPPORTED BY:

Microsoft

msdn

Microsoft Visual Studio

Visual Studio MAGAZINE

REGISTER NOW TO SAVE $300

Super Early Bird deadline ends August 10.

Use Priority Code: JULAD

PLATINUM SPONSORS

esri VERSANT

GOLD SPONSOR

DevExpress Download • Compare • Decide!

MEDIA SPONSOR

THE CODE PROJECT WWW.CODEPROJECT.COM

PRODUCED BY:

1105 MEDIA

WWW.VSLIVE.COM/REDMOND

Page Transitions and the Rule of Three

The Rule of Three exists in several disparate disciplines. In comedy, for example, the Rule of Three dictates that a joke begin by engaging the audience's interest, then raise anticipation and, finally, be somewhat less funny than anticipated.

In economics, the Rule of Three relates to the existence of three major competitors in a particular market. In programming, it's more like a Three-Strike Rule: Whenever a chunk of similar code appears three times, it should be refactored into a common loop or method. (I originally heard this rule as "Three or more? Use a for." Or perhaps I made up that jingle myself.)

This brings us to the subject of text layout. Sometimes programmers need to display a document that's too long to fit on a single screen. Perhaps the least desirable approach involves shrinking the font size until the document fits. A more traditional solution is a scrollbar. But in recent years—and particularly on smaller devices such as tablets and phones—there's been a trend toward paginating the document and letting the user flip through the pages as if reading a real printed book or magazine.

Displaying a paginated document also involves a Rule of Three: For the most fluid page transitions, the UI needs to support three distinct pages—the current page, the next page and the previous page. As the user pages forward, the current page becomes the previous page, and the next page becomes the current page. Going backward, the current page becomes the next page, and the previous page becomes the current page. In this article, I'll describe how to implement this technique in a manner flexible enough for three different page transitions—a page slide, a 3D-like flip and a 2D page curl.

Back to Gutenberg

In the previous installment of this column (msdn.microsoft.com/magazine/tktktktk), I demonstrated some fairly simple pagination logic in a Windows Phone 7 program named EmmaReader, so called because it lets you read Jane Austen's novel "Emma" (1815). The program uses a plain-text file downloaded from the famous Project Gutenberg Web site (at gutenberg.org), which makes available more than 30,000 public-domain books.

Pagination is a non-trivial process that can require an appreciable amount of time. For that reason, EmmaReader only paginates on demand as the user progresses through the book. After creating a page, the program stores information indicating where in the book that page begins and ends. It can then use this information to let the user page backward through the book.

For the code in this article, I chose the George Eliot novel, "Middlemarch" (1874), and called the downloadable Windows Phone 7 project MiddlemarchReader. This program represents another step closer to a generalized e-book reader for Windows Phone 7 based on Project Gutenberg plain-text files. You won't be able to use this e-book reader for current bestsellers, but it will certainly let you explore the classics of English literature.

The Project Gutenberg plain-text files divide each paragraph into multiple consecutive 72-character lines. Paragraphs are separated with a blank line. This means that any program that wishes to format a Project Gutenberg file for pagination and presentation needs to concatenate those separate lines into single-line paragraphs. In MiddlemarchReader, this occurs in the GenerateParagraphs method of the PageProvider class. This method is called each time the book is loaded, and it stores the paragraphs in a simple generic List object of type String.

This paragraph-concatenation logic falls apart in at least two cases: When an entire paragraph is indented, the program does not concatenate those lines, so each line is wrapped separately. The opposite

Code download available at code.msdn.microsoft.com/mag201107UIFrontiers.

Figure 1 **The Content Panel in MainPage.xaml**

```
<phone:PhoneApplicationPage.Resources>
  <local:MiddlemarchPageProvider x:Key="pageProvider" />
</phone:PhoneApplicationPage.Resources>
...
<Grid x:Name="ContentPanel" Grid.Row="1">
  <local:BookViewer x:Name="bookViewer"
                    PageProvider="{StaticResource pageProvider}"
                    PageChanged="OnBookViewerPageChanged">
    <local:BookViewer.PageTransition>
      <local:SlideTransition />
    </local:BookViewer.PageTransition>
  </local:BookViewer>

  <ListBox Name="chaptersListBox"
           Visibility="Collapsed"
           Background="{StaticResource PhoneBackgroundBrush}"
           ItemsSource="{Binding Source={StaticResource pageProvider},
                                 Path=Book.Chapters}"
           FontSize="{StaticResource PhoneFontSizeLarge}"
           SelectionChanged="OnChaptersListBoxSelectionChanged">
    <ListBox.ItemTemplate>
      <DataTemplate>
        <Grid Margin="0 2">
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
          </Grid.ColumnDefinitions>

          <TextBlock Grid.Column="0"
                     Text="&#x2022;"
                     Margin="0 0 3 0" />

          <TextBlock Grid.Column="1"
                     Text="{Binding Title}"
                     TextWrapping="Wrap" />
        </Grid>
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
</Grid>
```

problem occurs when a book contains poetry rather than prose: The program mistakenly concatenates those lines. These problems are impossible to avoid without examining the semantics of the actual text, which is fairly difficult without human intervention.

A New Chapter

Most books are also divided into chapters, and allowing a user to jump to a particular chapter is an important feature of an e-book reader. In EmmaReader, I ignored chapters, but the PageProvider class of MiddlemarchReader includes a GenerateChapters method that determines where each chapter begins. Generally, a Project Gutenberg file delimits chapters with three or four blank lines, depending on the book. For "Middlemarch," three blank lines precede a line indicating the chapter title. The GenerateChapters method uses this feature to determine the particular paragraph where each chapter begins.

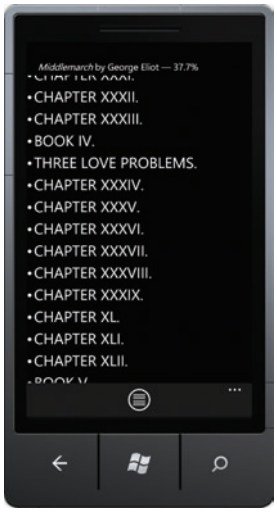


Figure 2 **The Scrollable List of Chapters**

Dividing a book into chapters helps alleviate the pagination problem in an e-book reader. For example, suppose a reader is near the end of a book and decides to change the font size. (Neither EmmaReader nor MiddlemarchReader have this feature, but I'm speaking theoretically.) Without chapter divisions, the entire book up to that point needs to be repaginated. But if the book is divided into chapters, only the current chapter needs to be repaginated. Back in the days of mechanical type, division of a book into chapters helped the typesetters in exactly the same way when lines had to be inserted or deleted.

The MiddlemarchReader program saves information about the book in isolated storage. The main class for persisting this information is called AppSettings. AppSettings references a BookInfo object that contains a collection of ChapterInfo objects, one for each chapter. These ChapterInfo objects contain the title of the chapter and a collection of PageInfo objects for that chapter. The PageInfo objects indicate where each page begins based on a ParagraphIndex that references the List of strings for each paragraph, and a CharacterIndex within that indexed string. The first PageInfo object for a chapter is created in the GenerateChapters method I described previously. Subsequent PageInfo objects are created and accumulated as the chapter is progressively paginated as the user reads that chapter. Also saved in BookInfo is the user's current page, identified as a chapter index and a page index within that chapter.

"Middlemarch" has 86 chapters, but the logic in the GenerateChapters method in PageProvider finds 110 chapters, including titles for the division of "Middlemarch" into eight "books," and material at the beginning and end of the file. Formatted for the phone, the novel is approximately 1,800 pages, or about 20 pages per chapter.

Figure 1 shows the content panel in MainPage.xaml. There are two controls that occupy the single-cell Grid, but at any time only one of them is visible. Both controls contain bindings to an object of type MiddlemarchPageProvider defined as a resource. I'll discuss the BookViewer control shortly. The ListBox displays a scrollable list of chapters, and you invoke it with a button on the program's ApplicationBar. **Figure 2** shows the list of chapters in MiddlemarchReader, scrolled to about the middle.

I wouldn't have chosen to display these headings and titles with all capital letters, but that's how they appear in the original file, and the program blindly pulls them out based solely on these lines being preceded by at least three blank lines.

The Back Page Problem

Like EmmaReader, MiddlemarchReader paginates on demand as the user reads the book, and the page information is stored so the user can page backward. No problem there. MiddlemarchReader also allows the user to jump to a particular chapter. No problem there, either, because the program already knows where each chapter begins.

However, suppose the user jumps to the beginning of a chapter and then decides to page backward to the last page of the previous chapter. If the previous chapter has not yet been paginated, that entire chapter must be paginated to show that last page. The paragraph might be short, or it could be quite long, depending on the book and the chapter. That's a problem.

Although MiddlemarchReader paginates on demand, it actually goes a little beyond that. When the user is reading one page, the next page and the previous page are ready for paging forward or back. Normally, obtaining these additional pages isn't a problem. But what should the program do if the user jumps to the beginning of a chapter? Should the program get the last page of the previous chapter to be ready for the possibility of paging backward? That's wasteful: In most cases, the user will only page forward, so why bother?

Obviously, it gets a little tricky. I've already mentioned that the PageProvider class is responsible for parsing the original Project Gutenberg file and dividing it into paragraphs and chapters, but it's also responsible for pagination. The MiddlemarchPageProvider class referenced in **Figure 1** is tiny. It derives from PageProvider and simply accesses the "Middlemarch" file so it can be processed by PageProvider.

I wanted the BookViewer control to have as little knowledge of the internals of PageProvider as possible. For that reason, the PageProvider property of BookViewer is of type IPageProvider (an interface implemented by PageProvider), as shown in **Figure 3**.

Here's how it works: The BookViewer object informs PageProvider of the size of each page and the font and font size to be used for creating the TextBlock elements that comprise the page. Most of the time, BookViewer obtains pages by calling GetPage. If GetPage

Figure 3 **The IPageProvider Interface**

```
public interface IPageProvider
{
  void SetPageSize(Size size);

  void SetFont(FontFamily fontFamily, double FontSize);

  FrameworkElement GetPage(int chapterIndex, int pageIndex);

  FrameworkElement QueryLastPage(int chapterIndex, out int pageIndex);

  FrameworkElement GetLastPage(int chapterIndex, out int pageIndex);
}
```

returns null, the chapter index or page index is out of range and that page does not exist. This could indicate to BookViewer that it's gone beyond the end of a chapter and needs to advance to the next chapter. **Figure 4** shows MiddlemarchReader at the beginning of a chapter. (Each chapter in "Middlemarch" begins with an epigraph, some of them written by George Eliot herself.)

When BookViewer navigates to the beginning of a chapter, IPageProvider defines two methods for obtaining the last page of the previous chapter. QueryLastPage is the fast method. If the previous chapter has already been paginated, and that last page is available, then the method returns the page and sets the page index. If the page is not available, QueryLastPage returns null. This is the method that BookViewer calls to obtain the previous page when the user navigates to the beginning of a chapter.

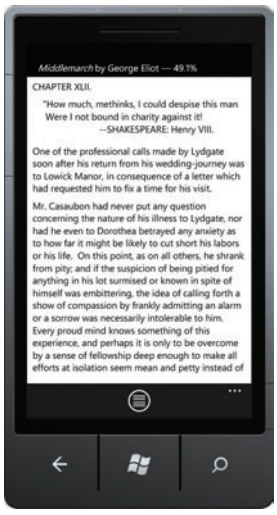


Figure 4 **The BookViewer Control Displaying a Page**

If QueryLastPage returns null and the user then pages back to that missing page, BookViewer has no recourse but to call GetLastPage. If necessary, GetLastPage will paginate an entire chapter just to obtain the last page.

Why not paginate the previous chapter in a second thread of execution after the user navigates to the beginning of a chapter? Perhaps the second thread will be finished by the time the user decides to page back. Although that's certainly a plausible solution, the pagination logic would need to be restructured a bit because it's creating StackPanel and TextBlock objects, and these can't be used by the primary thread.

The BookViewer Transitions

As I've mentioned, in the general case, BookViewer is juggling three pages at once. The control derives from UserControl, and **Figure 5** shows the XAML file. The three Border elements with names of pageHost0, pageHost1 and pageHost2 are used as parents of the pages obtained from PageProvider. (These pages are actually StackPanel elements with a TextBlock for each paragraph on the page.) The other three Border elements with names beginning with pageContainer provide a white background and a little margin of white around the borders of the page. These are also the elements manipulated for page transitions. A GestureListener (available in the Silverlight for Windows Phone Toolkit, downloadable from CodePlex at bit.ly/cB8hxu) provides touch input.

When BookViewer is created, it stores the three pageHost objects in an array named pageHosts. A field named pageHostBaseIndex is set to 0. As the user pages through a book, pageHostBaseIndex

Figure 5 **The XAML File for BookViewer**

```
<UserControl x:Class="MiddlemarchReader.BookViewer"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=
               Microsoft.Phone.Controls.Toolkit">

  <UserControl.Resources>
    <Style x:Key="pageContainerStyle" TargetType="Border">
      <Setter Property="BorderBrush" Value="Black" />
      <Setter Property="BorderThickness" Value="1" />
      <Setter Property="Background" Value="White" />
    </Style>

    <Style x:Key="pageHostStyle" TargetType="Border">
      <Setter Property="Margin" Value="12, 6" />
      <Setter Property="CacheMode" Value="BitmapCache" />
    </Style>
  </UserControl.Resources>

  <Grid x:Name="LayoutRoot">
    <toolkit:GestureService.GestureListener>
      <toolkit:GestureListener GestureBegin="OnGestureListenerGestureBegin"
                               GestureCompleted=
                                 "OnGestureListenerGestureCompleted"
                               Tap="OnGestureListenerTap"
                               Flick="OnGestureListenerFlick"
                               DragStarted="OnGestureListenerDragStarted"
                               DragDelta="OnGestureListenerDragDelta"
                               DragCompleted="OnGestureListenerDragCompleted" />
    </toolkit:GestureService.GestureListener>

    <Border Name="pageContainer0" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost0" Style="{StaticResource pageHostStyle}" />
    </Border>

    <Border Name="pageContainer1" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost1" Style="{StaticResource pageHostStyle}" />
    </Border>

    <Border Name="pageContainer2" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost2" Style="{StaticResource pageHostStyle}" />
    </Border>
  </Grid>
</UserControl>
```

Figure 6 **The PageTransition Class**

```
public abstract class PageTransition : DependencyObject
{
  public static readonly DependencyProperty FractionalBaseIndexProperty =
    DependencyProperty.Register("FractionalBaseIndex",
      typeof(double),
      typeof(PageTransition),
      new PropertyMetadata(-1.0, OnTransitionChanged));

  public double FractionalBaseIndex
  {
    set { SetValue(FractionalBaseIndexProperty, value); }
    get { return (double)GetValue(FractionalBaseIndexProperty); }
  }

  public virtual double AnimationDuration
  {
    get { return 1000; }
  }

  static void OnTransitionChanged(DependencyObject obj,
                                  DependencyPropertyChangedEventArgs args)
  {
    (obj as PageTransition).OnTransitionChanged(args);
  }

  void OnTransitionChanged(DependencyPropertyChangedEventArgs args)
  {
    double fraction = (3 + this.FractionalBaseIndex) % 1;
    int baseIndex = (int)(3 + this.FractionalBaseIndex - fraction) % 3;
    ShowPageTransition(baseIndex, fraction);
  }

  public abstract void Attach(Panel containerPanel,
                              FrameworkElement pageContainer0,
                              FrameworkElement pageContainer1,
                              FrameworkElement pageContainer2);

  public abstract void Detach();

  protected abstract void ShowPageTransition(int baseIndex, double fraction);
}
```

Figure 7 **The SlideTransition Class**

```
public class SlideTransition : PageTransition
{
  FrameworkElement[] pageContainers = new FrameworkElement[3];
  TranslateTransform[] translateTransforms = new TranslateTransform[3];

  public override double AnimationDuration
  {
    get { return 500; }
  }

  public override void Attach(Panel containerPanel,
                              FrameworkElement pageContainer0,
                              FrameworkElement pageContainer1,
                              FrameworkElement pageContainer2)
  {
    pageContainers[0] = pageContainer0;
    pageContainers[1] = pageContainer1;
    pageContainers[2] = pageContainer2;

    for (int i = 0; i < 3; i++)
    {
      translateTransforms[i] = new TranslateTransform();
      pageContainers[i].RenderTransform = translateTransforms[i];
    }
  }

  public override void Detach()
  {
    foreach (FrameworkElement pageContainer in pageContainers)
      pageContainer.RenderTransform = null;
  }

  protected override void ShowPageTransition(int baseIndex, double fraction)
  {
    int nextIndex = (baseIndex + 1) % 3;
    int prevIndex = (baseIndex + 2) % 3;

    translateTransforms[baseIndex].X = -fraction *
      pageContainers[prevIndex].ActualWidth;
    translateTransforms[nextIndex].X = translateTransforms[baseIndex].X +
      pageContainers[baseIndex].ActualWidth;
    translateTransforms[prevIndex].X = translateTransforms[baseIndex].X -
      pageContainers[prevIndex].ActualWidth;
  }
}
```

goes to 1, then 2, then back to 0. As the user pages backward, pageHostBaseIndex goes from 0 to 2, then 1, then back to 0. At any time, pageHosts[pageHostBaseIndex] is the current page. The next page is:

```
pageHosts[(pageHostBaseIndex + 1) % 3]
```

The previous page is:

```
pageHosts[(pageHostBaseIndex + 2) % 3]
```

With this scheme, once a particular page has been set in a page host, it stays there until it's replaced. The program doesn't need to transfer pages from one page host to another.

This scheme also allows fairly simple page transitions: Just use Canvas.SetZIndex so that pageHosts[pageHostBaseIndex] is on top of the other two. But that's probably not the way you want to transition between pages. It's rather bland and even somewhat hazardous. The imprecision of touch input is such that the user might accidentally move two pages ahead instead of one, and not even realize it. For this reason, more dramatic page transitions are desirable.

BookViewer achieves great flexibility with regard to page transitions by defining a PageTransition property of type PageTransition, an abstract class shown in **Figure 6**.

PageTransition defines a single dependency property named FractionalBaseIndex that BookViewer is responsible for setting based on touch input. If the user taps the screen, FractionalBaseIndex is increased by a DoubleAnimation from pageHostBaseIndex to pageHostBaseIndex plus 1. Animations are also triggered if the user flicks the screen. If the user drags a finger along the screen, FractionalBaseIndex is set "manually" based on the distance the user has dragged his finger as a percentage of the total width of the screen.

The PageTransition class separates FractionalBaseIndex into an integer baseIndex and a fraction for the benefit of derived classes. The ShowPageTransition is implemented by derived classes in a way that's characteristic of the particular transition. The default is SlideTransition, shown in **Figure 7**, which slides the pages back and forth. **Figure 8** shows a page being slid into view. The class attaches TranslateTransform objects to the page containers during the Attach call and removes them during Detach.

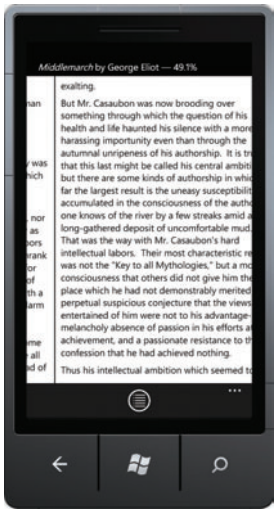


Figure 8 **A Page Sliding into View**

You can select the other two page transitions from the application bar menu: FlipTransition is similar to SlideTransition, except that it uses the PlaneProjection transform for a 3D look. The CurlTransition displays the page as if the upper-right corner is pulled back. (I originally wrote the CurlTransition code to curl from the lower-right corner, and was able to convert it simply by adjusting all the horizontal coordinates. You can change it back to the lower-right curl by setting the curlFromTop constant to false and recompiling.)

Because MiddlemarchReader allows the user to jump to the beginning of a chapter, displaying actual page numbers at the top of the program is no longer feasible. Instead, the program displays percentages based on text lengths.

Fear of Pagination

In actual use, MiddlemarchReader is starting to look and feel much like a real e-book reader. However, it still suffers from the poor performance of its pagination logic. Indeed, the first page of the first chapter of "Middlemarch" is delayed a bit when the program is running on a Windows Phone 7 device because the chapter begins with a paragraph that goes on for several pages. Paging back after jumping to a new paragraph sometimes requires a second or so, and I haven't yet been brave enough to introduce user-selected fonts or font sizes into the program because these features require repagination.

Is there a way to speed up the pagination? Perhaps. Undoubtedly such an improvement would be elegant, clever and more difficult than I anticipate. ■

CHARLES PETZOLD *is a longtime contributing editor to* MSDN Magazine. *His recent book, "Programming Windows Phone 7" (Microsoft Press, 2010), is available as a free download at bit.ly/cpebookpdf.*

DAVID PLATT

When Security Doesn't Make Sense

I just finished reading a truly brilliant research paper, "So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users," by Cormac Herley of Microsoft Research. Every one of you needs to read the whole thing, which is online at bit.ly/IZZsyr.

How often have we given users security instructions and had them ignore us? And then we got mad when our beautiful security code didn't prevent losses because the users wouldn't do what we told them to? Bad, naughty users, we said. It's your own dumb fault you got hurt.

Wrong. It's our fault for telling them to do things that we knew, or should have known, that they wouldn't do.

> An ounce of cure is not worth five pounds of prevention.

According to Herley, users who ignore our security instructions are being rational from their point of view. They subconsciously calculate that the constant efforts we demand of them are greater than the infrequent (albeit larger) losses to them if they don't follow our instructions. They then rationally decide to ignore us. Herley writes: "Consider an exploit that affects 1 percent of users annually, and they waste 10 hours clearing up when they become victims. Any security advice should place a daily burden of no more than 0.98 seconds per user in order to reduce rather than increase the [total] amount of user time consumed. This generates the profound irony that much security advice not only does more harm than good (and hence is rejected), but does more harm than the attacks it seeks to prevent, and fails to do so only because users ignore it." An ounce of cure is not worth five pounds of prevention.

A user will tolerate only so much security-related (or other) overhead before he either dumps your product or figures out a workaround. I call this amount the user's "hassle budget," a term I coined in my book, "Why Software Sucks" (Addison-Wesley Professional, 2006).

Example: Suppose your landlord put a combination lock on the bathroom door in your apartment. What would you do? You'd enter the combination the first time and maybe the second, but definitely not the third. After that you'd find some sort of workaround—you'd prop the door open, you'd tape down the latch so it wouldn't lock or you'd relieve yourself in the kitchen sink.

I recently saw a Web article entitled "37 Tips to Prevent ID Theft Online." If I have to remember 37 different items to keep my identity safe online, the bad guys can have the damn thing.

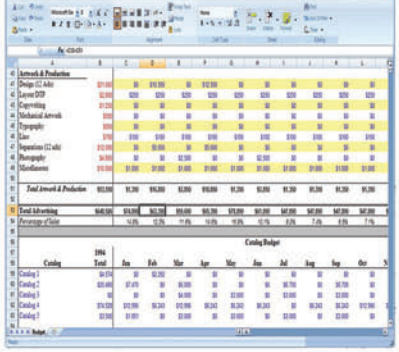
Herley applies rigorous cost-benefit analysis to such common security practices as changing passwords regularly. You're somewhat safer if you do this, but how much? And is that benefit greater or lesser than the cost of the time that you spend changing them and keeping track of them? You'll probably get better overall results if you spend a user's hassle budget ensuring that his initial password is strong, rather than on periodic changes.

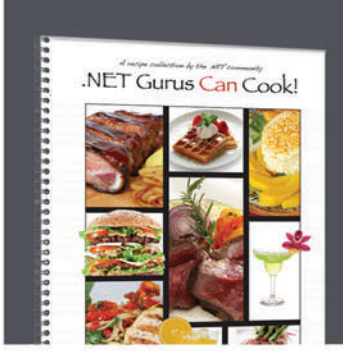
I've seen lots of security advice, but this is the first time I've seen anyone compare the cost of following that advice with the harm avoided by doing so. When you start putting the two together, a much more nuanced picture emerges. You can only understand it if you put yourself in your user's shoes—if you Know Thy User, Because He Is Not Thee. (Where have I heard that before?)

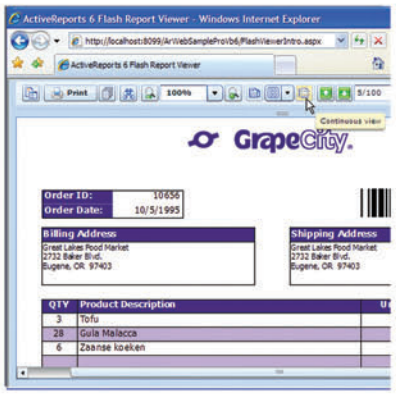
I'll leave you with this final thought from Herley, which I very much hope convinces you to read his entire paper:

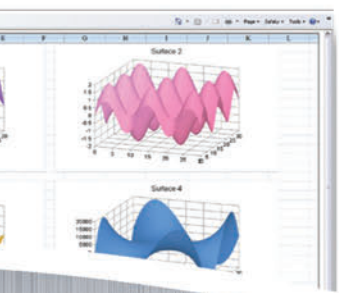
"There are about 180 million online adults in the U.S. At twice the U.S. minimum wage, one hour of user time is then worth $7.25 x 2 x 180e6 = $2.6 billion. … We suggest that the main reason security advice is ignored is that it makes an enormous miscalculation: it treats as free a resource that is actually worth $2.6 billion an hour. It's not uncommon to regard users as lazy or reluctant. A better understanding of the situation might ensue if we viewed the user as a professional who bills at $2.6 billion per hour, and whose time is far too valuable to be wasted on unnecessary detail." ■

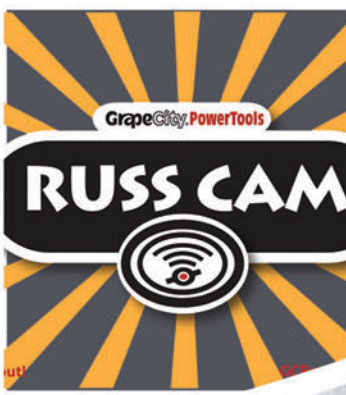
David S. Platt teaches Programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him 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.











Start building smarter applications

WE ARE REPORTING

GrapeCity ActiveReports is one of a suite of award-winning reporting tools and components for ASP.NET, WinForms, and Silverlight with unmatched features, fast performance, and high quality.
>re•port

WE ARE SPREADSHEETS

GrapeCity Spread is the world's best selling, award-winning suite of Microsoft Excel® compatible spreadsheet components and tools for ASP.NET, WinForms, COM and BizTalk.
>ex•cel

WE ARE ANALYSIS

GrapeCity ActiveAnalysis is the world's first free-form OLAP, data visualization and business analysis component with out-of-the-box end-user interface for the WinForms, ASP.NET and Silverlight.
>an•a•lyze

Download your FREE trial today.

GrapeCity PowerTools

www.GCPowerTools.com

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